



# Choosing the Right Database

## *The Case for OpenBase SQL*

*What are you looking for in a database? Reliability? Performance? Ease-of-use? A proven track record? An affordable price? Low cost-of-ownership?*

*This paper provides guidance on what features and "gotchas" to look for when choosing a database and database vendor.*

*It also describes the advantages offered by the OpenBase SQL relational database and how it compares to other database offerings.*

**OpenBase International, Ltd.**

16 Centre Street

Concord, NH 03301 USA

e-mail: [info@openbase.com](mailto:info@openbase.com)

tel: (603) 228-3339

fax:(603) 228-3637

### **The ACID Test**

A.C.I.D. stands for Atomicity, Consistency, Isolation and Durability – four standards which every database should meet, but few actually do.

While A.C.I.D. compliance is not the only consideration in choosing a database, it's a good place to start in comparing your database choices.

Here is a quick definition of each term:

- ◆ **Atomicity** – All database modifications must follow an "all or nothing" rule in which each transaction is "atomic." That means that if one part of the transaction fails, the entire transaction fails. No splitting of atoms allowed! It is critical that the database management system maintain the atomic nature of transactions in spite of any DBMS, operating system or hardware failure.
- ◆ **Consistency** – Only valid data is written to the database. If, for some reason, a transaction is executed that violates the database's consistency rules, the entire transaction will be rolled back and the database will be restored to a state consistent with its rules. Transactions that successfully execute always take the database from one state that is consistent with the rules to another state that is also consistent with the rules.
- ◆ **Isolation** – Multiple transactions occurring at the same time will not impact

each other's execution. For example, if Joe issues a transaction against a database at the same time that Mary issues a transaction, both transactions will operate on the database in an isolated manner. That is, the database either performs Joe's entire transaction before executing Mary's, or vice-versa. This prevents either transaction from reading intermediate data produced as a side-effect of part of the other's transaction that will, in the end, not actually be committed to the database.

- ◆ **Durability** – Transactions committed to the database are never lost. Durability is ensured through the use of database transaction logs that facilitate the restoration of committed transactions in spite of any subsequent software or hardware failures.

## Databases with A.C.I.D. Compliance

Complete A.C.I.D. compliance is actually relatively rare among the database offerings on the market.

Sybase, Oracle and OpenBase SQL have solid strategies for dealing with all four of these areas.

PostgreSQL also offers fairly good A.C.I.D. compliance, although their documentation warns that their write-ahead logs are not 100% reliable.

## Fault Tolerance and Durability

Choosing a product with a built-in and automated capability for avoiding data-loss is critical, both in preventing the costs of data loss and in lowering the costs of running the database.

Most databases lack durability primarily because they have no effective strategy for dealing with random access files, in which corruption can sometimes be unavoidable.

In fact, file corruption happens more frequently than most operating system vendors would care to admit. The sophisticated caching in today's modern operating systems and hardware enhances performance, but it also compounds the challenge of designing reliable databases.

The most common cause of file corruption is an unexpected shutdown or a system freeze-up. Under these circumstances, database systems are prone to failure because they write to disk more frequently than most other applications. If a write is interrupted by a power outage or system crash, it can result in a corrupt write or even a truncated file. A partial write or a truncated file can be devastating to your company data.

Hard disk RAID's alone cannot solve the problem. While they do guard against



a complete hard drive failure, RAID's often just duplicate corrupted data — and give database owners a false sense of security.

Good database durability starts with the assumption that random access files will become corrupt and that when they do the database software needs to be able to take action to guarantee that files can be accurately rebuilt.

OpenBase SQL does this automatically. Sybase and Oracle require a database administrator to monitor the database. Other databases, including MySQL and SQLite, are missing this level of fault-tolerance entirely.

## **OpenBase SQL Journaling — Keeping data safe**

OpenBase SQL employs a multi-file journaling system that delivers reliability through a foolproof mechanism for addressing the common forms of file corruption.

Here's how it works: OpenBase SQL simultaneously maintains both master and working copies of database data, along with a realtime journal which tracks all of the changes. Changes are flushed to the journal as transactions commit, followed soon thereafter by batched flushing of changes to "working" random access files.

Since an incomplete write can corrupt the middle of random access files, OpenBase SQL uses the journal as a safeguard to ensure that, if file corruption occurs, the random access files can be completely rebuilt from scratch.

In the case where a database needs to be rebuilt, OpenBase SQL combines the master copy with the transaction journal to bring your database up to its present state. In this way, OpenBase SQL offers a redundant and automatic system that keeps data safe.

In addition to maintaining data integrity, this journal file approach also eliminates the need to perform many random access writes at commit time. Instead, changes to the work files can be safely batched, resulting in significantly faster database performance.

OpenBase SQL databases perform journaling tasks transparently and automatically.

