AC 2009-2375: A SURVEY OF EMBEDDED DATABASE TECHNOLOGY FOR MOBILE APPLICATIONS

Kyle Lutes, Purdue University

Kyle Lutes is an Associate Professor for the Department of Computer & Information Technology (CIT) at Purdue University. Kyle joined the department in 1998 and is the chair of the department's software development curriculum. His teaching and scholarly interests cover a broad range of software development areas including software applications for mobile devices, data-centered application development, and software entrepreneurialism. He has authored/co-authored numerous papers and two college textbooks on various software development-related topics. Prior to his current appointment at Purdue, Kyle worked for 16 years as a software engineer and developed systems for such industries as banking, telecommunications, publishing, healthcare, athletic recruiting, retail, and pharmaceutical sales.

John Springer, Purdue University

John Springer, Ph.D. is an assistant professor in the Department of Computer and Information Technology at Purdue University where he specializes in Data Management and is the chair of the Data Management curricular area subcommittee. Dr. Springer's expertise and research interests lie in database implementation and information integration.

Kelly Howard, Purdue University

Kelly Howard is a web developer for the Armed Forces Institute of Pathology in Washington, D.C. She has a Master's degree in Technology from Purdue University, and her research interests are in data privacy and software security.

A Survey of Embedded Database Technology for Mobile Applications

Abstract

A recent project required us to develop a software application that runs on mobile devices of various form factors. Almost all non-trivial software requires data persistence of some sort, even those running on small mobile devices. For very small amounts of data, a simple sequential file is often adequate. However, for larger and more complex data sets, software developers of mobile applications frequently employ an Embedded Database Management System (EDBMS) to aid in data storage needs. While we have extensive experience in both using database management systems in software development projects, and developing applications for mobile devices, we did not feel we had enough information about EDBMS alternatives. A quick internet search shows there are a variety of commercial and open source embedded database products available on the market today. Adding to our confusion was the fact there does not seem to be any clearly defined set of standard features for embedded database products. The purpose of this study was then to educate ourselves on the various options available. In this paper, we discuss some of the options that must be considered when choosing a data persistence strategy and present the findings of our research of 17 embedded database management system products from 11 different vendors. In the paper, we list 20 desirable characteristics common in embedded database management system products, and summarize their availability in each of the products we reviewed.

Introduction

The term embedded can be defined as being "inserted as an integral part of a surrounding whole,"[1]. A quick Google search of the word "embedded" yields over 76 million results, with Information Technology (IT) centered applications of the term spanning computing, databases, processors, programming, systems, and more. According to Massa and Barr[2], an embedded system is a blending of hardware and software for a dedicated function.

One of the more interesting aspects of embedded systems is the embedded database. As embedded systems have advanced in complexity, reliable and efficient data management has become increasingly important. At one end of the spectrum, this includes network routers and gateways as well as storage devices and entertainment systems[3]. On the other end of the spectrum are devices such as PDAs, digital cameras, and mobile phones, which need data storage and synchronization but also require a small footprint[3.] All of these systems take advantage of the embedded database to eliminate the overhead of an SQL query to a remote database server for every transaction[3].

The embedded database is most commonly defined as "a software component that is part of the application," with its operations invoked by the application[4]. For the purposes of this study, a few general assumptions can be followed regarding embedded databases[3]:

- It runs as part of the application or on the host system (such as with a PDA); no separate or remote database server is required.

- End user query tools, graphical user interfaces, and reporting tools are not usually available.
- Generally, the programmer must write code to make direct calls to the database APIs in order to store or retrieve records.

Some examples of tasks that can be performed using embedded databases include sending an email or a short message service (SMS) from a Blackberry, storing favorite website URLs on a mobile phone, or shopping for ring tones. If implemented properly, virtually all of the data transactions are managed via program code, and the end user for the software application or device should not be aware of the existence of a database at all.

Coined in the 1980s, the term embedded database originally referred to a database management system (DBMS) embedded into an application, in contrast to large, centrally managed, enterprise level databases such as Oracle or IBM's DB2[5]. Today it can also refer to modern client/server database design, although generally on a much smaller scale than that of DBMSs such as Oracle or DB2[5]. There are several flavors of embedded database systems or models including in-memory, disk-based, XML, hierarchical, embedded SQL, and navigational APIs. As the demand for "smarter" devices increases, the market (and diversity of products) will continue to grow.

