



Life beyond the sleeping cat for Berkeley DB?

David Mitchell

October 2006



Life beyond the sleeping cat for Berkeley DB?

When Oracle acquired Sleepycat in February of this year there was much gnashing of teeth in parts of the open source community, fearing that Oracle was buying the company to stifle competition from open source upstarts – on the back the earlier acquisition of another open source database, Innobase, in October 2005. At the time, Ovum argued that these acquisitions were driven by the burgeoning opportunity in the embedded database market, rather than any negative desire to throttle competition, despite the protestations of some open source traditionalists. Oracle has recently released version 4.5 as an update to Berkeley DB, the first major release since the acquisition. Ovum's David Mitchell analyses the release to see whether the product shows signs of atrophy or whether there has been continued development.

Key messages

Announcement of the death of Berkeley DB is premature

Oracle has continued the development of Berkeley DB, maintaining a development path that targets the embedded and realtime environments. There is no sign of a lack of commitment to the future of the product, and we anticipate functional and technical updates to continue in the future – drawing on other database expertise in Oracle and through cross-group collaboration.

The three areas where there has been most significant development have been upgrades, replication and concurrency control. These areas represent facilities that are important in a realtime environment, where high performance, automatic recovery from failure, and high levels of fault tolerance are required.

Embedded database model should be more widely adopted

Although considered by some to be an inferior model to the client-server relational database management system (RDBMS) model, the embedded database is a mature and robust technology with application in many mission-critical environments. As the pressure for increased performance over large data volumes increases, there will be concomitant pressure to



implement embedded databases, either as a replacement or an architectural complement to RDBMS solutions. A key design goal will be to reduce hardware capacity needs, while at the same time improving performance. Expect the embedded database to be a key battleground in the next three to five years of the database market, and a facet of that interest will be on architectures, rather than blended embedded and relational models – using embedded models as cache structures for their relational cousins.

Licensing is a dual model

Rather than removing the open source licensing of Berkeley DB, Oracle has continued to offer it, signalling that those who doubted the longevity of the open source nature of Berkeley DB were wrong.

There are still two licensing models offered by Oracle: commercial and open source. The open source licence allows Berkeley DB to be freely embedded in other open source projects, while the commercial licence allows the embedder to protect the confidentiality of their code and to have contractual warranties and indemnities provided by Oracle. The open source licence is OSI-certified with the restriction that, if the resulting project or product is distributed, either for free or for a fee, it must also allow its source code to be freely re-distributable. This is GPL-like in its effect rather than being a pure GPL model.

Database portfolio growth – rationalisation and explanation needed

A glance at the database section of Oracle.com shows the following databases as available: Enterprise Edition, Standard Edition, Standard Edition One, Express Edition, TimesTen In-Memory DB, Berkeley DB and Lite Edition. In addition there are a variety of database options that include Real Application Clusters (RAC), partitioning, Content DB, Records DB, Spatial and Advanced Security. This is a large database product portfolio, with each product having both general-purpose and specialist use. However, it is not always clear to the developer or to the IT operations manager which product is most suitable for which environment.

Oracle needs to develop and communicate a set of simple guidelines for developers, IT managers, and for those involved in the purchase of its database technology – advising on which product in the various families is best suited to different technical and commercial environments. In particular, the relative merits of TimesTen and Berkeley DB need to be clearly explained, and a set of heuristics developed to help the developer select their weapon of choice.



Analysis

Embedded databases, such as Berkeley DB, have a wide applicability – in environments where the often ubiquitous relational database would not be suitable for either performance or management reasons. These include use in devices like handsets, switching equipment and other telecoms equipment.

The most recent release of Berkeley DB is version 4.5 and represents the first major release of the product since the parent company, Sleepycat, was acquired in February 2006. This release adds functionality in a number of areas that improve both manageability and performance of the product – two of the key functional requirements in the embedded database category.

Before the Oracle acquisition, Berkeley DB was a leading product in the category, and the recent enhancements help to cement that position. The embedded database market is growing rapidly, faster than other database markets – at around 35% per year. The backing of the Oracle sales and distribution channel should put Berkeley DB in a good position to exploit that growth.

In addition, the database teams within Oracle are working to integrate some of the R&D work from Berkeley into other products, and to move towards a blended database model where each type of database – relational or embedded – is used in the appropriate way, in an integrated data management architecture. These moves are still early, but are likely to be increasingly important in the next three years of development by the Berkeley DB and wider Oracle database team.

