# MyMED: A database system for biomedical research on MEDLINE data

by K. N. Lewis
M. D. Robinson
T. R. Hughes
C. W. V. Hogue

MyMED is a database system that mirrors MEDLINE™ data and that provides access to the data locally, without the restrictions involved in accessing the National Library of Medicine Web site. The implementation of MyMED is based on the IBM DB2 Universal Database™, the DB2® XML Extender and the DB2 Text Information Extender. The resulting system supports advanced queries, including wildcard searching abilities, proximity searching, scoring functions, and capabilities to build a thesaurus, and constitutes an important research tool for text mining of biomedical-citation data. The paper describes the challenges encountered in the use of DB2 Extenders™ products, the methods used to overcome these challenges, and the implementation of MyMED. The benefits of MyMED are illustrated through some use cases, and directions for future work are outlined.

The abundance of information contained in citations to, and abstracts of, biomedical journal articles is so great that text mining these data is becoming an essential tool in the field of bioinformatics. The primary source of citation data for biomedical research is MEDLINE**, a citations database from the National Center of Biotechnology Information, a division of the National Library of Medicine (NLM).[1] Access to MEDLINE, which is available to the public through PubMed**, a Web-based database system, is limited in terms of the number of daily requests allowed from the same computer. Due to this restriction on accessing MEDLINE, text-mining tasks involving large numbers of queries may take days, or even weeks, to complete. MyMED, which was developed by the authors within the Blueprint Initiative,[2] is a database system that mirrors MEDLINE data and that provides access to the data locally, without restrictions. The implementation of MyMED is based on the IBM DB2 Enterprise Edition Universal Database* (DB2, for short) enabled for Extensible Markup Language (XML), the DB2 XML Extender (or XML Extender, for short) and the DB2 Text Information Extender (TIE). Citation data in XML format are extracted from NLM-supplied XML files using the XML Extender and stored in the database, and then selected XML text fields are indexed using TIE.

The resulting system supports advanced queries including wildcard searching abilities, proximity searching, scoring functions, and capabilities to build a thesaurus, and constitutes an important research tool for text mining of biomedical-citation data. Text-mining algorithms can vary over a wide range, from simple (such as the recognition of specific terms in text), to complex (such as identifying complex relationships within sentences). An example of the latter is the use of machine-learning techniques to identify abstracts describing protein-protein interactions.[3]

Beyond removing the limitations in accessing MEDLINE data through PubMed, MyMED has many additional advantages. First, because the access is lo-




0018-8670/04/$5.00 © 2004 IBM

cal, the response times are much improved. Second, the confidentiality of the queries is assured. Third, whereas access to MEDLINE data is limited by the MEDLINE application programming interface (API), no such limitation exists in MyMED. Thus, MyMED incorporates additional functionality such as searching with queries that produce very large result sets. Fourth, through complex SQL (structured query language) queries users can manipulate result sets directly at the database level rather than by the additional step of post-processing the data obtained from PubMed. Fifth, MyMED supports large automated batch queries and the return of very large result sets at the user's convenience. Last, through the use of XML and Extensible Stylesheet Language Transformations[4] (XSLT), MyMED allows the user to create personalized views of the data on Web pages and to render the result of a query in different formats, rather than being limited to the formats available from the MEDLINE API.

**MyMED not only removes access limitations to MEDLINE data, but also improves query response times.**

**The data.** MEDLINE consists of more than twelve million biomedical bibliographic citations published from the mid 1960s to the present and distilled from more than 4600 biomedical journals published in over 70 countries. The National Library of Medicine leases MEDLINE citation data in XML format. MEDLINE citations include bibliographic information such as article title, abstract, author list, and journal information, but do not contain full-text articles. The 2003 baseline data contains all maintenance that was performed in 2002, and all XML content conforms to the November 2002 DTD (document type definition) version. It is available either on a Digital Linear Tape (DLT) or via File Transfer Protocol (FTP). It consists of 11 847 524 records packaged in 396 XML files and requires approximately 42.9 GB of disk space.[5]

NLM makes updates to the baseline data available five days a week via FTP. The updates contain new, revised, deleted, and in-process records. NLM projected adding approximately 535 000 MEDLINE citations during 2003. Update files must be processed in chronological order to ensure that the correct version of any given record is obtained.

**DB2 and the DB2 Extenders.** DB2 is a relational database management system that provides a secure and efficient way to store and query data. To retrieve or update the data in the database, one uses commands formulated as SQL statements.

The XML Extender, which works in conjunction with the database, provides additional data types and functions that extend the capabilities of DB2 to include XML data. In particular, the XML Extender provides the XML Collection method and the XML Column method. Whereas the XML Collection method stores the individual fields from an XML file into database table columns, the XML column method stores the entire XML document in a table column. These methods make use of a document access definition (DAD) file—an XML document that describes the access and storage methods that the database is using. It also contains the location of the DTD files and describes how the XML data map to the DB2 columns in the tables.

Like the XML Extender, TIE is also installed separately and is enabled to work with the specified database. It provides a number of functions for indexing fields, searching fields, building a thesaurus, and updating indexes—resulting in full-text search capabilities. The combination of these products results in a database system that can perform complex text queries for text-mining applications.

**Challenges.** Some of the challenges involved in developing MyMED arose from the data. In the MEDLINE DTD files there are multiple formats for some XML elements. For example, the journal issue publication date can be in the form ⟨year, month, day⟩ or ⟨year, season⟩ or in MedlineDate format, a format used when the date of publication does not fit into any other formats. This proved to be beyond the ability of the XML Collection method (the insertion of records into the database resulted in error). An attempt to use the XML Column method for this task led to the creation of a separate table for each different format, which resulted in the creation of a very large number of tables. Then, the queries involved a large number of SQL joins which resulted in slow response times.