Survey of Products and Features

There are a variety of commercial and open source embedded database products available on the market today. Some vendors, such as Birdstep, have been working on embedded database products for nearly 20 years, whereas other products, such as db4o (database for objects), have made their debut within the last five years.

For our study, we initially researched the features set for 17 products from 11 different vendors. A list of products and a summary of supported features is shown in Appendix A at the conclusion of the document. While there are some common features that can be identified among the products, there do not seem to be a standard set of features for embedded database products.

There is also some inconsistency among vendors in terminology usage, phrasing and definition of the features they offer. Surprisingly, few vendors discuss the type of database model used or more specific technical aspects of implementation. Not surprising is the frequent mention of "Zero Administration," "Small Footprint," and a heavy focus on synchronization and replication capabilities. The focus appears to be more on successful implementation examples and who is using the product. Overall, usage of these products seems to be geared toward one of two purposes:

1) Mobile or small hand-held devices
2) Other mission critical embedded systems, typically systems requiring very high reliability

Among the more important characteristics to consider when choosing an embedded database product are:

- Zero administration
- A GUI interface or other means of configuration (if not integrated, then as an add-on)
- Support of in-memory mode
- A small footprint (typically < 1 MB)
- Data storage capacity
- Locking, cursors
- Platform independence
- Support for multiple languages
- Support for SQL standards, such as SQL 92
- Support for more complex SQL, such as views, triggers, joins, etc.
- Threading and concurrency
- ACID
- Transactional
- Implementation or access methods
- XML support
- Encryption
- Synchronization and replication capabilities
- Price and Licensing

Some of the more abstract characteristics may include ease of use, stability, quality of support resources, and performance.

As a part of a related ongoing project, we are conducting performance testing using an application we developed to compare execution times for persistent storage options, and we expound below on additional research that we did concerning specific features of the two Embedded Database Management System (EDBMS) products in use on the project: Microsoft SQL Server 2005 Compact Edition and Vista DB. We chose these particular products since the test application runs on a Windows Mobile-based device, we developed the application using the Microsoft .NET Compact Framework and the C# programming language, and these two EDBMS products have a high-level of integration with these development tools.

Microsoft SQL Server 2005 Compact Edition (SQL CE) is an extension of the SQL Server 2005 Mobile Edition technology for all Windows platforms including tablet PCs, Pocket PCs, smart phones and desktop computers. SQL CE may be deployed and redistributed without cost. It allows development using components of Transact-SQL (T-SQL) and ADO.NET as well as synchronization capabilities. SQL Server 2005 Compact Edition does offer ACID properties and handles databases up to four GB in size.

Vista DB is an embedded database engine marketed as an alternative to Microsoft's mobile database solutions. It has a C# architecture built for .NET and is managed code. Note that .NET applications are JIT compiled and accordingly tend to run slightly slower than counterparts written in C and/or Assembly. In addition, Vista DB is compatible with Microsoft SQL Server 2005's data types and has integration with Visual Studio 2005 and 2008 using its Server Explorer. T-SQL procedures are reported to be supported in a future version.

When comparing memory footprints when deployed, we found significant differences between SQL CE and Vista DB. To understand how much additional memory footprint was added by the EDBMSs, we also compared file sizes to what would be required if traditional sequential file I/O were used. For our research, we created databases with a single table then populated the table

with four different numbers of records. For sequential file I/O, we used data from a simple comma-separated values (CSV) file. The results are shown in Table 1.

	DBMS Engine Size	Database file size for *n* records			
		n = 100	*n* = 1000	*n* = 5000	*n* = 7146
SQL CE	< 1 MB	92 KB	320 KB	1.43 MB	2.12 MB
Vista DB	~ 712 KB	397 KB	3.88 MB	10.4 MB	15.1 MB
Sequential File I/O	N/A	8.17 KB	103 KB	549 KB	782 KB

Table 1: Memory footprints

We should also call attention to the extremely large sizes for the Vista DB databases. We are uncertain as to why they are so much larger than the alternatives. Since Vista DB can also run on Windows desktop computers, we repeated the tests on a standard PC and the Vista DB file sizes were closer to what we expected. Further testing with Vista DB would be required to determine if this is bug or perhaps an anomaly with our dataset.

