

ENHSIN Evaluation Caching Solutions

25.03.2003

Report for the

Freie Universität Berlin
ZE Botanischer Garten u. Botanisches Museum
Berlin-Dahlem
Königin-Luise-Straße 6-8
14191 Berlin
Germany

by

V.I.M. - Verlag für interaktive Medien GbR
Norbert Hirneisen
Dr. Christian Köppel
Orchideenweg 12
76571 Gaggenau
Germany

Tel. +49 (0) 7225-79137
FAX +49 (0) 7225-79132
Email: postmaster@vim.de
Internet: <http://www.vim.de>

Content

1.	Task description	3
2.	Evaluation IT data provider.....	4
3.	Caching in General.....	5
3.1.	Fundamentals and Terms	5
3.2.	Conclusion	6
4.	Storage of XML data	7
4.1.	XML Repositories	7
5.	XML repository products.....	8
5.1.	OpenSource products	8
5.2.	Tamino: XML database system by the Software AG	8
5.3.	XIS, a XML database by eXcelon.....	9
5.4.	Conclusion.....	9
6.	Proposed Solution	10
7.	Resources	11
8.	List of native XML databases	12

1. Task description

- Analysis of (not fully specified) database management systems which are in use by collectors. These systems are about to be evaluated regarding their ability to process parallel requests on linked tables, views, or queries. Example views are to be defined by the BGBM. Possible database systems could be: Access 97, Access 2000, Access with MSDE, spreadsheet using ODBC, DBase (cf. BioCISE survey)
- On the basis of this evaluation, (pragmatic) methods are to be developed, which are used to resolve found performance bottlenecks (e.g. cyclic export of Queries of static tables, cyclic export of databases to powerful servers, cyclic export of tables associated to the respective views to powerful servers, further indexing and caching mechanisms)
- Development of a preferred solution and required methods for updating caches of collectors' data. The resulting specification has to support a feasible implementation for the BioCASE project.
- The recommended cache server is to be developed exemplarily for a number of existing databases (specified by the BGBM).

2. Evaluation IT data provider

The evaluation of all data provider registered in the CORM database revealed the following distribution of used database: 40% Microsoft Access, 28% dBase or dBase dialects, and approx. 10% client-server databases (70% Oracle of it).

It is very difficult to make a general evaluation in terms of performance, since this is highly dependent on available hardware and software and its configuration. Besides, the configuration is the responsibility of the data provider and can neither be determined, controlled, nor modified from the outside.

Performance and response time of a request by the CORM database depends on three substantial technical parameters of the data provider:

1. Hardware (clock rate, memory, hard disc access)
2. Internet connectivity (bandwidth, leased line/direct connection, Internet by Call)
3. Database management system (used DB software, DB application design, usage of indices and optimized queries, etc.)

Since the majority of data providers using Microsoft Access, the essential parameters are listed to achieve a low response time. According to Microsoft the maximal amount of accessing users is 255, the technical press says 25 users and according to the opinion of many Access developers the maximal amount of users is 5. A table should not exceed 100.000 data records. A MDB file should not contain more than 25 tables. Database access via ODBC should be avoided.

The overall response time of a request depends on the data provider with the slowest response time, because the CORM database waits for all inquired data providers before it creates the result. To speed up the response time, data of slow providers should be cached on a fast server with a direct Internet connection.

3. Caching in General

3.1. Fundamentals and Terms

Caching is a mechanism to speed up the access to data of a slow memory or database. Frequently used data is stored in a faster memory (*Cache memory*, short *Cache*) and is served from there. In the context of databases or Web Services, responses of frequently placed queries are stored in the cache.

A Cache consists of a limited number of storage spaces, in which individual entities reside. Each entity is addressed by a key, e.g. SQL queries are used for relational databases. If the Cache is full, the *Caching strategy* defines which entity remains stored and which are replaced by new ones. A common strategy is the “least recently used” strategy. As the name suggest, the entity which was least recently used is replaced by a new entity in the case of a Cache overflow.

If the result for a query is already stored in the Cache and is served from there, a *Cache hit* occurred. If the result for a query is not yet present in the Cache, it has to be requested from the slower memory or database. In this case a *Cache miss* occurred.

The efficiency of a Cache respectively its Caching strategy is judged by its *Hit ratio*, which is specified as the percentage of all queries served from the Cache out of the total number of queries. The higher the Hit ratio, the better the Cache works.