The XML files use UTF-8 (8-bit Unicode Transformation Format), encoded in Unicode Normalized Form C.[5] The MEDLINE data contain citations from 49 different languages using many Unicode-encoded characters. This caused errors when using the XML Extender functions until the codeset of the database was explicitly set to UTF-8 encoding.

Challenges also arose from continuous changes to the MEDLINE DTD files. The MEDLINE DTDs describe approximately 112 elements, 21 entities, and 11 attribute lists and are under constant revision. MEDLINE records are associated with three DTDs: the NLMMedline DTD, the NLMCommon DTD and the NLMMedlineCitation DTD (the NLMMedline DTD is the parent DTD[6]). The DTD files are structured so that files created using previous DTD versions can also be validated using the newest DTD available.

Changes to the MEDLINE DTD files by NLM include the renaming, addition, and deletion of XML elements. The table definitions and DAD file in the Collection or Column methods required constant updating in such cases. Thus, changes in the DTD files meant that at any given time the database contained citations defined by multiple DTD versions. This meant that the database system had to handle documents associated with more than one DTD version.

The MEDLINE baseline data gets updated once a year to the most current DTD version. This means the MyMED database is rebuilt once a year with a new DTD version; thus, the database build needs to be able to handle new DTD versions every year. Finally, the daily updates to the database records mean that these transactions (creating, updating, and deleting, including text index updates) must have acceptable execution times.

In principle, the DB2 XML Extender and DB2 TIE Extender should be straightforward to use. With third-party data and DTD files that are controlled by other organizations, we found that this configuration was not sufficient to support the dynamic nature of MEDLINE data. Experience revealed many exceptions that these tools were not able to deal with, such as the inability to process files containing a large number of citations and incompatibilities with the MEDLINE DTD details. The XML Extender is still not capable of extracting 30000 citations from a file (the MEDLINE baseline data is packaged as 390 XML files, each containing exactly 30000 citations). Consequently, the files must be preprocessed into smaller files, each containing approximately 1000 citations (it was determined that this was the largest number of entries per file that could be processed without errors).

The next section describes the MyMED implementation, including the methods used to overcome the challenges discussed earlier. In particular, it includes a description of the "lower-level column method," a method developed to overcome the difficulties encountered in the direct application of the XML Extender. The section that follows describes the results obtained through the use of MyMED and directions for future work. The paper concludes with a short summary.

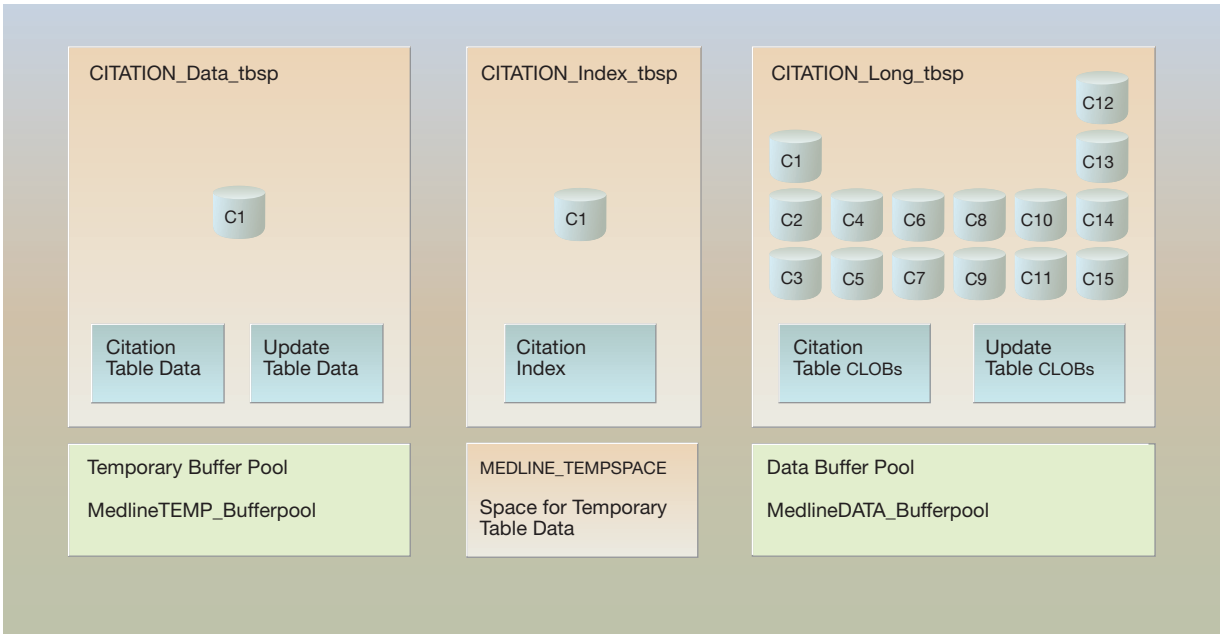
## Implementation

This section describes the implementation of MyMED including the hardware and software used, configuring the database, building the citation table, building the text index, updating the database, and the lower-level column method.