Pricing and Licensing Options

Pricing and licensing options can be quite variable and complex for embedded database products but can be extremely important considerations for choosing a product. It is imperative to understand the terms of a licensing agreement, as this is a legally binding contract for how the product may be used and/or redistributed and what terms may apply to such actions. Two of the more common forms of licensing agreements are closed source (often somewhat erroneously referred to as commercial) and open source, although there can be huge variations with these two umbrella terms. Pricing is the cost associated with the licensing agreement, if any. Generally pricing is per license, but a royalty schema may be employed when per license charges become cost prohibitive, which allows for wide or unlimited distribution of the product using the embedded database for a percentage of profits or price of each product sold**.**

There are no hard and fast rules about what licensing model to use, and some vendors may even choose to license their products under combined licensing programs, leaving some parts of its software proprietary and some truly open source. Other companies may provide open source licensing, but require users to pay for support agreements. Creativity with licensing can provide greater opportunity for vendors and customers alike, but such creativity can also make it very challenging for a company to know exactly what it is agreeing to when selecting a product. Licensing and pricing for Microsoft SQL Server 2005 Compact Edition essentially depends on how a product is developed and used. According to Microsoft.com, "SQL Server Compact Edition is free to download, develop, and deploy applications. SQL Server Compact Edition is also free for third parties to redistribute, as long as a redistribution agreement (http://www.microsoft.com/sql/editions/compact/redistribute.mspx) is completed."

Other considerations include:

- In the development environment, the developer would need to purchase a copy of Visual Studio 2005 or later, if they should choose to develop using an integrated development environment.

- For the client use only, the user needs to have Windows XP, CE or Mobile (depending on device platform).
- If connecting to a server, the independent software vendor would need to purchase SQL Server 2005 Standard or Enterprise, plus a Client Access License, or CAL, for each connection. If there are more than 30 connections, a processor license is recommended instead. Refer to Table 2 for a summary of pricing options for this product.

Product	Price
Visual Studio 2005 Standard or Professional Edition	$299-$799
Books Online	Free
ActiveSync	Free
SQL Server 2005 Compact Edition	Free
SQL Server 2005 Standard – Single License	$885 per server
Client Access License (CAL)	$162 ea
SQL Server 2005 Standard – Processor License (> 30 Connections)	$5,737

Table 2: Pricing for Microsoft SQL Server 2005 Compact Edition

Vista DB offers a variety of plans for purchasing its software. The license is per developer, which means the software can be installed on any number of computers but used only by the licensee. There are no deployment or redistribution fees once a license is purchased. Pricing options include add-ons for subscription plans, upgrades, and incident support. Source code is available for an additional cost and with a special licensing agreement. The terms of this agreement essentially state that source code cannot be released in any manner, or used to create a competing product, SDK or API for resale, and source code must be kept secure within the licensee organization. Refer to Table 3 for a summary of pricing options for the Vista DB product.

Product	Price
Vista DB 3.2 (1 Developer)	$169
Vista DB 3.x (1 Developer / 1 year Subscription)	$259
Vista DB 3.x Subscription Upgrade (1 Developer / 1 year) (from Vista DB 3)	$119
Vista DB 3.x (5 Developer / 1 yr subscription)	$777
Vista DB 3.x Subscription Upgrade (5 Dev / 1 year) (from Vista DB 3.0 / 5 Dev)	$387
Vista DB 3.x Source Edition with 1 year subscription	$999
Vista DB 3.x Source Edition w/1 yr Upgrade (from Vista DB 3)	$499
Vista DB 3.x Site License (Source / Single Location / 1 yr subscription)	$2,999
Vista DB 3.x Upgrade (1 Developer / 1 yr subscription) (from Vista DB 2)	$119

Table 3: Pricing for Vista DB 3.x

Performance

Performance, while less concrete, is still an important consideration when selecting an embedded database product for mobile applications. Mobile device applications are typically used by people for just seconds at a time compared to minutes or hours at a time for desktop applications. Users of mobile applications expect immediate responses from their devices[6]. The speed at which a database product can execute queries can make or break an application.

In the aforementioned related project, we are conducting performance testing on a test application to compare execution times for two persistent storage options: sequential file I/O and

Microsoft SQL Server 2005 Compact Edition. A preliminary set of results that utilize these two options plus a third, Vista DB 3.2, can be found in the study "A Comparison of Embedded Database Management Systems for Windows Mobile Devices,"[7].

