Trends in Open Source Database Management Systems

Naoya Kotani†, Masakazu Okumura, Toshifumi Enomoto, and Tetsuo Sakata

Abstract

Open source software, of which the most well-known example is Linux, is being used in many commercial IT (information technology) systems for database management systems and other kinds of middleware. The great reductions in cost compared with commercial software have drawn attention to this area as the next open system wave in IT systems. This article describes two trends in this area: one for open source database management systems (DBMSs) such as PostgreSQL, MySQL, and Firebird and the other concerning the development of Xmics/PostgreSQL, which is an XML-DBMS based on PostgreSQL (XML: extensible markup language).

1. Introduction

Open source database management systems (OSS-DBMSs) have a long history as open source software (OSS) comparable to the well-known UNIX toolkits and Linux. The different systems offer different features, but they all continue to be developed in the direction of the functions and performance of commercial DBMSs.

Some DBMSs that were developed and marketed as commercial products have recently been released as OSS. These days, there is nothing preventing a DBMS from being used in a commercial IT system simply because it is open source. One cannot hide from the fact that OSS-DBMSs have the capability to replace commercial DBMSs. Indeed, their use will be a key factor in reducing the cost of IT systems.

2. OSS-DBMS trends

The well-known OSS-DBMSs that are receiving the most attention are compared in **Table 1**.

† NTT Cyber Space Laboratories
Yokosuka-shi, 239-0847 Japan
E-mail: kotani.naoya@lab.ntt.co.jp

2.1 PostgreSQL

PostgreSQL [1] is an OSS-DBMS project that grew out of work begun by Dr. M. Stonebreaker of the University of California, Berkeley and has the same roots as Ingres. It is based on POSTGRES Version 4.2 and continues to be developed by the open source community. PostgreSQL supports SQL92/99 [2] of the database language SQL (structured query language) specified by the ISO (International Organization for Standardization) and other state-of-the-art functions. In terms of functionality, it is unsurpassed, even by commercial DBMSs. It is covered by the BSD (Berkeley Software Distribution) license [3], which places no restrictions on use, modification, or distribution for any private, commercial, or educational use.

2.2 MySQL

MySQL [4] is an OSS-DBMS that was started in 1995 by Michael Widenius of the TcX Data Konsalt company in Sweden. It is currently being developed by MySQL AB, also in Sweden. MySQL development emphasizes performance, so it has fewer functions than commercial DBMSs. Nevertheless, it is regarded as offering particularly fast query processing and is widely used as a database back-end for Web servers. The licensing terms allow users to select either the GPL (GNU General Public License) [3] or a commercial (paid) license.

Table 1. Features of the main OSS-DBMSs.

	PostgreSQL	MySQL	Firebird	Ingres	Derby
Latest stable version (As of January 20, 2005)	Ver. 8.0.0	Ver. 4.1.9	Ver. 1.5.2	Ingres r3	Ver. 10.0.2.2
License terms	BSD license	GPL or commercial	IPL	CA-TOSL	Apache license, version 2.0
Main target platforms	Linux, Solaris, AIX, HP-UX, FreeBSD, etc.	Linux, Windows, Solaris, AIX, HP-UX, FreeBSD, etc.	Linux, Windows, Solaris, HP-UX, FreeBSD, MacOS X	Linux, Windows, UNIX, OpenVMS	Java-based
Features	Implements the important functions of SQL92/SQL99	Functionally inferior to commercial DBMS, but processing speed is high (query processing is particularly fast)	Functionality fully conforms to the SQL92 entry level. SQL99 is also implemented for the most part.	Functionality fully conforms to the SQL92 entry level. Has the same origins as PostgreSQL.	Small program size (about 2 MB) allows application to small-scale systems.

2.3 Firebird

Firebird [5] is an OSS-DBMS that is being developed by the OSS community on the basis of the InterBase 6.0 source code released by Borland in 2000. Functionally, Firebird conforms fully to SQL92 Entry Level 1, and most of SQL99 has also been implemented. It is a lightweight program that features easy setup and management. The licensing terms are specified by IPL (InterBase Public License) [3] and it can be used without charge even for commercial purposes.

2.4 Ingres

Ingres [6] is an OSS-DBMS that was announced for release as open source in May 2004 by Computer Associates International. The source code was released in October 2004. Tracing back through its history reveals that Ingres and PostgreSQL have the same roots. The licensing terms are specified by CA-TOSL (CA-Trusted Open Source License) [6], which allows incorporation into the products of other companies under the condition that the Ingres source code is supplied along with the product.