**Software and hardware.** The initial implementation used an IBM RS/6000* M80 computer with 32 GB of RAM, eight RS64 III 500 MHz CPUs, and the operating system AIX* v4.3. Other implementations included a Sun Microsystems Enterprise 450 machine with 1.5 GB of RAM, four UltraSPARC II 300 MHz CPUs, and the Solaris** Version 8 operating environment. The current implementation utilizes IBM DB2 Enterprise Edition Version 7.2 with fixpak 10 and is installed on a Sun Microsystems SunFire** V880 with 16 GB of RAM and eight UltraSPARC III 900 MHz CPUs running the Solaris Version 9 operating environment. All available fixpaks were installed along with the DB2 XML Extender Version 7 with fixpak 10 and the DB2 TIE with fixpak level 2. An installation of Perl[7] with the Perl Database Interface and DB2 modules also proved useful for some query scripts. The XML Extender can operate on the AIX, Linux**, Solaris, Microsoft Windows** 2000, and Microsoft Windows NT** operating systems. Because the TIE supports AIX, Solaris, Windows NT, and Windows 2000 systems only, to text-index documents on a Linux operating system it is necessary to use the IBM DB2 Net Search Extender*[8] (NSE) rather than the TIE. The NSE is the newest text extender product from IBM. It incorporates the TIE and includes further optimization abilities and scalability with query load.

**Configuring the database.** The buffer pool database configuration is used to cache table and index pages

Figure 1 MyMED database organization




in memory and has a large impact on system performance. Tuning this parameter has the greatest effect on performance. When a database is created, one buffer pool is automatically defined. The default page size for the buffer-pool configuration is 4 KB. Because we calculated the optimal page size to be 16 KB, we created two additional buffer pools, each with a 16 KB page size, one for temporary tablespace data and one for regular and long tablespace data, MedlineTEMP_Bufferpool and MedlineDATA_Bufferpool, labeled as Temporary Buffer Pool and Data Buffer Pool in Figure 1.

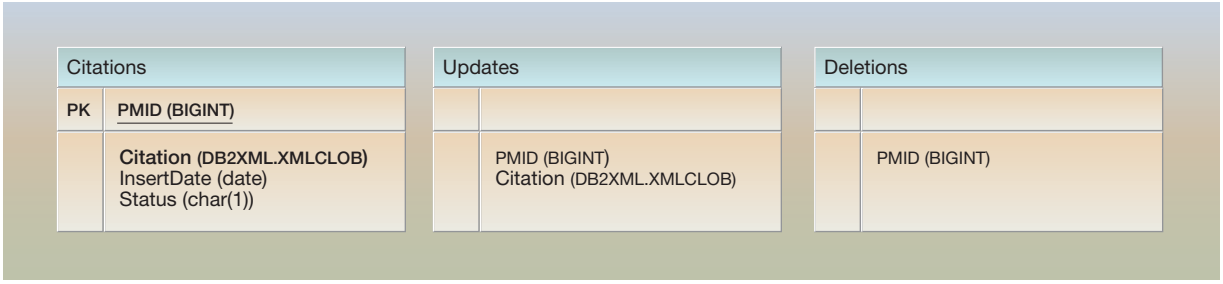
A *tablespace* consists of one or more containers (which can be either a file or some device used to store data) used for distributing data evenly. There are three distinct tablespace types: long tablespace (for columns defined as character large objects [CLOBs], or binary large objects [BLOBs]), temporary data tablespace, and regular data tablespace. For the MyMED database, four tablespaces, each with a 16 KB page size were created: one temporary tablespace, MEDLINE_TEMPSPACE, two regular tablespaces (one for data, CitationDATA_tbsp, and one for index data, CitationINDEX_tbsp, consisting of one container each), and one long tablespace—CitationLONG_tbsp—for the LOB (large object) data (consisting of 15 containers) as shown in Figure 1.

MEDLINE XML data files use UTF-8 encoding; therefore, the database must be created with a codepage equal to UTF-8.[9] If this is not set correctly, the XML Extender will fail to read and insert XML records into the database due to the presence of characters in the vernacular titles and author names that are not English.

After the database is created, there are a number of alterations to the database configuration as well as the database manager configuration that are necessary; some of these include changes to the default-query heap size, the default-application heap size, and the default log-file details. The MyMED database cannot be built with the default values due to the inadequate size of the files for insertion and the large number of rows in the database.

Three tables were created—the citation table, the update table, and the deletion table. Figure 2 shows the columns and column types for each table in UML**-style format[10] (UML stands for Unified Modeling Language). The citation table is composed of four columns—citation, PubMed Unique Identifier (PMID), status, and date. The citation column is defined as type db2xml.XMLClob and contains an XML fragment of one MEDLINE citation. Storing the data as a CLOB eliminated problems that arose due to the

Figure 2 MyMED database tables




dynamic length of citations—there is no enforced limit to their length. The PMID is an integer of up to eight digits in length that is present in all citations and cannot have an empty value. Additionally, PMID values are never reused after a record has been deleted. These attributes ensure the uniqueness of PMID; therefore, it is selected as the primary key for the citation table. The status column is stored as a CHAR of size 1; when a citation is initially inserted into the table, the status is set to "N" for new; if it is marked for deletion, it is set to "D." The date field is of the DATE data type and contains the date of the last change made to the corresponding record.

The update table is composed of two columns—citation and PMID. This table is used to temporarily store the updated and new citations from the daily update files before they are used to update the citation table. This table is necessary because the PMID value is not known until a second pass through the table with an SQL update statement that extracts it and updates the corresponding rows. Then the PMIDs can be compared to the citation table to determine if the resulting update is a new or revised record. Again, the citation column is of the type db2xml.XMLClob, and the PMID is stored as an integer.

The deletion table is composed of one column—a PMID column. This table is used to store the PMIDs of records indicated for deletion from the database as they occur in the daily update files.

**Building the citation table.** The files are used as input to an SQL insert statement that contains the db2xml.ExtractCLOB function—a defined function used by the XML Extender. The XML Extender proved to be unstable when processing files with a large number of citations (30 000). By experimenting with different record counts in the files, the optimum number of records per file to ensure insertion was 5000. The preprocessing of all the large files resulted in 2371 files of no more than 5000 citations each. Inserting the entries in these files into the citation table required 2371 SQL insert statements.