Database portfolio – in need of clearer explanation

The Oracle database portfolio includes a number of products, some developed organically while others have been added to the portfolio through acquisitions. In the Enterprise database category there is Oracle Database 10g, a family of products that run from Express Edition and Standard Edition One through Standard Edition and onto Enterprise Edition. Here each member of the family is positioned as targeting progressively larger systems and more functionally rich database applications, with additional functionality being offered through database options packs – where, for example, clustering or partitioning is required.

Even with this positioning there can be uncertainty in the mind of developers and IT managers about which database product to select. In the past, things were simpler in the small footprint category where there was a single product on offer – Oracle Lite. Oracle Lite is a database that is supported on a number of environments, including Windows Mobile, Linux and Symbian. However, Oracle's merger and acquisition (M&A) campaign



has added further products into this category, expanding the category to offer more support for embedded and realtime database markets.

The notable recent acquisitions have been:

- **Sleepycat** – acquired in February 2006, giving Oracle access to one of the most popular open source embedded databases – Berkeley DB. In addition, there were two other databases produced by Sleepycat – one an embeddable, pure Java database and the other an embeddable XML database. Oracle continues development, support and dual licensing of those as well
- **TimesTen** – acquired in June 2005. This is an in-memory database product that is used in a number of realtime environments, particularly in the telecoms sector. One area where TimesTen has been used is as a high-speed persistent cache for Oracle Database
- **Innobase** – the small open source database company was acquired in October 2005. This acquisition caused consternation in part of the open source community, as InnoDB is one of the transaction engines for the MySQL product.

Although the rationale for each individual acquisition was well explained, in hindsight, the portfolio now appears to need more clear explanation. For example, when would a developer use TimesTen or Berkeley DB, or when should they use Oracle Database 10g – with a 'silent install' and licensed on a restricted basis, such as an application-specific, full user (ASFU) or other appropriate embedded-style licence? Also, when should TimesTen and/or Berkeley DB be used in combination with other Oracle databases, and what configuration and deployment advice is available?

Licensing and redistribution

One aspect of the open source licensing is the focus on redistribution. This is where a product that uses the Berkeley DB open source licence, and is then redistributed, must make its own source code 'freely redistributable under reasonable conditions'.

This presents a developer with a potentially difficult situation – determining whether this licence would still allow them to use open source licences like GPL or BSD for their own products. A challenge in answering this question is to understand the potential constraints and implications of the term. Here there are some interesting inclusions and exclusions.

Exclusions

Applications that are used internally and that are deployed on company servers are specifically excluded. However, there is potential ambiguity in the situations where the internal-use applications are provided as a managed service by a third-party service provider or hardware owned by



that service provider. In the more traditional licensing scenarios around commercial products, these scenarios also cause issues and ambiguity. It is genuinely very difficult to legislate for the range of different asset ownership and asset operation scenarios that are possible, and the Berkeley DB open source licence is not immune to those generic issues.

Inclusions

Pre-release software of all kinds is included, meaning that open source projects need to make their source code available during the beta and alpha stages – something that is entirely normal in any case. Therefore, in practice, the licence does not alter the day-to-day development practices of an open source developer.

Scripting languages are an area of potential consternation for some developers. The scenario is the one where a developer has a script in either PHP, Perl, Python or other scripting language that uses the Berkeley DB libraries. The question then is, 'do I need to release the source code of my scripts?' At first sight this is a complex issue. However, a simpler analysis is that it's the PHP, Perl or Python 'engines' which are open source and are calling or using the Berkeley DB libraries, rather than the script written by the end developer. The script interpreter engines themselves are open source, so they satisfy the requirements of Berkeley DB's open source licence. The script itself is 'data' interpreted by the engine. Based on this analysis there is no argument – the developer does not need to release the source code for their scripts, as long as the source code for the engines are open, and the scripts are not compiled.

Why are embedded databases gaining market interest?

There are two main reasons why there is increasing interest in embedded databases – whether they are commercial or open source. Indeed, it is a real mistake to equate embedded databases with open source, as is often done, since the things that drive embedded database adoption are distinct from those that are driving open source.