2.5 Derby (Cloudscape)

Derby [7] is a Java-based DBMS for which IBM released the source code to the Apache Software Foundation (ASF) in August 2004. It features a small program size of about 2 megabytes. The licensing terms are specified by Apache license version 2.0 [8] and it can be used without charge even for commercial purposes.

The source code for Ingres and Derby has only recently been released, so it is not yet clear how future development will proceed, that is to say, whether OSS development communities will form or whether a particular company will undertake development, etc. Nevertheless, a clear start has been made and these systems will probably establish a position for themselves among the OSS-DBMSs.

3. Overview of OSS-DBMS functionality

Here, we take PostgreSQL as an example of an OSS-DBMS that offers a large function set to show what functions these programs implement. The many functions of a DBMS can be categorized as 1) query processing functions for searching and updating the database, 2) transaction support functions, and 3) operation support functions.

(1) Basic search and update functions

Although the current SQL standard is SQL99, the function set has expanded greatly, so “SQL99 Core Features” are being recommended for implementation as the basic function set. PostgreSQL 7.4 implements about 90% of the core features, excluding only those functions that are seldom used. In terms of support for SQL, this system is in no way inferior to commercial DBMSs.

In addition, a cost-based optimizer for the SQL query processing mechanism is implemented. This provides a mechanism for selecting the optimum processing method based on the distribution of values within the database to achieve high efficiency. In this respect too, a practical stage of implementation has been reached.

Table 2. PostgreSQL transaction functions.

Property	Explanation	Supporting functions	Extent of implementation*
Atomicity	This means that either all of the database operations in a transaction must be completed or none of them will be completed.	SQL statements such as BEGIN, COMMIT, and ROLLBACK for specifying the beginning and end of a transaction are available.	Good
Consistency	Transaction operations on a database that is in a consistent state (i.e., having no contradictions) must leave the database in a consistent state when the transaction is completed.	SQL statements such as ASSERTION are available for detecting violations of consistency when the database is updated.	Good
Isolation	Even when multiple transactions are executed in parallel, they are seen as having been executed sequentially, so that no abnormalities result.	The MVCC function is implemented for efficient execution of transactions.	Good
Durability	Once a transaction has been completed and the result committed as correct, the result will persist in the event of any kind of failure.	The WAL function, which allows the database content to be restored to normal even after a system crash, is implemented. In the recent version 8.0, recovery from media failure is also possible.	Fair (v. 7.4) Good (v. 8.0)

* compared with commercial DBMSs

(2) Transaction support functions

One purpose of using a DBMS is to implement transactions. The essential properties to a transaction (atomicity, consistency, isolation, and durability) are implemented by PostgreSQL to the extent shown in **Table 2**.

One point to note in Table 2 is the implementation of a multiple version concurrency control (MVCC) mechanism to achieve highly efficient transaction execution while maintaining isolation. Another point is the implementation of a write ahead logging (WAL) mechanism, which protects the results of transactions in the event of a system crash or other failure. In addition, the most recent version 8.0, which was released on January 19, 2005, implements a point in time recovery (PITR) mechanism that uses backup files and log records to recover a database after a disk crash or other media failure. In this respect as well, this system is approaching the same level as commercial DBMSs.

(3) Operation support functions

As utility functions for supporting operation, OSS-DBMSs are also equipped with the basic ones for backing up the database and issuing warnings when errors occur, etc.

Commercial DBMSs, on the other hand, have recently been focusing on providing complete sets of maintenance tools in addition to these basic functions. In this respect, OSS-DBMSs have yet to catch up with commercial DBMSs.

4. Using OSS-DBMS in the development of XML-DBMS

Here, we describe an example of an R&D activity with OSS-DBMS in NTT Cyber Space Laboratories.

As a format for message exchange in transactions between companies on the Internet, standards based on XML (extensible markup language) are becoming mainstream. Anticipating an increasing demand for storage and retrieval of XML data in its native form, NTT Cyber Space Laboratories has been conducting research and development of XML-DBMSs.

There are two approaches to storing XML data with a DBMS. One is to store the XML data with its own structure, which is known as a native XML-DBMS. The other approach is to alter the structure of the XML data so that it can be stored in a relational DBMS (RDBMS), which has a proven record of reliability. Here, we describe Xmics/PostgreSQL [9], which is the application of PostgreSQL to the latter approach.

There are three general methods for storing XML data in a relational database (RDB), as described below (**Fig. 1**).

1) Store the XML data in its entirety in one column and create indexes only for the elements that serve as search keys.
2) Store it with the elements of the XML data mapped to columns of their respective tables.
3) Separate the elements into nodes and links based on the XML tree structure model and store them in respective tables.