These SQL insert statements use the XML Extender function db2xml.ExtractCLOB that takes as input a file name and a path and then extracts an XML fragment found in the given path. Element paths can be found in the DTD files. The entire citation was extracted for storage in the citation column. In the current MyMED implementation, this initial insertion of the baseline data took approximately 8.5 hours on four parallel processors.

After all citations are inserted into the citation table, the PMIDs are extracted and inserted for each row because it is not possible to extract the CLOB and an element from within the CLOB at the same time. The PMID is extracted from each citation using the db2xml.ExtractInteger defined function. The column name and path to be extracted are arguments to the function, and the PMID is returned and inserted into the appropriate column for each row in the table. Duplicates resulting from human error, such as starting and stopping an insert script, were identified and removed from the table. The PMID was then set as the primary key, and the initial build of the table was concluded.

**Building and updating the text index.** Most medical abstract queries search for terms within the unstructured free text area—mostly the abstract and the title fields of a citation. This requires advanced indexing techniques of free text fields. The Medical Subject Heading[11] (MeSH**) terms (the NLM-controlled vocabulary thesaurus), abstracts, and titles were indexed so that advanced queries could be made on

these fields. TIE allows a single index to be created on a column and multiple XML element values to be indexed in a db2xml.CLOB, and therefore, any fields in the MEDLINE citation may be indexed. A document model is created which defines tags of interest using XML Path Language [12] (Xpath) expressions; this allows elements with the same name to be distinguished by the element's full path. The document model includes fields named "abstracts," "titles," and "mesh." Because a column can have only one text index specified for it, all of these are included in the same text index, but each text-indexed field can still be accessed separately. For example, it is possible to search for terms that occur in all of the indexed elements in the column as well as only in the MeSH section of the citations.

To create the index, the document model file needs to be written, and the path to it is used as input in the "create index" command. The directory paths for the index and working directories are specified as input as well. It is necessary to ensure that the fenced user has appropriate permissions and that they are part of the db2adm group in order for the updates to work properly. Because at the time the software could not handle this task, a manual change was required—this became apparent after troubleshooting this problem with an IBM team. During the "create index" command, it is also possible to specify the update frequency of the index. This can be set to occur on a regular day and time of the week or after a certain number of changes to the column. Because data updates were available daily, updates were set to occur regularly every evening. The "commit count" is another feature that can be specified during the creation of the index; it is the number of inserts or updates on the text column after which an intermediate commit statement is issued by DB2. This feature may be useful for large tables because it can avoid errors involving insufficient log space, but it does slow down the index build and update considerably. In this case, there was no commit-count value specified; therefore, all records were updated and committed for the whole table at once. The text index was created instantly. The "update index" command is used to build (in the initial call) and update (in subsequent calls) the text index. The initial update of the index for the 2003 baseline data took approximately 19 hours to complete for the current implementation. Subsequent update times varied with the number of records.

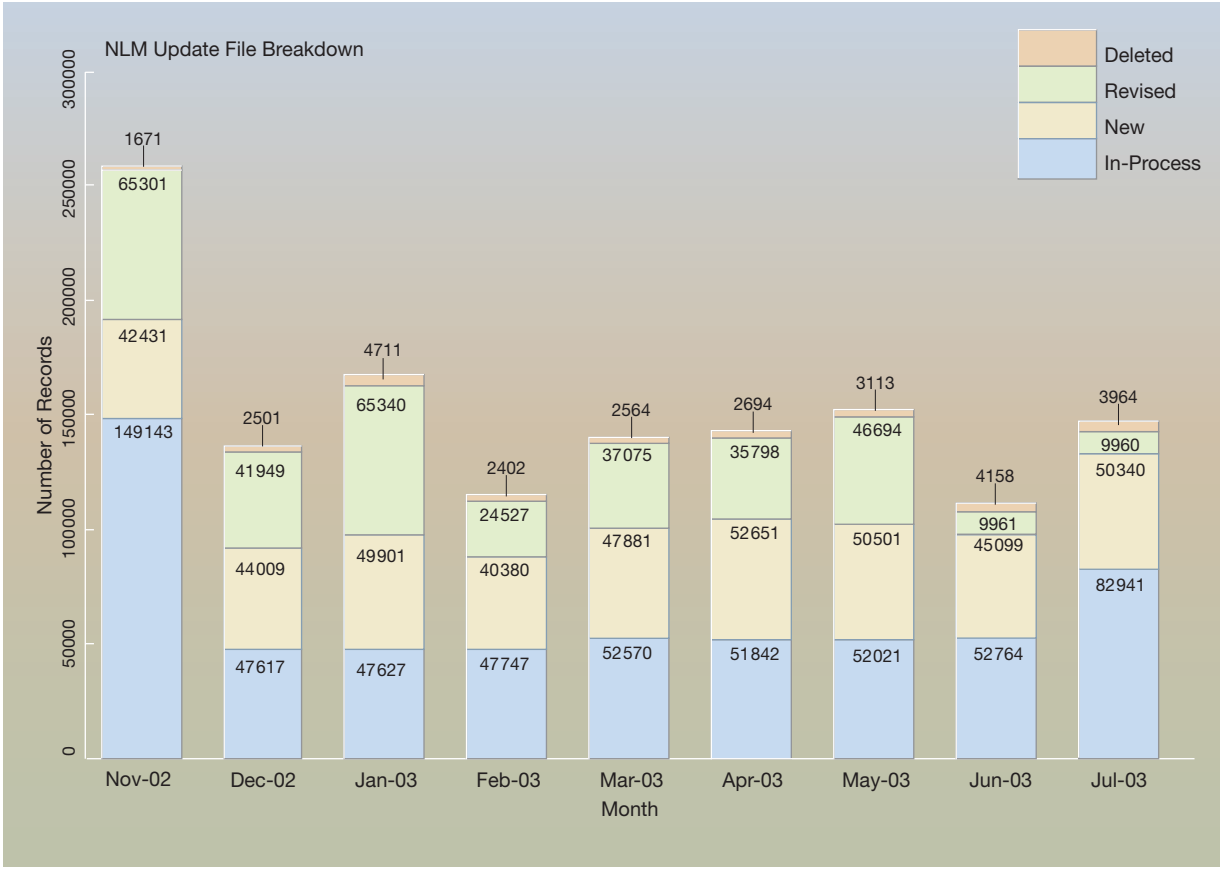
**Updating the database.** Update files are available five days a week through the NLM FTP site. Update files contain new, revised, in-process, and deleted citations. Details of the distribution of update records over the past nine months are shown in Figure 3. In-process records are not deemed complete records, and MeSH terms are missing. Update files must be processed in chronological order to ensure database integrity.