The technical reasons

Often, the application is required to run completely unattended, so it is not possible for the database to require human administration. The application may be deployed in a location that is inaccessible, or the cost of manual administration may be prohibitive. In these situations, an embeddable database is an ideal choice. Another reason for choosing an embedded database is that applications typically depend on specific versions of underlying infrastructure technology, such as databases, so a developer needs to be sure that these components are available as the application is installed. This has two consequences. First, that the install scripts or install facilities need to be able to check that a range of underlying components



are installed – making install scripts potentially onerous to write and maintain. Secondly, when applications are maintained or upgraded, the dependencies on underlying components can also change, making configuration testing a major challenge. Having the database facility built into the application helps to ensure that this issue is reduced.

The commercial reasons

When a customer buys or uses a business application, such as Siebel, PeopleSoft, Oracle or SAP, that application will use some element of an underlying database infrastructure – and that usage will need to be appropriately licensed. This sounds simple. However, there is potentially a great deal of complexity and ambiguity. Licensing of non-embedded databases for an embedded application is usually very complex, with only a few individuals able to fully describe the appropriate licensing mechanisms. The licensing models around an embedded database such as Berkeley DB are simpler and more readily understood.

As applications become progressively more SOA-enabled, the range of licensing complexities are likely to increase exponentially, and traditional licence agreements may explode in length to cope with the range of different technical circumstances that are possible and that need to be legislated for. Unless the industry is careful, the licences of the future will be a ghastly pastiche, reminiscent of the way that some accounting rules in some jurisdictions have exploded simple rules into enormous tomes that document every minutia.

In contrast, the embedded licence allows the software vendor to focus on creating licences that suit the business application, rather than expending energy on the infrastructure layers. This is not to say that licensing of business applications is simple – it is not. It is simple to say that having an embedded model for the underlying infrastructure allows the licensing manager to concentrate on the area that should have most relevance to a business customer.

Limitations of RDBMS models and the needs of an embedded and realtime database market

The main scenario where embedded databases excel is where:

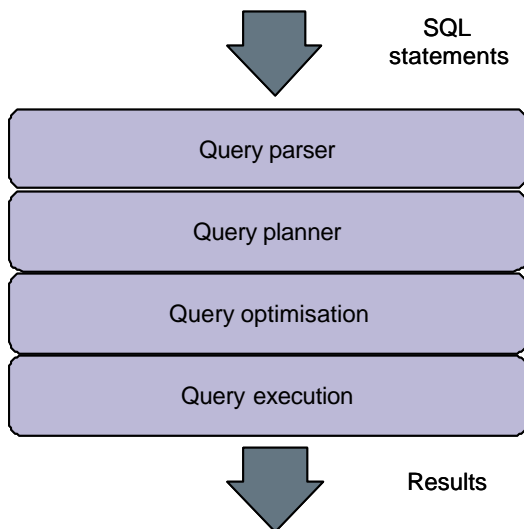
- there is a relatively fixed set of queries used by the application; for example, there is a static query domain
- high performance is needed across large volumes of data, even where the data itself can be changing a lot; for example, a dynamic data domain with large volumes of data
- the normal architectural elements of databases are still needed, such as full ACID assurance.



The RDBMS model has come to dominate the thinking of the database industry, with the two being almost synonymous. It cannot be denied that the RDBMS is the dominant model for structured data, with simple file systems still being used for the majority of less structured data.

However, there are constraints imposed by the SQL-based client-server model RDBMS. The most important of the issues is performance, and this in turn is driven by the layered architecture of most SQL-based RDBMS products. The query path of an SQL-based RDBMS product is shown in *Figure 1*.

Figure 1 **The query path of an SQL-based RDBMS product**



Source: Ovum

This path is needed because of the completely dynamic nature of SQL queries. For example, a simple SQL query could be:

```

SELECT amount
FROM customer;
  
```

while a more complex query could involve multiple tables, with various types of sub-queries and joins, such as:

```

SELECT dept.name FROM dept
WHERE NOT EXISTS (SELECT dept_id
                  FROM emp
                  WHERE emp.dept_id = dept.id)
  