### ***How do other databases compare?***

While there are a variety of approaches to the problem of avoiding data corruption, many are flawed or require intervention from a database administrator. Both can be costly.

Sybase and Oracle use a journaling mechanism similar to OpenBase to ensure data integrity. However, they also require a database administrator (DBA) to periodically empty journal files and increase database partitions as needed.



That's because the journal files used by Oracle and Sybase are fixed in size and cannot grow without intervention. When the space fills up, the databases stop working until someone services it. While this may be acceptable for companies with their own in-house database administrator, it is not a realistic option for businesses with applications requiring turn-key and unattended database operation.

At the other end of the spectrum, there are open source databases. MySQL was originally designed without any thought given to the reliability of the operating system. But as customers began to lose data — as well as their confidence in MySQL — file mirroring was added. While this mirroring may provide the backup benefits of a soft-RAID, it also significantly degrades performance. And while it may reduce the chance of file corruption, it does not eliminate it entirely, because random access files, which are prone to corruption, are still used for the mirrored copies. Database clustering, where databases are clustered between two servers, appears to be the only option for MySQL users requiring real durability. But with a price tag of \$13,000 for MySQL clustering software, fixing the shortcomings of MySQL may not be the most cost-effective option.

PostgreSQL has recently added a write-ahead log to address their reliability problems. It operates on the same principle as a journal, except that the log is written before SQL is evaluated, rather than as the transaction commits. While this provides better reliability for PostgreSQL users, PostgreSQL documentation warns that there are still known problems with their write-ahead logging system, which could cause data loss. The answer? Many people offering PostgreSQL also sell database consulting services.

A company called FrontBase has also attempted to add durability to their database system to address problems with reliability. Their solution, called shadow paging, is strikingly similar to the MySQL strategy of mirroring files, except that FrontBase does it all in a single file. Besides slowing maintenance operations, this strategy puts all data in a single random access file. With only one file, there is no real redundancy. Shadow paging does nothing to solve the problem of recovering your data when that one file becomes corrupt.

SQLite is a free, open-source database which lacks most of the features of a relational database, including protection against data loss. Even so, it is widely used for applications that do not require reliability. If you choose this database product, keep frequent backups. There is no durability in SQLite.

FileMaker does not have anything like journaling and is prone to file corruption. This is one of many complaints that FileMaker customers have about the product.

4th Dimension (4D) does have a journal as a backup option, but does not appear to offer automatic journaling and recovery within the database. If you are a 4D developer, OpenBase offers a 4D plugin, which allows 4D to be used as the front-end for an OpenBase SQL database.

## Transaction Support

The attributes of atomicity, consistency and isolation are inseparable from a database's ability to provide transaction support.

Atomicity ensures that transactions either totally commit or they roll back. Consistency rolls back transactions that violate database rules. Isolation means that the database serializes transactions so they can operate without interfering with one another.

Databases that do not support transactions cannot offer A.C.I.D. compliance. And databases without A.C.I.D. compliance cannot support transactions.

Transaction support also ensures that complex data sets are saved correctly. A good example of a complex data set is in an accounting application where a credit to one account debits another. Wrapping both of these operations inside a single transaction ensures that either both succeed or neither succeeds.

Without transaction support, a system failure or other interruption could result in one account being credited without the corresponding accounts being debited — and no way to know of the disparity.

Transaction support is especially critical in a relational database system where typically large networks of records relate to other records.

## OpenBase SQL Transactions

All OpenBase SQL databases support transactions, so that either the entire transaction commits or it is entirely rolled back. This includes sets of changes, so that when a change successfully commits, you are guaranteed that all of the interrelated changes have been made to the database.

Power, network or other outages that occur during or immediately after a transaction cannot affect the consistency of data on an OpenBase database. This is doubly true, because OpenBase databases ensure transaction atomicity, consistency and isolation at the data-file level as well as at the transaction level.

### ***How do other databases compare?***

Most SQL databases these days support transactions by default. However, there are a variety of differences in implementation.

As noted, there are actually two levels which affect fault tolerance and transaction control. Some databases implement transactions in memory, but still rely on the file system, which may not ensure atomicity across the complete set of changes. Interrupting the process of making changes during a commit can sometimes lead to inconsistent databases or corruption. Any database that does not use a journal runs this risk.