The *Granularity* of a Cache describes the size of the stored entities. Concerning Web Services or XML databases the following granularity levels are conceivable:

- A HTTP proxy resembles the simplest form of a Cache for Web Services. It caches Web Server responses for each HTTP request.
- For each SOAP/XPath request, SOAP messages or XML fragments returned by the server are cached.
- Complete structures associated with single requests are stored in a cache. This could be complete XML documents.
- All data will be cached. In this case however Caching is not the right term to use. It is rather called data replication or data mirroring.

The *Coherence* of a Cache describes how up-to-date its stored entities are. For example, if data is changed in a database, the associated data in the Cache have to be updated. Depending on the application requirements, different levels of coherence and therefore different updating strategies are conceivable. Especially for distributed scenarios (e.g. parallel computing) a high coherency is important.

To achieve a decent coherence, cached data can be updated cyclical. The update is initiated by the Cache in fixed time intervals by re-requesting the data from the server for each cached entity. In another method the server initiates the update by informing the Cache about entries made invalid due to a write operation. The Cache itself removes or updates these entries.

3.2. Conclusion

It is to be expected that once the system is in service, requests are rarely similar. A caching on the level of single requests would most likely induce a bad utilization of the cache (low hit ratio). It is therefore recommended to store larger data structures (whole documents or databases), which can serve a larger spectrum of requests.

A high availability is desired but cannot be assured for all data providers. It is suggested to periodically replicate all data to a powerful database server to achieve the needed availability.

4. Storage of XML data

Caching of query results available in ABCD schema can be accomplished in two different forms.

1. XML documents are stored in a native XML database (XML repositories) which store XML in its native form, either as indexed text or as a variant of the Document Object Model (DOM), which can be mapped to an underlying data storage system.
2. The result is distributed to tables of a relational database management system (RDBMS). XML tags and their attributes are stored in relational tables and views are defined which represent the structure of XML elements. Data stored in this form can easily be queried, changed, or reformatted by using SQL.

4.1. XML Repositories

XML documents can be divided into two categories: data-centric or document-centric.

Data-centric XML documents are used for data exchange. This includes for example order data, data about persons, and scientific data. The physical structure of such documents is insignificant in many cases. A special case of data-centric documents are dynamic Web pages which are generated from well-known static data, e.g. Online catalogues or address lists.

To store and manage data in data-centric documents, a database is required which is tuned for data storage, for example a relational or object-oriented database. To make this data available on the Web additional publishing capabilities are required.

Document-centric XML documents are used with their text decoration abilities, as for example in manuals, static Web pages, and (product) flyers. Such documents are characterized irregular structure with mixed content; their physical structure is of significance.

To store and manage document-centric documents, native XML databases or a Content Management System (CMS) is required. Both were developed to store content, such as newsletter articles, chapters, and glossary entries. They contain the documents metadata, such as the author name, revision information, and document identifiers.

Native XML databases are optimized for XML needs, which results in a better performance and support of particular XML requirements like a complex document structure, quick navigation on (access) paths, and full-text search.

Relational and object-relational databases on the other hand provide plenty of proven performance-optimized methods, which can also be used for XML data. Furthermore, the use of two different DBMS, one for SQL and one for XML, is avoided.

5. XML repository products

5.1. *OpenSource products*

We have tested three different OpenSource native XML databases, Xindice by the Apache Group, eXist, and the 4Suite platform.

Apache's Xindice is completely written in Java and runs on JDK 1.3 or higher. Although the installation and usage is very straightforward, this database is not suitable to be a satisfying solution. It is designed for storing collections of small sized XML documents, but not to handle large sized files. It is not possible to store files with a size of 5MB or larger. For our tests, we used a XML file (300kb) representing a flat table with six columns and 842 rows. Indices were created for columns (XML tags) relevant for queries. Response times for in XPath syntax defined queries however were in dimension of one minute.

Tests of the XML databases eXist (Java) and 4Suite Platform (Python) revealed that these products are in an early state of development. Therefore, they are not suitable for use in a practical environment either.

5.2. *Tamino: XML database system by the Software AG*

Tamino developed by the Software AG is to be considered as commercial solution of a native XML database. This product has a market share of 40%.

Tamino is the abbreviation of „Transaction Architecture for the Management of Internet Objects“. The platform consists of three product groups for

- Database management
- Integration of Applications and Data
- Application development

The database management component is the Tamino XML database. It is responsible for storage, publication, and exchange of any kind of structured and unstructured data.

Integration is ensured by the components Tamino X-Bridge and Tamino X-Node. Tamino X-Bridge implements the transformation and routing between XML documents and applications, Tamino X-Node integrates conventional data sources (e.g. SQL databases) into the XML world and consolidates all data into an integrative view. While reading and writing, Tamino X-Node converts externally stored data into XML data streams and vice versa.