```

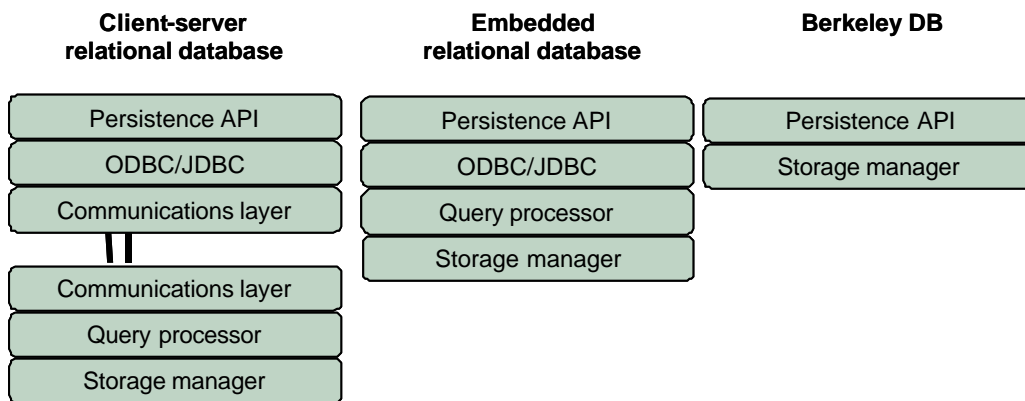
Due to the great difference between the two queries, the SQL engine needs to be layered, to parse the query and then work out the best way to



execute it and return the results using the minimum amount of processor and I/O resources. This is a complex task.

The other issue inherent in the relational model is that it is typically implemented in a distributed or client-server model, with code being executed across multiple processes and across multiple computers. While there are enormous examples of architectural flexibility, scalability and robustness, the issue is that large amounts of inter-process communication can take place – something which can generate performance issues. *Figure 2* compares the query paths of the client-server, embedded relational and Berkeley DB models, showing where additional performance issues are likely to be caused.

Figure 2 **Query paths of the client-server, embedded and Berkeley DB**



Source: Ovum

The model used by Berkeley DB and other embedded style databases is that database code runs in the same process as the application, being linked to the programmers own code and invoked through simple function calls. The other important characteristics of Berkeley DB's embedded model are:

- local persistence rather than client-server
- databases can be in-memory or on-disk
- flexible configuration, so that database sub-systems can be avoided when they are not required
- no manual database administration required

The net impact is that the Berkeley DB, and the more general embedded database model, is aimed at the developer community who produce applications that need high performance from a database, with



predictability and stable query patterns. In particular applications that are being embedded into device where database maintenance is impractical or impossible suit the embedded model that Berkeley DB and others provide.

Berkeley DB 4.5 – data management features

The use of a set of in-process embedded database libraries cannot be at the expense of the defining elements of a database, such as ACID compliance. At the basest level, there is no point in improving the speed of query access if there is a risk that the database will lose your data, or corrupt it. In Release 4.5 of Berkeley DB there are a number of areas where functionality has been expanded to address some of the more complex data management challenges.

Replication

Database replication is widely used, either as part of a business continuity arrangement, or as part of a data distribution framework that aims to spread a query workload. The commercial RDBMS market recognises replication facilities as a 'must-have' requirement, but the embedded database community has traditionally been slower to recognise the need. With Berkeley DB 4.5 an improved and supported set of replication frameworks were provided, to allow complex replication functionality to build into applications – making it easier to produce highly fault-tolerant systems.

Upgrade

Software upgrades are a big challenge for systems that cannot tolerate any downtime. Usually upgrades and patches are performed by taking a database offline, performing the upgrade and then restoring the database to production. Berkeley DB introduced a feature that allows a replicated database to be upgraded in-situ, without taking down an entire production asset. Continued improvement in this area will only increase the suitability of Berkeley DB in the 99.999% requirement space.