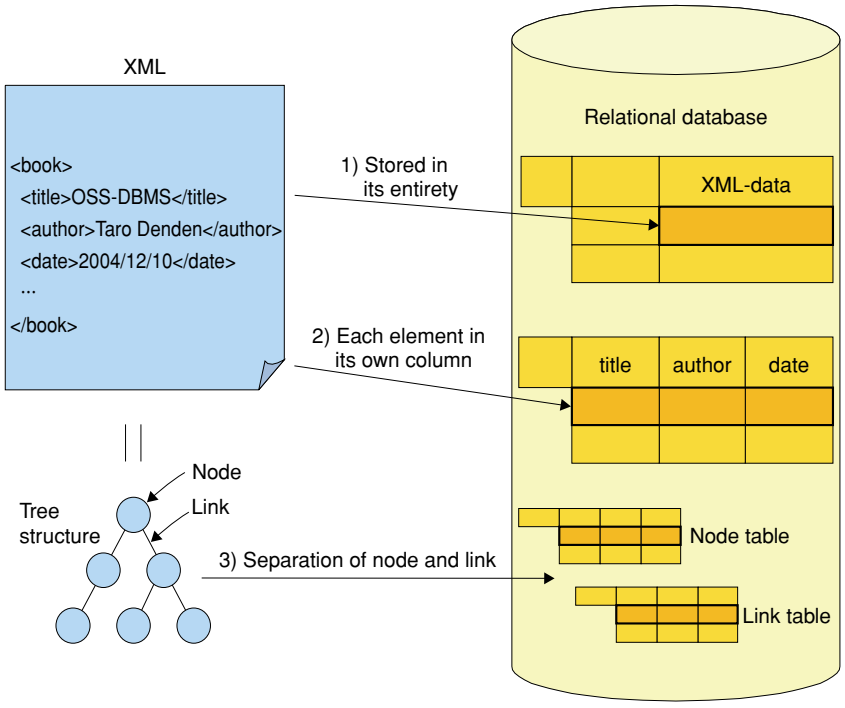

Fig. 1. How XML data is stored in a relational database.

Here, we explain the first two methods. The first is suitable for storing and retrieving all of the XML data as a whole and offers fast operation for that purpose. Another advantage of this method is that it can handle semi-structured XML data, which is known as "schemaless" or "self-describing". For partial operations such as retrieving or updating only part of the XML data, however, this method takes a lot of time. In contrast, in the second method, although it takes time to input all of the data because the XML structure must be reconstructed, subsequent partial operations can be performed at high speed. Furthermore, SQL operations can be used after the data has been stored, which has the advantage of reducing the need to add to the RDBMS.

The value of using each method depends on the application program and the XML data, so is desirable for the DBMS to support both, allowing the user to select one or the other.

Xmics/PostgreSQL has functions that support both methods and they can also be used in combination. For example, method 2 can be used to update a certain range of the data in a set of XML data, and the remaining data set can be stored as a whole by method 1.

Xmics/PostgreSQL is implemented as a middleware component between the application program and PostgreSQL and another component that extends the functions of PostgreSQL (**Fig. 2**). This provides the application program with access to what it sees as an XML-DBMS rather than an RDB. Furthermore, a schema conversion tool that automatically creates an RDB schema from the XML schema is also provided. Although the basic technology of the Xmics/PostgreSQL system can be applied to a general-purpose RDBMS, the fact that the PostgreSQL source code is open allows optimization of the processing within PostgreSQL, making it possible to implement a fast and reliable XML-DBMS that is also highly cost effective.

5. Future development

NTT Cyber Space Laboratories will continue to improve the performance, reliability, utility, and operability of OSS-DBMSs and increase the added value of their functions in order to gain expertise in making use of them and extending their range of application.

References

[1] http://www.postgresql.org/
[2] "Information Technology—Database Languages—SQL," ISO/IEC 9075-1999.