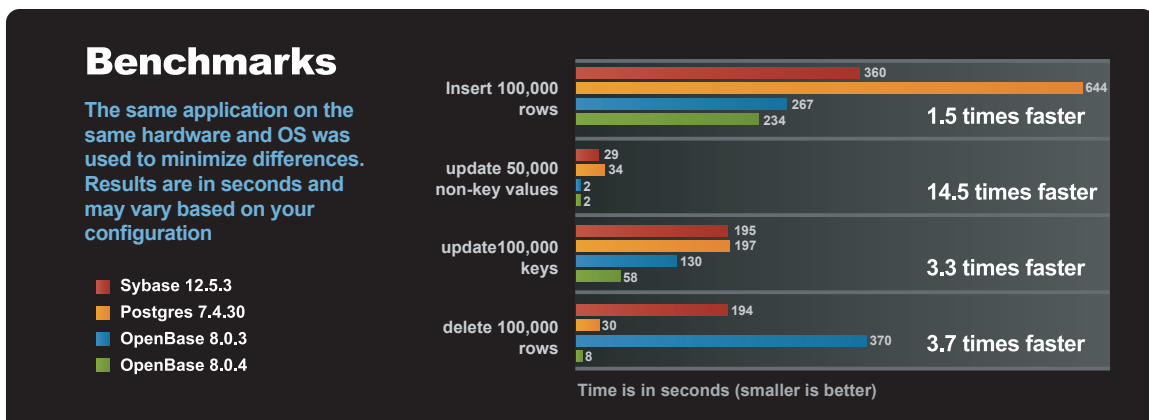
Sybase, Oracle and OpenBase SQL offer fault-tolerant transactions on this level. PostgreSQL also provides reasonable transaction support, although, as noted, they do warn of problems with the reliability of their write-ahead journal.

MySQL has only added transaction support relatively recently. Their implementation requires users to use a special mode which compromises MySQL's otherwise good response time. MySQL does not use a journal and so is subject to failure when interruptions occur. Products like FileMaker and SQLite do not support transactions.

## Performance Benchmarks

While performance is not one of the requirements of the A.C.I.D. standard, it is an important factor to consider in choosing a database.

While good performance is essential, we caution the reader not to over-value it. One way to get better performance is to sacrifice features such as transaction support and fault-tolerance. For most applications this is a mistake. A few milliseconds of performance gain will not matter if your database can not survive a system crash. All the factors discussed in choosing a database should be carefully balanced to determine the right mix of fault-tolerance, usability and performance for your application or business.



### About the benchmarks

The benchmarks above measure how databases perform out-of-the-box.

While a seasoned DBA could probably tweak all of these products to get more performance, the average small to medium sized business does not have a DBA.

The full value of OpenBase SQL is that you get great performance *without* an expert on-hand to tune and maintain your databases.

The benchmarks were performed by a seasoned consultant with 12 years of

Sybase experience. Great care was taken to make sure each database had the exact same data and indexes. Tests were performed on the exact same hardware.

Oracle is not included in the tests because we could not install it successfully. FrontBase is not included because it was so slow it could not complete the tests in a reasonable amount of time. MySQL is not included because the default setup does not support important features common to most other databases.

The first test, insertion of 100,000 records, measures both the performance of the server and the latency of the communication. Both are important metrics in real-life application performance and user experience. Round-trips to the database affect all SQL operations.

The three final tests were performed using a single SQL statement to compare raw server speeds. Since only one round-trip is made in each of these tests, network and communication latency become much less of a factor.

While OpenBase SQL beats both free “open source” solutions and costly “big iron” databases in performance comparisons, it is important to note that it *also* consistently outperforms these solutions in the areas of customer satisfaction and low cost-of-ownership. While these factors are harder to quantify with numbers, they can be just as important, or often, even more vital to actual business concerns and goals.

## Scalability

Besides out-of-the-box performance comparisons, it is important to consider the ability of a database to scale simply and transparently to handle user, data and application growth.

The performance and scalability features built into OpenBase SQL include:

- ◆ *Variable Length Records*  
Automatic compression of blank space in record data makes data files small and compact. Since blank space is removed, it also results in smaller reads, making file access performance considerably faster.
- ◆ *Multi-Threaded Server*  
Simultaneous processing of queries ensure that client applications never have to wait.
- ◆ *Index Data Clustering*  
Performance is improved by placing information likely to be retrieved together on the same disk pages.
- ◆ *Optimized Index Maintenance*  
Index changes are made with a single maintenance pass, delivering unbeatable response time. In contrast to databases that perform index insertion and maintenance each time a row is inserted or updated, OpenBase

SQL caches index changes until the server is idle or the indexes are needed. Since even a thousand inserts on an OpenBase SQL database require only a single maintenance pass on each index, the performance gains can be tremendous.

◆ *Data Page Caching*

Sophisticated page caching statistically reduces the number of reads and writes, further boosting performance.

## **Database footprint**

File size and growth of files are key considerations when it comes to scalability. Prospective customers often ask about file size because their existing databases have grown to be quite huge. But database size has as much to do with database design as it does the data itself.

Most databases are designed around fixed file offsets, which make it easy to calculate the location of data given a specific record number. For example, if you have a fixed record size of 200, and you need record 10, the record is located at position 2000 (10 X 200).