Conclusions

The current trend of growth in technology shows us that as mobile technology continues to develop, mobile applications will become more advanced. Companies wishing to take advantage of this platform will have to identify what products and development strategies will best suit their needs. As can be seen from the tables shown in Appendix A, a wide variety of EDBMS products with varying sets of features are available.

For applications relying on small datasets of less than 100 records, it may be perfectly suitable for an application to utilize sequential file I/O. This is also an acceptable solution where a small footprint, nominal costs and minimal effort toward deployment are required. If data security were a critical factor, however, it would be better to rely on an embedded database product, as this can handle encryption with very little impact on performance. Another advantage to this choice is the ease of implementation; encryption is usually an on or off setting with an embedded database product, whereas with file I/O it can require several lines of code. Additionally, many database products offer password protection.

When using larger datasets (more than 100 records) an embedded database product will most likely offer the best performance, particularly when utilizing search functions where data fields can be indexed. The disadvantages to this choice can be ease of use, the amount of effort involved in distribution or deployment, a larger footprint, and of course, licensing requirements and costs. Because our testing has shown markedly different execution times and memory footprints between the two EDBMS solutions we tested in detail, we encourage developers to perform their own tests.

As with any software development endeavor it is important to conduct a thorough analysis of the business requirements and the resources available. In the end, companies should invest the time required to conduct a thorough comparative examination of feature sets, pricing and licensing schemas, and performance analysis data (if available) in order to identify the product that will fit best with their business needs.

Bibliography

1. Random House. (2006). Random house unabridged dictionary. Retrieved August 23, 2007 from http://dictionary.reference.com/browse/embedded
2. Massa, A. & Barr, M. (2006, October 1). Programming embedded systems. North Sebastopol, CA: O'Reilly Media.
3. Chaudhri, A. Rashid, and R.Zicari, XML Data Management: Native XML and XML-Enabled Database Systems (Boston, MA, 2003).
4. Koopman, J. (2005, April 12). Embedded database primer. Retrieved August 16, 2007 from http://www.dbazine.com/ofinterest/oi-articles/koopmann5

5. Graves, S. (2006). Embedded databases: Data management for real-time and embedded systems. Embedded Technology Magazine. Retrieved August 29, 2007 from http://www.embeddedtechmag.com/content/view/50/122/
6. Lutes, K. (2004). InformIT, Retrieved on September 20, 2007 from http://www.informit.com/articles/article.aspx?p=170200&seqNum=1
7. Lutes, K., Springer, J., & Howard, K. (2008). A Comparison of Embedded Database Management Systems for Windows Mobile Devices. Paper presented at the 5th International Conference on Cybernetics and Information Technologies, Systems and Applications (CITSA 2008).

Appendix A: Features Matrices

Features Matrix Part 1

	Zero Admin	GUI/ Configurable	In Memory Mode	Disk Based	Footprint	Engine	Disk	Memory	Capacity	Locking
Apache Derby	X	External GUI Add-ons		X		2 MB				
Berkeley DB	X		X	X		400 KB Min			256 TB/Table	Page based
Berkeley DB Java	X		X	X		820 KB			Gigabytes	Record level
Berkeley DB XML	X			X						
Birdstep Embedded	X			X					4 GB Max	X
Birdstep Server	X	X		X					18 quintillion records/table	X
Birdstep Mobile	X		X							Dynamic
db4o	X	X	X	X		600 KB		1 MB	254 GB/DB File	Paging
IBM Pointbase Embedded	X			X		1 MB			Terabytes	X
IBM Pointbase Micro	X		X	X		45-90 KB				Table
MSSQL 2005 Compact	X	X		X		5 MB	< 2 MB		4 GB Max	Row level
MySQL Embedded	X	X	X	X			4 MB Min	1 MB Min		Row level
Perst (MC Object)	X		X					30-300 KB		X
SQLite	X	Command line		X		< 250 KB			Terabytes	X
SyBase Ultralite	X	X		X		300 KB			2 GB	Row level
TimesTen (Oracle)	near zero	API, Command line	X							Row level
VistaDB		X	X	X	< 1 MB				4 TB	DB, Table, Row Level

