

## The Hidden Costs of Using Open Source Databases for Java Application Development

## Executive Summary

In today's turbulent economy, many software vendors, systems integrators and corporate IT groups are trying to develop more cost-effective mission-critical Java applications. Thanks to the growing recognition of Linux open source solutions, some companies are now also seriously considering the use of open source databases for their applications.

However, these firms do not realize the many hidden costs of utilizing open source database technologies for Java development. These costs include:

- The "contagious" nature of open source database software licensing, which can force associated proprietary technologies into the not-for-profit open source arena
- Security and privacy risks of open source databases
- Increased productivity costs of database development and support
- Reduced productivity due to maintenance and upgrades requirements
- A lack of features and concurrency, especially for Java open source databases

While open source databases may have their place, it is not within mission-critical enterprise applications. Products such as PointBase Embedded and PointBase Micro provide more reliable, cost-effective functionality to developers while avoiding the many hidden costs of open source databases.

## Introduction

Over the past few years, open source has developed some momentum in business environments, largely due to a number of support solutions offered around the Linux operating system. These open source technologies are based on a free license that gives users access to the source code, allowing them to run programs and systems for any purpose, to modify the program and to redistribute copies without having to pay royalties to the application vendor.

In today's tough economic climate, a no-cost, relational, open source database can sound like a developer's dream come true. It is an appealing concept: a completely free database that can be modified in-house to meet the specific needs of the firm. As a result, some enterprise Java developers, system integrators and independent Java software vendors are beginning to consider open source databases to support mission-critical products and applications.

However, these companies may not be aware of the many hidden costs, security issues and legal issues of using open source databases. Technological immaturity, poor technical support and documentation, lack of features and lack of adherence to standards are among the many problems these databases present to Java application developers and enterprise IT groups. Ill-defined licensing laws are also creating major problems within this arena. For example, independent software vendors (ISVs) are finding that they can lose the intellectual property rights to their applications because they are based on an open source database. The laws are so hazy that some companies, such as MySQL™ AB, have even been able to set up licensing programs that allow them to resell work contributed for free by the open source community.

For some situations, such as small web publishing companies or ISPs offering simple web hosting, open source databases may be the best option. However, for any enterprise engaged in creating, selling or operating mission-critical applications, open source databases should be approached with extreme caution. This paper reviews the current state of the open source business model and discusses the hidden costs of open source databases when used in conjunction with Java application development. It also reviews more cost-effective, reliable options available to Java developers, including PointBase databases.

### **Today's Open Source Database**

Open source databases have not yet reached the level of sophistication of other open source technologies and developers using them must be aware of certain inherent weaknesses when assessing their usability. First, the nature of open source databases is that they cannot exceed a certain level of complexity. To do so would require an organized, concentrated effort that cannot be carried out via the uncoordinated work of disparate open source technology contributors, with each rewriting source code to achieve different personal goals. An open source technology must by definition accommodate the lowest common denominator of its contributors. In fact, many open source contributors are not highly qualified; almost none are experienced database experts who have worked for years on advanced relational databases. Typically, open source contributors are recent college graduates who work on open source database technologies in their spare time. Unlike commercial database developers, they have no requirements or incentives to respond to the specific needs of paying customers.

Second, the pace of new development for open source databases is much slower than for commercial projects—sometimes twice as long. For this reason, open source technologies in general are still much less advanced than their commercial counterparts. Many industrial-strength features of commercial databases that have become expected and required over the past two decades are now just being "discovered" by open source database developers. As just one example of this, MySQL announced in early 2002 that it now supports database transactions that have been a standard feature of relational databases for more than 25 years.

Open source Java technologies are even further behind, with no technical support in existence. Based on the newest development language, it will be some years before open source Java technologies are able to catch up to even the C-based open source database products that have been in existence for a significant number of years.

### **The Hidden Costs of Open Source Databases in Java Development**

It is easy to see why companies may be attracted to the idea of a no-cost database, without realizing the hidden costs of utilizing such a technology on a regular basis. It is indeed tempting to save the company's money by using an apparently free system – at least, until something goes wrong. To assure that IT specialists are making informed decisions, it is vital for those considering such a move to be aware of the potential problem areas in open source database technologies.

## 1. The Open Source Licensing Boondoggle

One of the most significant problems of using open source databases lies in licensing – so much so that if a company does choose an open source database, it is extremely advisable to have the corporate legal team double-check all the licensing ramifications beforehand. When an enterprise selects an open source database, it may do so by utilizing a GNU Public License (GPL) – a free development agreement – or it can buy a commercial license. Under the GPL, which is the most common form, companies agree to redistribute any modifications to the database back to the open source community. Under a commercial license, which includes a support agreement, proprietary source code is protected.