Application development with Tamino is supported by Tamino X-Studio. It provides tools to create and manage XML documents as well as Java-based electronic business applications.

The core of the Tamino XML platform is the Tamino XML database which is designed to store structured information in native XML format. XML is used thoroughly as major format to structure, organize, and store information in this native XML database. The storage of XML in its original form shortens the response time while accessing this data. Tamino structures every valid XML document and creates a structure definition (DTD) even if none is present. The structure of data can be changed dynamically.

Queries are specified with X-Query, a XPath-based query mechanism. Tamino includes solutions for the Document Object Model (DOM) and the XML Query Language (XQL), it allows the processing of XML documents with XSL and CSS and partially support the Simple Object Access Protocol (SOAP).

The disadvantages of Tamino are its price of \$ 45.000 per CPU and its complexity.

5.3. *XIS, a XML database by eXcelon*

XIS (eXtensible Information Server) is a XML database system provided by the eXcelon corporation. According to own statements, XIS is the fastest XML system available for storage and supply of content information. It has a market share of 25% by now.

XIS stored XML data directly as Document Object Model (DOM) trees. XIS makes changes in structure and data of XML documents dynamically and in real time by processing only those XML elements or sub-elements needed for a particular business process or transaction. Paired with a patented internal memory system for node-level caching and locking, speed and throughput is significantly enhanced.

Tests made by eXcelon demonstrated, XIS in its third generation is capable of storing more than 10 million documents with a data amount of several Gigabytes. It is reported to support more than 40.000 users per day. These tests were conducted on a server having two Pentium CPUs (each 800 MHz).

The possibility to use XIS as supplemental data management solution for other kinds of databases or database managements systems was extended in the products' third generation. It includes complete XA-support and allows participation in distributed transaction with it. Furthermore, the new JCA-support (with JCA transaction semantics) provides a simplification of interoperability and a higher usability within J2EE application servers.

The XIS basic price of \$ 30.000 includes licenses for five developer workstations and one developer server. A 30-days evaluation license of XIS can be downloaded and used for product test free of charge.

5.4. *Conclusion*

OpenSource XML databases we have tested could not meet the requirements and therefore remain out of consideration for the use as Cache-Server. With a commercial solutions a small part of their offered functionality is used and therefore they are most likely too expensive as solution.

6. Proposed Solution

The following scenario we are proposing as an easy to implement “Caching” solution:

Data provider showing bad response times for queries should receive the possibility to provide their data on a Caching-Server (actually a Mirror-Server). The affected providers are marked within the CORM database, so CORM know to whom to make the query, directly to the data provider directly or to the Caching-Server.

All data required for queries is exported off-line by the data provider into a single XML file of the ABCD schema which will be compressed afterwards with a ZIP tool to minimize network usage during the upload. XML documents can be compressed to approx. 5% of their original file size. 1MB compressed XML data is equivalent to 20MB data.

The Caching-Server provides a Web form consisting of one or more HTML pages to allow a simple upload of the compressed XML file using the HTTP POST method. Uploaded files are decompressed on the server. The document is then validated against the requirements of the ABCD schema using a XML parser. If this test is passed, the import of the data starts.

All existing data records of this data provider are deleted from all relational tables. The affected records are determined by their ProviderID which is a field available in all tables. The XML file is analyzed by a SAX parser and the data will be distributed to single tables together with relations in the form of generated IDs which preserve logical associations. Data sets of a single provider are always exchanged completely, therefore all data is deleted first and then is newly imported. This method is much easier than inserting, manipulating, or deleting single data records. Always go by the slogan: Keep it simple.

The result of a successful or incorrect import along with statistical information is sent to the provider using e-mail. This method is recommended, because the import of large amounts of data usually takes a lot of time. It would not be expected of data providers to stay on-line the whole time to wait for the result.

After a successful import the updated data is available for queries. Using SQL-select statements, the data can be queried rather quickly. The result is always provided in XML format. If equal queries arrive in short intervals, their results are most likely still available in the internal database cache. In this case results are provided even quicker. Furthermore, requests to the Caching-Server concerning multiple data provider can be combined into a single query.

We recommend Oracle as relational database management system. An alternative would be the OpenSource database MySQL. Since version 4.1, MySQL supports SQL-Queries containing Subselects which simplify the handling of complex SQL statements.

A reasonable extension to the whole procedure would be an automatic upload. This counteracts the tendency of rather rare and erratic updates by data providers due to the manual work to export and upload their data. To implement this extension the Cache-Server periodically sends a request to the each data provider, e.g. by using HTTP. Thereupon, the database server at the data provider automatically creates the XML file if necessary. The file will then be uploaded to the Cache-Server in a way no manual involvement by the data provider is required anymore.