Concurrency control

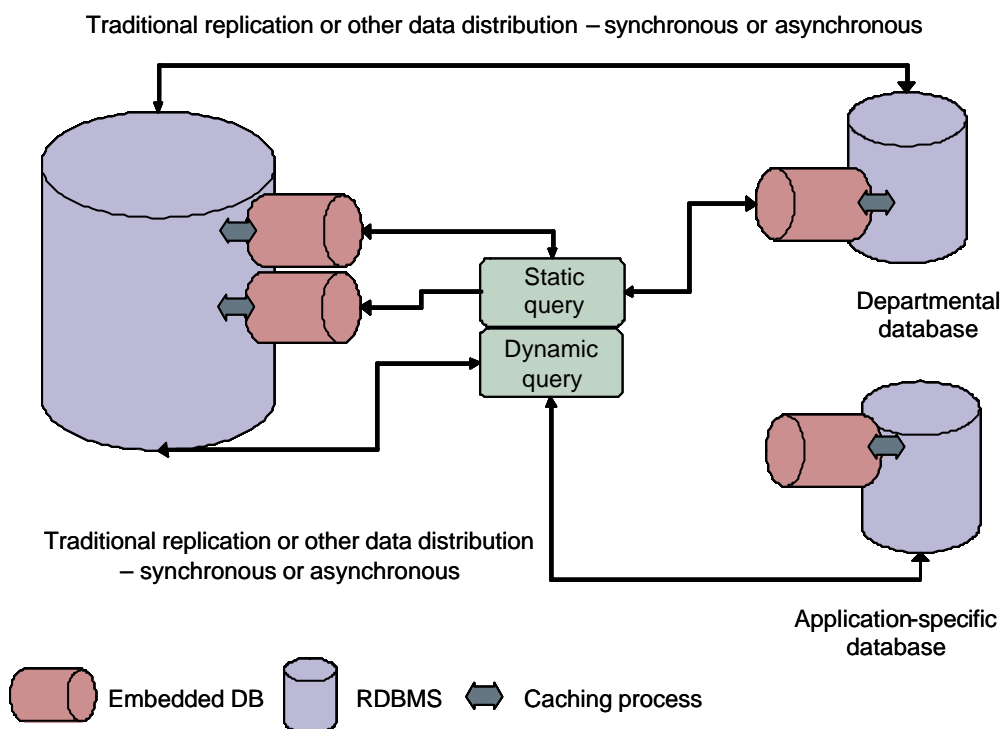
One traditional disadvantage of the embedded model can be a difficulty in delivering performance when there are mixed query and update loads on the database, and where there is high user concurrency. The latest release of Berkeley DB allows users to have their own snapshots of a database, so that readers are not blocked by writers, and automatically ensures that changes happening in different snapshots are properly managed.



The emergence of a blended database environment

It is a mistake to polarise the debate too much, and to position relational databases and embedded databases as being in distinct camps – with no middle ground. Several potential future developments will make the distinction even more difficult to sustain, as companies find that they have database architectures that deploy both in a blended model – using each for its strengths, and resulting in a blended architecture like *Figure 3*.

Figure 3 **A blended database architecture**



Source: Ovum

In-memory meets RDBMS

The in-memory technology that is inherent in many embedded databases will find its way to the RDBMS platform – most likely in the form of an in-memory database acting as a highly performant in-memory cache for a RDBMS platform. This will allow the performance of RDBMS platforms to be posted.



RDBMS to embedded integration

Embedded databases may be simple key/value data storage engines while RDBMS systems have more complex data structuring. It is likely that improved integration between the two models will develop, so that one or more embedded databases can be created from an RDBMS, and then automatically kept synchronised with the parent RDBMS.

On-the-fly embedding and query optimisation

Many of the issues of RDBMS versus embedded still arise at application design time rather than at the execution phase. One area where progress would be useful is for the RDBMS engine to predict where an application would be better served by an embedded database (based on query patterns), to create this automatically and keep it synchronised with the master RDBMS. The embedded database libraries would, of course, need to provide reciprocal facilities.

Edge and inner circle together

Embedded databases are most often positioned at the edge of the corporate data infrastructure, in remote devices and network equipment components. One other potential development enhancement is for embedded databases at the periphery to communicate, in a lightweight manner, with embedded databases that are acting as a local cache to a corporate database – well within the corporate data infrastructure. This could allow greater access to corporate data resources from remote devices and components without thrashing the corporate data infrastructure.



Client re-use disclaimer

- This is a verbatim reproduction of independent material that has previously been published by Ovum within the last 6 months
- Ovum operates under an Independence Charter. For full details please see www.ovum.com/about/charter.asp
- Neither Ovum nor the analysts were paid by the client to write any part of the material
- Ovum may have been paid by the client for the right to re-use the material
- Ovum may have a deal with the client to supply research or consultancy. However, no other relationship exists between the 2 companies (e.g. shareholdings, loans, non-executive directorships etc)
- Ovum does not endorse companies or their products
- While we take every care to ensure the accuracy of the information contained in this material, the facts estimates and opinions stated are based on information and sources which, while we believe them to be reliable, are not guaranteed. In particular, it should not be relied upon as the sole source of reference in relation to the subject matter. No liability can be accepted by Ovum Limited, its directors or employees for any loss occasioned to any person or entity acting or failing to act as a result of anything contained in or omitted from the content of this material, or our conclusions as stated
- This material is the copyright of Ovum Europe Ltd.