However, many companies do not realize that the GPL license is also "contagious" - that is, when a program is linked to and distributed with a GPL program, all the source code for all aspects of the resulting product must also be released under the GPL. In other words, an ISV utilizing an open source database must make its product an open source application as well, to be freely adapted, modified and utilized by users in any way they wish. Legally, the ISV no longer owns the intellectual property rights for the application. They've just contributed their application source code to the open source community.

As outrageous as this sounds, some open source database companies have been seeking to aggressively enforce this interpretation of the GPL. For example, MySQL AB brought suit against NuSphere™/Progress in 2002 and actually succeeded in forcing NuSphere to make its Gemini transactional storage engine an open source application, since it was developed based on MySQL technologies.

As a result of all this, companies such as Microsoft® and Oracle® have become concerned enough about the "viral" nature of the GPL to put restrictions on some toolkits that can be distributed prohibiting them from being paired with any open source software that may impair the firms' rights. For example, Oracle's license states:

*"Open Source" software-software available without charge for use, modification and distribution-is often licensed under terms that require the user to make the user's modifications to the Open Source software or any software that the user 'combines' with the Open Source software freely available in source code form. If you use Open Source software in conjunction with the programs, you must ensure that your use does not: (i) create, or purport to create, obligations of us with respect to the Oracle programs; or (ii) grant, or purport to grant, to any third party any rights to or immunities under our intellectual property or proprietary rights in the Oracle programs. For example, you may not develop a software program using an Oracle program and an Open Source program where such use results in a program file(s) that contains code from both the Oracle program and the Open Source program (including without limitation libraries) if the Open Source program is licensed under a license that requires any "modifications" be made freely available. You also may not combine the Oracle program with programs licensed under the GNU General Public License ("GPL") in any manner that could cause, or could be interpreted or asserted to cause, the Oracle program or any modifications thereto to become subject to the terms of the GPL.*

Such restrictions have a major impact on systems integrators and software vendors that may wish to pair their open source database with libraries associated with commercial application development tools. As with all such situations, it is not always clear where the GPL's influence begins or ends and many companies become aware of this problem only too late.

For minor applications, revealing the source code may be immaterial. However, for larger mission-critical applications, it is a major concern. Utilizing open source database code that is also freely available to other companies, hackers and competitors increases the company's security and competitive risk. The safest answer is therefore to buy a commercial license for the open source database that protects the developer's proprietary information. However, there is little difference between buying a commercial license for an open source database versus buying a commercial database that can be embedded directly into the application – except that the embedded commercial database will create less work and probably be cheaper as well.

MySQL AB has developed a "Trojan Horse" business model that takes advantage of these licensing laws to lure companies into relying on their open source database. Drawn in by the promise of a "free" database, developers find themselves forced to buy commercial licenses from MySQL as their dependence on the technology grows. MySQL AB's commercial licenses come with various levels of support: Premium support packages, which include the services necessary for developers, cost nearly \$50,000.

Interestingly enough, MySQL AB has also set up a licensing situation that takes even greater advantage of current laws, while demonstrating a serious flaw in the open source database model. When using a MySQL database, all the modifications a developer makes under the GPL are released to the open source community but are actually legally owned, not by the open source users, but by MySQL AB. These modifications in turn are often integrated, without any payment to the developer, into the company's commercial database option and licensed for a fee to its customers. Apparently, most open source database developers are not yet aware that their work is being co-opted in this fashion.

## **2. Costs of Development and Support**

Organizations considering the inclusion of an open source database within their application must also consider the cost of modifications to the database code. The time needed to read and understand source code should not be underestimated. Modifying an application database requires a software developer with expertise in database internals and SQL standards. They must also understand the large transactional systems on which databases are built in order to support enterprise-class features and thousands of users. Most application developers do not have these skills – this is the reason the database industry developed in the first place. As a result, the additional task of database modification in the development process lengthens the production cycle, increases administrative overhead and introduces additional risks and personnel costs.

Unless the enterprise or software vendor purchases a complete support contract, they must trust the skills of their in-house developers and DBAs for the reliability of their application. These experts must be able to function without one of the key elements of strong technical support - good documentation. Open source database documentation is notorious for being poorly written, inadequately policed and highly inaccurate. Lacking the support of a unified database development organization, developers must be exceptionally experienced to ensure that all bugs are fixed, security breaches are protected and the database is completely robust.

Jonathan Schwartz, executive vice president of Sun Microsystems, summarizes the thoughts on open source from CIOs who are responsible for enterprise applications, "Most CIOs I talk to want less source code, not more."

### **3. Costs of Maintenance and Upgrades**

Developers must remember that even a simple open source database that runs today with no modifications to its code may not work at all after their first application upgrade. Open source databases provide no steady upgrade path because there is no commercial imperative to provide upward compatibility. Instead, developer companies must spend time and money to consistently monitor and track all upgrades to the open source database. The database will then have to be completely upgraded and modified to work with the next application release, while at the same time providing backward compatibility.