Update files are downloaded and preprocessed similarly to the baseline files. The citations are inserted into the update table, and PMIDs are updated. A second pass through the files inserts the deletion PMIDs into the deletion table. Both insert statements use the same XML Extender defined functions as the original build. When a row in the citation table has a PMID that matches a PMID in the update table, the row is updated; all other rows in the citation table are new records. All PMIDs in the deletion table are deleted from the citation table. Lastly the text index is updated at the scheduled time.

**The lower-level column method.** One of the largest hurdles in the MyMED implementation was working with the XML Extender. As previously mentioned, there are two access methods available for data in a database enabled for the XML Extender—the Column method and the Collection method. Initially the Collection method was attempted—this way every XML element was mapped to a column in a table. This resulted in many errors as there were some limitations regarding elements with the same name but different paths. The MEDLINE DTD contains numerous logical OR statements in various elements due to multiple formats for some fields such as author name or date formats. Moreover, during the first year that NLM released the XML data, the DTDs changed three times, which required a revision of the DAD file with every new release and the storage of documents following multiple DTD versions in the database. All these compounded the difficulties in using the Collection method.

The second approach attempted was the Column method. This method allows the insertion of the XML document in its entirety into a column along with the creation of side tables where element values can be simultaneously inserted based on location path. The column where the XML document is to reside must be enabled for XML and then can be searched, updated, and retrieved. Data in the side tables can then be indexed for fast searching. Similar problems were encountered based on the limitations that could not handle all the specifications in the DTD and resulted in a large number of side tables that slowed

Figure 3 NLM update files: November 2002 through July 2003




down query response time. Rules for the column DAD file specified that values with multiple element types must be in separate tables. This introduced an increased number of SQL joins during queries, which caused a slowdown in query response time and created difficulties in normalizing the database. Fixpak installations have updated and corrected various problems or documented limitations as they have become apparent, and these are available from the IBM Web site.[13]

The current implementation abandons both the traditional Collection and Column methods as described in the XML Extender administration and programming documentation[14] and invokes a type of manual column method or low-level column method that does not require a DAD file and is not as sensitive to changes in the DTD. The yearly update of the baseline data ensures that all the XML citations are in the same format and use the most current DTD. It is not necessary to enter or store the DTD, which means that there is no validation of the XML data, but it does allow one to extract and insert data in the database using this method.

There were several advantages to storing documents in columns as intact XML documents. First, information is not lost when parsing specific fields out. Second, any element can be accessed and retrieved in the document using the XML Extender functions, for example the title of a citation can be extracted using the following SQL statement:

```
SELECT db2xml.extractVarChar (cita-
tion,'/MedlineCitation/Article/Article
Title') FROM medline.citation WHERE
pmid = 433569
```

Third, documents can be stored with different DTD versions (this is necessary since only NLM has con-

trol over baseline data and changes to the DTD versions). Finally, because it is necessary to put a limit on column lengths when defining a table, this method of storing the entire citation as a db2xml.XMLClob type is sensitive only to the length of the entire citation, which can get as large as 2 GB without error. This is useful because the element lengths are not static and NLM does not provide character field-length information. Our experience of estimating field length resulted in fields (such as title or abstract) that were too small, which resulted in truncation of the field.

Using this new low-level column method as explained earlier, we tried two different approaches: 1) extracting the specific fields to text index, inserting them into a separate table, and indexing the fields in separate indexes, and 2) keeping the XML in the column and indexing the citation based on the XML path. To implement the first method, an embedded SQL script was written that took as input a PMID range and a commit-count value. One limitation was the inability to extract values consistently from more than 1000 records at a time without error. This was overcome by implementing a commit count in the embedded SQL code enabling the build to be automated. Three separate text indexes were created, meaning that three model documents were necessary to update and maintain three indexes. This method has two drawbacks. First, returning an entire record through text search required table joins that slowed performance. Second, the addition of a table to store the extracted indexed fields almost doubled the required disk space for the database. The preferred implementation was the second one, which involved having one main table with the citations in one column. It was simple, easy to build and maintain, and took up much less disk space.

## Results

The TIE has three defined search functions: CONTAINS, SCORE, and NUMBEROFMATCHES. The CONTAINS function returns the value 1 if a search argument is found in the specified text field and the value 0 if none is found. The SCORE function searches for an argument in the specified text field and returns a numerical value between 0 and 1 indicating how well the document can be described by the search argument.[15] The NUMBEROFMATCHES function returns an integer describing the number of times the search argument is found in each document.