7. Resources

- Extensible Markup Language (XML)
<http://www.w3.org/XML/>
- XML Path Language (XPath)
<http://www.w3.org/TR/xpath>
- XQuery 1.0: An XML Query Language
<http://www.w3.org/TR/xquery/>
- Document Object Model (DOM)
<http://www.w3.org/DOM/>
- SAX (Simple API for XML)
<http://www.saxproject.org/>
- Product information for Apache Group Xindice
<http://xml.apache.org/xindice/>
- Product information for 4suite
<http://4suite.org/index.xhtml>
- Product information for eXist
<http://exist.sourceforge.net/>
- Product information for Tamino XML Database
<http://www.softwareag.com/tamino/>
- Product information for eXcelon eXtensible Information Server (XIS)
<http://www.exceloncorp.com/products/xis/>
- Daniel Nützler. XML Datenbanksysteme: Tamino, eXcelon, Oracle 8i, DB2. 2001.

8. List of native XML databases

Product	Homepage	Summary
4Suite, 4Suite Server	http://www.4suite.org/4SuiteIndex.html	Developer: FourThought License: Open Source DB Type: Object-oriented Language: Xpath, Xpointer, Xlink, XSLT, RDF (indexing) Server: Corba, SOAP, http APIs:
Birdstep RDM XML	http://www.birdstep.com/database_technology/rdm_xml.php3	Developer: Birdstep License: Commercial DB Type: Object-oriented Language: Xpath Server: APIs: SAX, DOM, Java, C++
Centor Interaction Server	http://www.centor.com/solutions/technology.shtml	Developer: Centor Software Corp. License: Commercial DB Type: Proprietary Language: Server: APIs: Java (J2EE), C++, JDBC, ODBC
Cerisent XQE	http://cerisent.com/cerisent-xqe.html	Developer: Cerisent License: Commercial DB Type: Proprietary Language: Xpath, Xquery Server: APIs:
Coherity XML Database (aka Xdisect)	http://www.coherity.com/products/xml_database.html	Developer: Coherity License: Commercial DB Type: Proprietary Language: Xpath, XML/SQL (proprietary) Server: APIs:
DBDOM	http://dbdom.sourceforge.net/	Developer: K. Ari Krupnikov License: Open Source DB Type: Relational Language: Server: APIs:
dbXML	http://www.dbxml.com/product.html	Developer: dbXML Group License: Commercial DB Type: Proprietary Language: Xpath, XSLT Server: APIs:
DOM-Safe	http://www.ellipsis.nl/	Developer: Ellipsis License: Commercial DB Type: Proprietary Language: Xpath, Xlink Server: WebDAV APIs: SAX2, DOM L2

Product	Homepage	Summary
eXist	http://exist.sourceforge.net/	Developer: Wolfgang Meier License: Open Source DB Type: Relational Language: Xpath Server: HTTP, XML-RPC APIs:
EXtc	http://www.ellipsis.nl/	Developer: M/Gateway Developments Ltd. License: Commercial DB Type: Multi-valued Language: Xpath (subset) Server: SOAP, WSDL, HTTP APIs: DOM L2
eXtensible Information Server (XIS)	http://www.exln.com/products/xis/	Developer: eXcelon Corp. License: Commercial DB Type: Object-oriented (ObjectStore). Relational and other data through Data Junction Language: Xpath, Xquery, XSLT Server: APIs: Java, VB, COM, .NET
GoXML DB	http://www.xmlglobal.com/prod/db/index.jsp	Developer: XML Global License: Commercial DB Type: Proprietary (Text-based) Language: Xpath, Xquery Server: APIs: Java
Infonyte DB, PDOM	http://www.xmlglobal.com/prod/db/index.jsp	Developer: Infonyte GmbH License: Commercial DB Type: Proprietary (Model-based) Language: Xpath, XQL, XSLT Server: APIs: DOM L2, Java (JAXP)
Ipedo XML Database	http://www.ipedo.com/html/products_xml_data.html	Developer: Ipedo License: Commercial DB Type: Proprietary Language: Xpath, Xquery, XSLT Server: SOAP, HTTP APIs: Java (J2EE), .NET
Lore	http://www-db.stanford.edu/lore/home/index.html	Developer: Stanford University License: Research DB Type: Semi-structured Language: Server: APIs:
Lucid XML Data Manager	http://www.lucid-it.com/dev/lxdm.html	Developer: Ludic'i.t. License: Commercial DB Type: Proprietary Language: Xpath Server: APIs:

Product	Homepage	Summary
MindSuite XDB	http://xdb.wiredminds.com/	Developer: Wired Minds License: Commercial DB Type: Object-oriented Language: Xpath, XSLT Server: APIs: C++, Java
Natix	http://www.dataexmachina.de/natix.html	Developer: data ex machina License: Commercial DB Type: File system(?) Language: XPath Server: APIs: C++, Java
Neocore XML Management System	http://www.neocore.com/products/products.htm	Developer: NeoCore License: Commercial DB Type: Proprietary Language: Xpath, Xquery Server: HTTP(S)/SSI APIs: Java, C++, Microsoft COM
ozone	http://ozone-db.org/frames/home/what.html	Developer: ozone-db.org License: Open Source DB Type: Object-oriented Language: Server: APIs: ODMG3.0, DOM
Sekaiju / Yggdrasill	http://ozone-db.org/frames/home/what.html	Developer: Media Fusion License: Commercial DB Type: Proprietary Language: Xbath (based on XQL) Server: HTTP APIs: COM
Socrates XML	http://ozone-db.org/frames/home/what.html	Developer: Cincom License: Commercial DB Type: Object-relational? Language: Xpath, extended SQL, RegEx Server: APIs: JDBC, ODBC
SQL/XML-IMDB	http://www.quilogic.cc/	Developer: QuiLogic License: Commercial DB Type: Proprietary (native XML and relational) Language: Xquery, SQL92 (subset) Server: APIs: Microsoft .NET, Visual C++, Visual Basic, Office, and IIS/ASP, Borland C++ and Delphi, Perl, and PHP.
Tamino	http://www.softwareag.com/tamino/	Developer: Software AG License: Commercial DB Type: Proprietary. Relational through ODBC. Language: X-Query (extended Xpath), XSL, CSS Server: SOAP, HTTP, WebDAV APIs: XML:DB, SAX, (J)DOM, ODBC, JDBC, OLE DB

Product	Homepage	Summary
Tendara Mobile XML Database		Developer: Tendara License: Commercial DB Type: Proprietary Language: Server: APIs:
TeraText DBS	http://www.teratext.com/get/page/browser/browser?category=Products/TeraText%20DBS	Developer: TeraText Solutions License: Commercial DB Type: Proprietary Language: Server: SOAP, WebDAV, LDAP APIs:
TEXTML Server	http://www.ixiasoft.com/products/textmlserver	Developer: IXIA, Inc. License: Commercial DB Type: Proprietary (Text-based) Language: Full text Server: APIs: COM+ , Java
TigerLogic XDMS	http://www.rainingdata.com/products/tl/about/tl.html	Developer: Raining Data License: Commercial DB Type: Pick Language: Xpath, XSLT Server: HTTP, SOAP APIs: Java (J2EE)
Virtuoso	http://www.openlinksw.com/virtuoso/	Developer: OpenLink Software License: Commercial DB Type: Proprietary. Relational through ODBC Language: Xpath, Xquery, (SQL) Server: SOAP, WebDAV APIs:
XDBM	http://sourceforge.net/projects/xdbm/	Developer: Matthew Parry, Paul Sokolovsky License: Open Source DB Type: Proprietary (Model-based) Language: Server: APIs:
XDB	http://zvon.org/index.php?nav_id=61	Developer: ZVON.org License: Open Source DB Type: Relational (PostgreSQL) Language: Xpath (subset) Server: APIs:
Xfinity Server		Developer: B-Bop Associates, Inc. License: Commercial DB Type: Relational. Others through "data connectors". Language: Xpath, Xquery, XSLT, TQL Server: SSL APIs: SAX, DOM L2, Java (J2EE)

Product	Homepage	Summary
X-Hive/DB	http://zvon.org/index.php?nav_id=61	Developer: X-Hive Corporation License: Commercial DB Type: Object-oriented (Objectivity/DB). Relational through JDBC Language: XQuery, Xpath, XSLT + XSL-FO, Xupdate, Full text Server: APIs: DOM L3
Xindice	http://xml.apache.org/xindice/	Developer: Apache Software Foundation License: Open Source DB Type: Proprietary (Model-based) Language: Xpath, Xupdate Server: APIs: XML:DB, Corba, XML-RPC
Xyleme Zone Server	http://xml.apache.org/xindice/	Developer: Xyleme SA License: Commercial DB Type: Proprietary Language: Xquery Server: SOAP APIs: C++, Java
XYZFind Server		Developer: XYZFind Corporation License: Commercial DB Type: Proprietary Language: XYZQL (similar to Xpath, XQL) Server: HTTP APIs: Java