Features Matrix Part 2

	Cursors	Platforms (OS) Supported	Multiple Languages	Java	.NET	C/C++	Other	Model	SQL Standard/ API
Apache Derby	X	Mac OS X, Win, Suse, Unix, Sun Solaris, IBM, more…		X				Relational	
Berkeley DB	X	BREW, S60, Unix, VxWorks, Windows, Windows CE	X	X	?	X	X		
Berkeley DB Java	X	Supports all versions of Unix, Linux and Windows as well as most other major operating systems		X					
Berkeley DB XML	X	Supports all major OS's							
Birdstep Embedded	X	Unix, Linux, Windows varieties, others on request	X					Network, Relational	X
Birdstep Server	X	Unix, Linux, Windows varieties, others on request	X					Network, Relational	
Birdstep Mobile	X	Supports the common wireless OS's including Windows CE, Palm OS; others on request	X					Hierarchical	
db4o	X	Blackberry, Palm OS, Windows Mobile/Pocket PC, Mono, Harmony		X	X				Native
IBM Pointbase Embedded	X	Completely portable across all platforms		X					
IBM Pointbase Micro	X	Compatible with any device with a JVM; all major server platforms including UNIX, Linux, and Microsoft Windows		X					
MSSQL 2005 Compact	X	Windows: Tablet PC, Pocket PC, Windows CE			X			Relational	X
MySQL Embedded	X	Depends on support level: Linux varieties, Sun, Windows, Mac OS X, IBM, Novell, etc.	X	X	X	X	X	Relational	
Perst (MC Object)	X	Platform Independent for hosts of Java, C#; includes Windows CE and Pocket PC		X	X				X
SQLite	X	Windows CE, Linux varieties							
SyBase Ultralite	X	Palm OS, Windows CE, Symbian EPOC32, MS-DOS, WindRiverVxWorks real-time		X	X			Relational	
TimesTen (Oracle)		IBM, HP, Linux, Sun, Windows		X	?	X		Relational	X
VistaDB		Linux, Mac OS X, Sun Solaris, BSD, Windows, Pocket PC 2003, Windows Mobile			X			Relational	

Features Matrix Part 3

	SQL-92	SQL-99, 2003	Enhanced (Joins, BLOB, SP)	Threading/ Concurrency	ACID Transactions	Transactional	Access Methods	XML	Encryption
Apache Derby		X		X			Class library, JDBC		
Berkeley DB			X	X	X	X	API, Native, Linked Libraries		
Berkeley DB Java			X	X	X	X	API, Native, JAR file (drop and go)		
Berkeley DB XML				X	X	X	Native, API, Xquery	X	
Birdstep Embedded				X	X	X	API, Drivers	X	
Birdstep Server			X	X	X	X	API, Drivers		X
Birdstep Mobile	X				X	X	API, Drivers		
db4o				X	X	X	Native, Class library ("one-line-of-code" drop and go)	X	X
IBM Pointbase Embedded	X	X	X	X		X	JDBC, JAR file (drop and go)		X
IBM Pointbase Micro	X		X	X		X	JDBC		X
MSSQL 2005 Compact			X	X	X	X	ADO.NET		X
MySQL Embedded	X	X	X		X	X	ODBC, JDBC, .NET, C++ Drivers, Connectors		X
Perst (MC Object)				X	X	X	API	X	
SQLite	X		X	X	X	X	Drop and Go, Library		
SyBase Ultralite	X		X			X	API		X
TimesTen (Oracle)			X	X	X	X	API, JDBC, ODBC		
VistaDB	X	X	X	X	X	X	API, DDA Engine, ODBC, etc.		blowfish

Features Matrix Part 4

	Synchronization	Replication	Open Source	Pricing	Comments
Apache Derby			X	Free	
Berkeley DB	X	X	X	Free, Commercial	Optimized for Random Access
Berkeley DB Java	X		X	Free, Commercial	Optimized for Sequential Access
Berkeley DB XML	X	X	X	Free, Commercial	
Birdstep Embedded		X		Commercial	
Birdstep Server		X		Commercial	
Birdstep Mobile	X	X		Commercial	
db4o			X	Free, Commercial	Reporting
IBM Pointbase Embedded	X			Commercial	
IBM Pointbase Micro	Wireless			Commercial	
MSSQL 2005 Compact	X	X		Free	Heavy focus on Synchronization capabilities
MySQL Embedded	X	X	X	Commercial for OEM	
Perst (MC Object)			X	Free, Commercial	
SQLite			X	Free	SQL statements compiled into virtual machine code
SyBase Ultralite	X	X		Free (dev), Commercial	
TimesTen (Oracle)		X			
VistaDB				Commercial	