The search argument has many options including:

- Specifying a thesaurus to use if one has been created,
- A return result limit,
- Boolean arguments or free text arguments,
- Section of the index to search,
- Proximity searching,
- Precise searching,
- Fuzzy searches,
- Stemmed searches, and
- Language specification.

The first query in Table 1 searches for the terms 'cardiovascular' and 'Disease' in the index found for the column CITATION in the MEDLINE.CITATIONS table. For matching documents it returns the PMID and the title. Selection of the title uses the XML Extender function db2xml.extractVarchar ( ). The CONTAINS ( ) function in the where clause is a TIE function. More example queries using the XML Extender functions and TIE functions can be found in Table 1.

**An experiment that resulted in more than a million citation matches and took 21 hours on MyMED would have taken several weeks on PubMed.**

Queries containing 7970 aliases of gene names along with the keywords "mouse" or "musculus" were performed to create an index of PMIDs associated with each gene represented by a spot on a microarray. A total of 141 231 citations were found containing the search terms in the abstract, title, or MeSH sections. Citations with matching terms, occupying about 1.3 GB in disk space, were returned in tab-delimited format containing the PMID with the entire XML citation in approximately four hours of running eight searches in parallel. The same aliases queried along with keywords "human" or "sapiens" returned 1 424 588 citation matches. Due to the large number of matches, we returned the citation PMID without the entire citation itself in approximately nine hours running eight searches in parallel. In testing we also found we could return entire citations in 21 hours. When PubMed was queried for a similar alias list along with "yeast," "saccharomyces," or "cerevisiae," it took a few weeks to gather all the results using several different IP addresses, given the restric-

Table 1 MyMED example queries

| | |
|---|---|
| Searching for terms in any sequence | SELECT PMID, db2xml.extractVarchar(citation, '/MedlineCitation/Article/ArticleTitle')<br>FROM MEDLINE.CITATIONS<br>WHERE CONTAINS (CITATION, '("cardiovascular", "Disease")') = 1 |
| Searching for terms in fixed sequence | (SELECT PMID, db2xml.extractVarchar(citation, '/MedlineCitation/Article/ArticleTitle')<br>FROM MEDLINE.CITATIONS<br>WHERE CONTAINS CITATION, '("cardiovascular disease")') = 1 |
| Searching for terms in the same sentence | SELECT PMID, db2xml.extractVarchar(citation, '/MedlineCitation/Article/ArticleTitle')<br>FROM MEDLINE.CITATIONS<br>WHERE CONTAINS (CITATION, '"Cardiovascular" IN SAME SENTENCE AS "Disease"') = 1 |
| Fuzzy searches | SELECT PMID, db2xml.extractVarchar(citation, '/MedlineCitation/Article/ArticleTitle')<br>FROM MEDLINE.CITATIONS<br>WHERE CONTAINS (CITATION, 'FUZZY FORM OF 70 "Cardiovascular"' = 1 |
| Precise searches | SELECT PMID, db2xml.extractVarchar(citation, '/MedlineCitation/Article/ArticleTitle')<br>FROM MEDLINE.CITATIONS<br>WHERE CONTAINS (CITATION, 'PRECISE FORM OF "Coronary Disease"' = 1 |
| Stemmed searches | SELECT PMID, db2xml.extractVarchar(citation, '/MedlineCitation/Article/ArticleTitle')<br>FROM MEDLINE.CITATIONS<br>WHERE CONTAINS (CITATION, 'STEMMED FORM OF "Mouse" & "Coronary"' = 1 |
| Searching with wildcard (%) | SELECT PMID, db2xml.extractVarchar(citation, '/MedlineCitation/Article/ArticleTitle')<br>FROM MEDLINE.CITATIONS<br>WHERE CONTAINS (CITATION, '"%ronary"') = 1 |
| Searching for one character (_) | SELECT PMID, db2xml.extractVarchar(citation, '/MedlineCitation/Article/ArticleTitle')<br>FROM MEDLINE.CITATIONS<br>WHERE CONTAINS (CITATION, '"_oronary"') = 1 |
| Searching with Boolean (OR) | SELECT PMID, db2xml.extractVarchar(citation, '/MedlineCitation/Article/ArticleTitle')<br>FROM MEDLINE.CITATIONS<br>WHERE CONTAINS (CITATION, '"Coronary"\|"Disease"') = 1 |
| Searching for both terms (AND) | SELECT PMID, db2xml.extractVarChar(citation, '/MedlineCitation/Article/ArticleTitle')<br>FROM MEDLINE.CITATIONS<br>WHERE CONTAINS (CITATION, '"Coronary" & "Disease"') = 1 |
| Searching the number of matches | SELECT PMID, NUMBEROFMATCHES(CITATION, '"disease"')<br>FROM MEDLINE.CITATIONS<br>WHERE NUMBEROFMATCHES(CITATION, '"disease"') > 0 |
| Searching and returning the score | WITH T1(PMID, SCORE) AS (SELECT PMID, SCORE(CITATION, '"Disease"&"Coronary"')<br>FROM MEDLINE.CITATIONS )<br>SELECT * FROM T1 WHERE SCORE > 0 ORDER BY SCORE DESC |
| Searching for terms in two sections | SELECT PMID<br>FROM MEDLINE.CITATIONS<br>WHERE CONTAINS (CITATION, 'SECTIONS ("abstracts, titles") "coronary disease"') = 1 |

Table 2 Query statistics for BIND database

| Interaction Data Details | Count | Homodimers |
|---|---|---|
| All protein-protein interactions | 16546 | 8052 |
| Protein-protein interactions involving two S.Cerevisiae proteins | 507 | |
| Protein-protein interactions involving two mammalian proteins | 4986 | 2335 |
| Protein-protein interactions from mammalian protein subset that involved two human proteins | 2653 | 1419 |

tions implemented by PubMed. Given that the list of aliases for "mouse" and "human" are even larger than for "yeast," it was projected that search time would increase significantly. Using MyMED to query MEDLINE citations proved to be a significant improvement.