At best, this serves as a major distraction to application developers who must become researchers and technologists rather than focusing on producing profitable application releases in a timely fashion. As stated in a recent Enterprise Developer News case study, "Many solutions providers find the constantly evolving process of libraries, patches and updates unmanageable." Much time can be spent trying to update databases from live open source repositories. Even more can be spent if the new revisions seem to work, but actually don't. Coming from a variety of developers, working at various levels of expertise, it is inevitable that not every piece of open source code will work with a specific application, nor will patches necessarily work together properly.

### **4. Costs of Security**

Levels of security vary between open source databases: Some provide encryption and some do not. Being able to see the source code allows developers to more easily identify potential security breaches, but it obviously also provides the same advantage to anyone wishing to break into an application that is developed on the database. As previously pointed out, open source code is open to everyone, including competitors, and it can essentially provide a roadmap for hackers. This fact has already caused major security breaches. For example, in December 2002, the IDG News Service reported security holes in the MySQL open source database that enabled denial-of-service attacks both in the server and the client components of the software.

While these security breaches have been patched, fixes after the fact are still too late for any enterprise. In general, few open source databases can meet the stringent encryption and security standards of most organizations, including large business enterprises and government agencies. Compared to commercial products, open source databases rarely provide complete J2EE™ security for database and network, and they also introduce new security risks by running as a separate process.

## 5. Costs of Lack of Features and Connectivity

Open source databases offer little in the way of features, making them largely a technology for the in-depth programmer. They do not provide tools to help manage and control data, nor such capabilities as subqueries, triggers, Java stored procedures, views, multiple connections or scalability. They are also less likely to be compliant with SQL 92, SQL 99, J2EE and other standards that enterprise developers take for granted.

Lack of concurrency support is just as important: Few open source databases support complete concurrency such as transaction isolation, row-level locking, two-phase commits and connection pooling. This makes applications less functional, lessening the value of the software to the end-user while lengthening the development cycle and creating more work for the programmer. The few Java-based open source databases available suffer from a great many technical difficulties and are not mature technologies.

Overall, as stated by Intelligent Enterprise magazine (Sept. 18, 2001): "Open source databases must improve dramatically in reliability, scalability, and... features before they are fit for enterprise use." Companies developing mission-critical applications and software vendors selling commercial products should search for other options to maximize their database capabilities.

### The PointBase Embedded Database

Java databases such as PointBase Embedded protect companies from the many hidden costs of open source databases. This robust, full-featured RDBMS product is integrated within the application itself and is extremely cost-efficient in terms of licensing fees, development and deployment. For developers, nothing makes more sense than using an embedded Java database within a Java application. Offering a low overhead and rapid, reliable performance, PointBase Embedded has already proven itself with leaders of the Java community such as BEA®, Sun® Microsystems and Macromedia®, among others. Many of these companies examined and discarded the idea of using open source databases before turning to PointBase.

PointBase, a division of DataMirror, is the only company in the industry whose primary focus is on Java relational databases. PointBase Embedded runs anywhere with a Java Virtual Machine (JVM), giving vendors the flexibility to develop and test on demand with the same JDBC syntax and data types. This makes it completely transparent to the end-user and enables one-step application deployment from a single JAR file, without risking conflict with existing databases.

PointBase Embedded offers complete connectivity, supporting J2EE, SQL and JDBC. No DBA is required to support developers: The database is administration-free, allowing organizations to save time and resources. PointBase Embedded maintains a small footprint (approximately 1MB) while storing up to a half-terabyte of data, enabling applications to scale as needed. It also provides complete Java-based security.

Unlike any open source database, PointBase offers sister products, PointBase Micro and PointBase UniSync, which allow PointBase Embedded to support applications across and outside of the enterprise. These products have been designed and developed to work with each other as well as enterprise databases, providing a complete database and synchronization solution. This synergy enables expanded business strategies for mobile and remote operations, deployed across multiple mobile platforms including notebooks, PDAs and beyond.

Last but certainly not least, PointBase is consistently commended by developers for its responsive and knowledgeable support team. Rather than reading through masses of open source documentation or calling for help on the open source usergroups, users have instant access to a comprehensive, commercial and online knowledge base. Utilizing PointBase's flexible, friendly help procedures assures that any development problems are resolved quickly, allowing applications developers to make continual progress toward the completion of their project.

### **Conclusion**

Despite a certain level of interest in the open source database concept over the past few years, it is apparent that these technologies are not ready to compete against commercial products. While open source databases have their place in some environments, in most cases a proven, cost-efficient database product is best for Java application development. PointBase Embedded provides an elegant, powerful solution to the needs of Java developers, enabling new enterprise-class software applications, middleware and tools that are both cost-effective and completely reliable.

---

**Corporate Headquarters**  
**3910 Freedom Circle, Suite 104**  
**Santa Clara, CA 95054**

**Main. 408.961.1100**  
**1.877.238.8798 (US and Canada)**  
**[information@pointbase.com](mailto:information@pointbase.com)**  
**[www.pointbase.com](http://www.pointbase.com)**