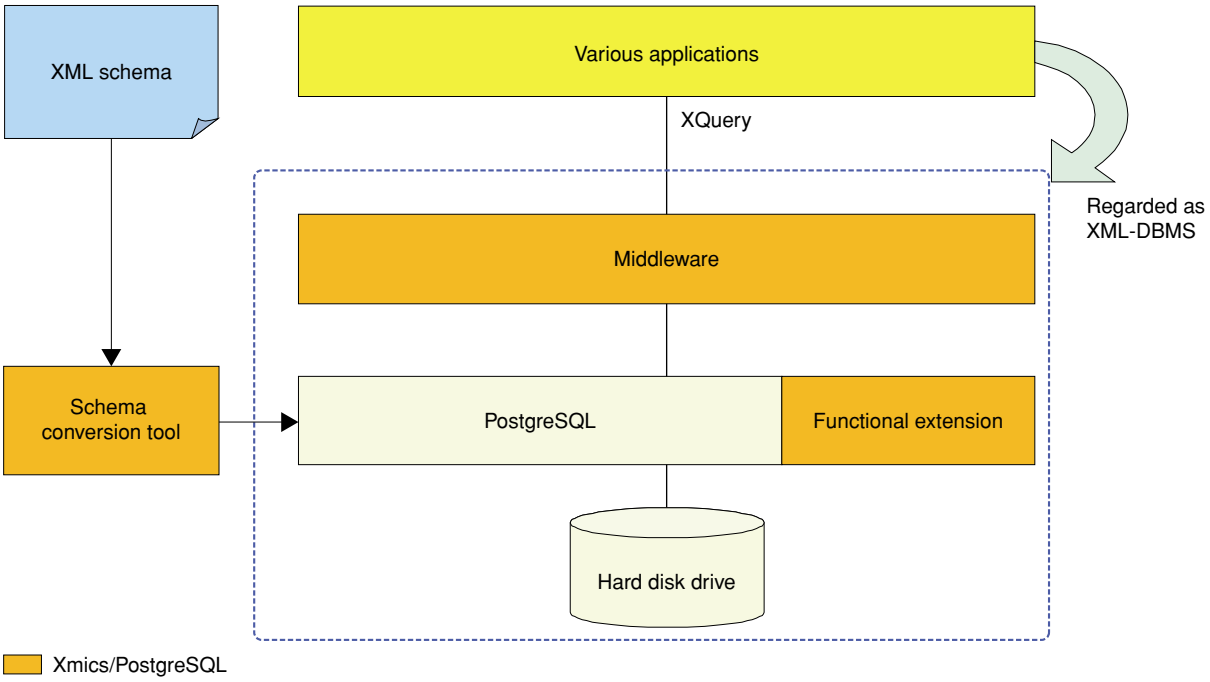

Fig. 2. Xmics/PostgreSQL configuration.

[3] http://www.gnu.org/licenses/license-list.html
[4] http://www.mysql.com/
[5] http://firebird.sourceforge.net/
[6] http://opensource.ca.com/projects/ingres/
[7] http://incubator.apache.org/derby/
[8] http://www.apache.org/licenses/
[9] S. Kasuga, T. Sakata, N. Kotani, and T. Honishi, "An Implementation of XQuery Processing System using Relational Database," IPSJ SIG Technical Reports, 2004-DBS-134 (II), pp. 293-299, 2004 (in Japanese).

Naoya Kotani
Senior Research Engineer, NTT Cyber Space Laboratories.
He received the B.E. and M.E. degrees in electrical engineering from Science University of Tokyo (now Tokyo University of Science), Tokyo in 1986 and 1988, respectively. He joined NTT Communications and Information Processing Laboratories in 1988. Since then, he has been engaged in R&D of multimedia application systems with database management systems. He is a member of the Information Processing Society of Japan (IPSJ).

Toshifumi Enomoto
Research Engineer, NTT Cyber Space Laboratories.
He received the B.E. and M.E. degrees in systems engineering from Osaka University, Suita, Osaka in 1992 and 1994, respectively. In 1994, he joined NTT Communications and Information Processing Laboratories. Since then, he has been engaged in R&D of information retrieval systems. He is a member of IPSJ and the Japanese Society for Artificial Intelligence.

Masakazu Okumura
Research Engineer, NTT Cyber Space Laboratories.
He received the B.E. and M.E. degrees in electrical and computer engineering from Kanazawa University, Kanazawa, Ishikawa in 1992 and 1994, respectively. In 1994, he joined NTT Information and Communication Systems Laboratories. Since then, he has been engaged in R&D of database application systems and user terminals for IPv6 networks. He is a member of IPSJ.

Tetsuo Sakata
Senior Research Engineer, NTT Cyber Space Laboratories.
He received the B.E. and M.E. degrees in applied system science from Kyoto University, Kyoto in 1988 and 1990, respectively. He joined NTT Communications and Information Processing Laboratories in 1990. Since then, he has been engaged in R&D in the data engineering field ranging from database management system and database design methodology to database applications for multimedia retrieval. He is a member of IEEE Computer Society and IPSJ.