Though originally developed to work with the MEDLINE data, this methodology has been applied to other XML data, such as BIND[16] interaction data and BioMed Central[17] article data. From the BIND data, 16 546 BIND XML records were selected and inserted into a relational table as XML CLOBs (Table 2). Next, specific interaction fields were extracted using XML Extender functions and inserted into another table where relational queries could be applied to the data. Statistics resulting from such queries can be seen in Table 2. The BIND DTD is more complex, and it was encouraging to find that we could successfully extract all interaction data from these records. The BioMed Central Bioinformatics article DTD is a simple DTD with approximately 96 elements and 33 attribute list items. At the time there were 2895 full text articles available ranging in size from 4 KB to 695 KB. We successfully created and built a database in approximately eight minutes and indexed these article abstract and body sections in an additional eight minutes.

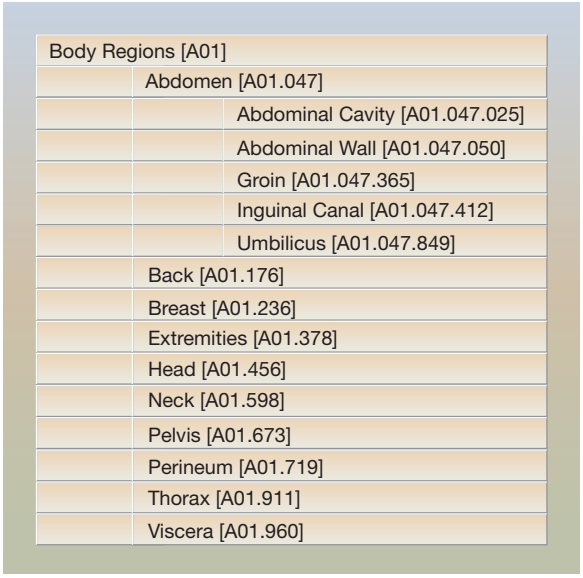
## Future Directions

Future work includes the development of an API similar to the Seqhound API[18] that can be used to access the database either locally or remotely, from other computers within the organization. This would represent a user-friendly layer that avoids the need to interact directly with the DB2 Extenders. SQL stored procedures have been built to accept as input the search argument and the element to be searched.

Searching through MeSH terms can be made more powerful. The MeSH terms are available in a hierarchical structure that allows for either general or specific levels of searching. For example, when searching for 'abdomen'—a very general term—all subunits in the abdomen branch of the tree could also be included, or everything under the branch such as 'abdominal cavity', 'abdominal wall', 'groin', 'inguinal canal', and 'umbilicus,' as shown in Figure 4. This hierarchical information can be incorporated into the table, and recursive SQL can be used for searching through the MeSH terms. This can also be implemented by using code that preprocesses the MeSH queries before creating the SQL statements.

Using XSLT, customized views of MEDLINE data can be created in different formats, such as the formats in MEDFILE. These may be used as input for various bibliography software packages. The interface for a Web submission form for MEDLINE searches and its resulting matches can be customized using XSLT stylesheets.

Figure 4 MeSH hierarchy example




An implementation that makes use of the newest version of DB2—the newly released Version 8 and the NSE rather than the TIE—is planned once the XML Extender is available to be installed with DB2 Version 8. It would be interesting to implement this tool and see how it compares to the performance of the TIE.

## Conclusions

The MyMED database project arose from the need to make complex queries on MEDLINE citations that return large result sets. Such queries are generated during text-mining investigations for biomedical research, such as the PreBIND project. During the MyMED implementation, which uses DB2, XML Extender, and Text Information Extender, we have developed an alternative method to insert and query XML documents using a low-level XML Extender column technique. The same methodology has proved successful with other XML data with varying DTD complexity.

There were many challenges that arose and, through trial and error, methods were discovered to overcome these limitations. In addition, trial fixes were made available by IBM, or work-around solutions were developed for each problem. These methods provide a stable relational database implementation of the MEDLINE data. The resulting MyMED database effectively mirrors the MEDLINE database, can be built

in a reasonable amount of time, can be queried in a reasonable amount of time, and allows text mining of MEDLINE citations with unrestricted access.

## Acknowledgments

We would like to thank Ramiz Baykara, Angel Reyda, and Juergen Metter for many useful discussions and suggestions. The Blueprint Initiative is supported by the Canadian Institutes of Health Research, the Ontario Government's Research and Development Challenge Fund, Genome Canada, and industry partners Sun Microsystems, MDS Proteomics, and Foundry Networks.



## Cited references

1. *United States National Library of Medicine*, National Institutes of Health, http://www.nlm.nih.gov/.
2. The Blueprint Initiative, http://www.blueprint.org/.
3. I. Donaldson, J. Martin, B. De Bruijn, C. Wolting, V. Lay, B. Tuekam, S. Zhang, B. Baskin, G. D. Bader, K. Michalickova, T. Pawson, and C. W. Hogue, "PreBIND and Textomy—Mining the Biomedical Literature for Protein-Protein Interactions Using a Support Vector Machine," *BMC Bioinformatics* **4,** No. 1, (March 27, 2003).
4. *XSL Transformations (XSLT)*, World Wide Web Consortium (W3C), http://www.w3.org/TR/xslt.
5. *MEDLINE Characters*, United States National Library of Medicine, http://www.nlm.nih.gov/databases/dtd/medline_characters.html.
6. *Documentation for Distribution of 2003 Baseline MEDLINE Database*, United States National Library of Medicine, http://www.nlm.nih.gov/bsd/licensee/2003_baseline_doc.html.
7. *The Perl Directory—perl.org*, The Perl Foundation, http://www.perl.org/.
8. *DB2 Net Search Extender—Product Overview—IBM Software*, IBM Corporation, http://www-3.ibm.com/software/data/db2/extenders/netsearch/index.html.
9. *Unicode Home Page*, Unicode Inc., http://www.unicode.org/.
10. *UML*, Object Management Group, http://www.uml.org/.
11. *Medical Subject Headings—Home Page*, United States National Library of Medicine, http://www.nlm.nih.gov/mesh/meshhome.html.
12. *XML Path Language (Xpath)*, World Wide Web Consortium (W3C), http://www.w3.org/TR/xpath.
13. *DB2 XML Extender—Download—IBM Software*, IBM Corporation, http://www-3.ibm.com/software/data/db2/extenders/xmlext/downloads.html.
14. *XML Extender Administration and Programming Version 7*, IBM Corporation, (1999, 2000), pp. 131–147.
15. *Text Information Extender Administration and User's Guide Version 7.2*, IBM Corporation (1995, 2001), pp. 99–105.
16. G. D. Bader and C. W. Hogue, "BIND—A Data Specification for Storing and Describing Biomolecular Interactions, Molecular Complexes and Pathways," *Bioinformatics* **16,** No. 5, 465–77 (2000).
17. *BioMed Central | About Us | Data Mining Research*, Biomed Central Ltd., http://www.biomedcentral.com/info/about/datamining/.
18. K. Michalickova, G. D. Bader, M. Dumontier, H. Lieu, D. Betel, R. Isserlin, and C. W. Hogue, "SeqHound: Biological Sequence and Structure Database as a Platform for Bioinformatics Research," *BMC Bioinformatics* **3,** No. 1, (2002).

*Accepted for publication June 30, 2004.*

**Kimberly N. Lewis** *(kimberlynina@hotmail.com).* Kimberly Lewis is currently working as an independent contractor in bioinformatics development in Singapore. Research for this paper was conducted while she was an employee of the Blueprint Initiative. She received a B.Sc. degree in computer science from the University of Victoria in 2000 and a B.Sc. degree in biology from the University of Western Ontario in 1996. She also attended the Canadian Bioinformatics Workshops where she obtained a Certificate in Bioinformatics. Before joining the Blueprint Initiative in 2002 as a bioinformatics database developer, she worked as a software developer at MDS Proteomics.

**Mark D. Robinson** *Banting and Best Institute, 112 College Street, Toronto, Ontario M5G 1L6, Canada (mrobinson@mdsp.com).* Mark Robinson is currently a Bioinformatics Associate at MDS Proteomics. He received a B.Sc. in applied mathematics and statistics from the University of Guelph in 1999 and a M.Sc. in statistics from the University of British Columbia in 2001. He worked as a statistician in the laboratory of Dr. Timothy Hughes at the Banting and Best Department of Medical Research at the University of Toronto for two years and has been working at MDS Proteomics since July 2003.

**Timothy R. Hughes** *Banting and Best Institute, 112 College Street, Toronto, Ontario M5G 1L6, Canada (t.hughes@utoronto.ca).* Dr. Hughes received his Ph.D. in cell and molecular biology from Baylor College of Medicine and did his postdoctoral work at Rosetta Inpharmatics (now a subsidiary of Merck). He is now an Assistant Professor in the Banting and Best Department of Medical Research at the University of Toronto, and holds a Canada Research Chair. Dr. Hughes has extensive experience in the manufacture and use of DNA microarrays, and is well known for innovative approaches in functional genomics. He played a key role in the development of a new microarray technology in which oligonucleotides are synthesized *in situ* by delivering phosphoramidite microdroplets with ink-jet printer heads. He also authored one of the first manuscripts formally demonstrating that microarray expression patterns can be used to distinguish hundreds of cellular states, facilitating both characterization of novel genes and identification of the activities of poorly characterized drugs. His laboratory is focused on developing and applying new DNA microarray methods and new approaches to the analysis and utilization of genome-based experimental data.

**Christopher W.V. Hogue** *Samuel Lunenfeld Research Institute, Mount Sinai Hospital, 600 University Avenue, Room 1060, Toronto, Ontario M5G 1X5, Canada (hogue@mshri.on.ca).* Dr. Hogue is a scientist at the Samuel Lunenfeld Research Institute of Mount Sinai Hospital and is an Assistant Professor in the Department of Biochemistry at the University of Toronto. Dr. Hogue has been

developing bioinformatics applications since 1986, working in both the commercial sector and in academic software development, and since that time he has published over 40 scientific articles and six patents. He is involved in the training of undergraduate and graduate students and in the University of Toronto Proteomics and Bioinformatics program. Dr. Hogue earned his B.Sc. degree from the University of Windsor and his Ph.D. from the University of Ottawa, both located in Ontario, Canada. He was a GenBank Fellow during his postdoctoral term at the U.S. National Institutes of Health in the National Center for Biotechnology Information (NCBI). At NCBI Dr. Hogue helped develop a new 3-dimensional structure database and wrote a widely used program for visualizing biological molecules called Cn3D. He has published 30 papers in the field of bioinformatics since 1987, but also has several well-cited publications including work in the fields of protein engineering, semi-enzymatic chiral synthetic methods, methods for detecting protein interactions, lanthanide luminescence, and protein structure-function studies using fluorescence spectroscopy. Hogue has received several major grants for his research, including a CDN $29 million effort to build the BIND database, making him one of the top-funded biological scientists in the world. He was selected as one of Canada's Top 40 under 40 in 2001.