The problem with this fixed approach is that data files quickly grow out of control, since every record stored takes up the same maximum amount of room on the disk. With fixed file offsets, application designers have to be very careful to restrict the maximum length of record values to conserve space.

## **OpenBase SQL's dynamic variable length records**

In contrast, OpenBase SQL databases use a variable length record technology that eliminates blank space and compresses record size. With blank spaces accounting for as much as 80% of the space with some database products, variable length records permit OpenBase SQL files to be considerably smaller.

### ***How do other databases compare?***

We can give a concrete example of the advantage of a variable length record. After a few attempts to complete the benchmarks on Sybase (see page 6), our Sybase expert had to increase the size of the log device to 500 MB and the storage device to 200 MB, for a total footprint of 700 MB.

In comparison, OpenBase SQL's dynamic footprint and variable length records completed the tests with less than a 20 MB database footprint — while also providing significantly better performance. The numbers speak for themselves.

The next time a database vendor brags about being able to support terabyte databases — you may want to question whether this is an excessive requirement brought on by poor database server design.

Some databases also require pre-allocation of database files, which can, again, be problematic for businesses that do not have a database administrator.

The truth is that no one really knows how large your database will become, so if you need a database expert on-hand to increase the size of your files when you don't guess correctly, and you don't have one, you will have a problem.

In contrast, OpenBase SQL automatically expands and contracts without causing downtime. Of the SQL databases mentioned in this paper, only OpenBase offers variable length records with no pre-allocated space requirements. This is exactly what businesses need to lower ongoing IT costs.

## **Multi-Threading**

Multi-threading is another important feature to look for when considering performance and scalability. Multi-threading is the ability for a database server to perform multiple operations at exactly the same time. It is essential for most applications with multiple users — and for any application expected to support a growing number of users over time.

Without multi-threading, each request must be queued up and performed serially, one after another. This inability to process multiple requests simultaneously makes non-multi-threaded databases virtually unusable for multi-user systems or for Web sites receiving any type of significant traffic. Without it, a long query made by one user will block shorter queries and the database becomes a major bottleneck.

In a multi-threaded environment, A.C.I.D. becomes more complicated to implement. Inserts, updates and deletes must happen simultaneously in isolation, yet they also need to be “serialize-able” so that transactions maintain atomicity. The challenge is dealing with deadlocks. Deadlocks occur when, in the process of serializing multi-threaded operations ‘impossible situations’ develop where different threads wait for one another to release resources. The result is clients are left hanging in a deadlocked state. This is a common problem for database designers which is not easily solved.

### ***How do databases compare?***

Sybase, Oracle, PostgreSQL, FrontBase and OpenBase SQL all provide multi-threaded access. Oracle, however, offers it only in their higher priced licenses.

MySQL, SQLite, 4D and FileMaker are single-threaded, so requests are queued and processed one after another. This creates the potential for serious bottlenecks in multi-user environments.

Among the databases that offer multi-threading and A.C.I.D. compliance, there are varying strategies for handling serialization and avoiding deadlocks. Oracle is known for deadlocking. However, transaction modes can be tweaked by a

knowledgeable Oracle database administrator to help avoid them. Nevertheless, there is a cost to figuring out the magic customization which will work for each situation. Other databases, such as Sybase and PostgreSQL, have the same challenges.

OpenBase offers a deadlock avoidance mechanism which transverses the complex tree of dependencies between transactions competing for the same resources, detects deadlocks and resolves them by forcing one of the transactions to roll back. This avoids the possibility of users waiting for interdependent transactions that will never commit.

## Database Clustering

The ability to cluster a database, or replicate the same database on a series of servers, has many benefits. Database clustering can be used to provide complete redundancy and 100% uptime, allowing applications to failover to a second database when a primary database goes down. It mirrors databases in two different locations to provide uninterrupted database access if a site goes down. And it can provide better database access from remote locations with slow or unreliable communication links.

Clustering is powerful, but not all database clustering is the same. There are as many implementations as there are databases. Costs vary significantly — and too many vendors leave too many of the practical issues of clustering up to application programmers, which significantly impacts the cost and complexity of the solution.

Even among databases that do provide clustering, many of the challenges of clustering are not addressed in the design of the database. Instead, the expectation is that application designers will do the necessary programming to make it work.

Primary key generation, for instance, can be a serious problem in clustered databases. If the connection between clustered databases goes down and they are forced to operate independently, the databases may generate the same primary key and use it to insert a different record at each location. If this happens, the cluster will completely fail when the link comes back up and the keys conflict. OpenBase clustering resolves this problem. Other databases don't offer a solution.

Another often-overlooked issue is auto-failover, or the ability for clients to automatically connect to a second database server when the primary server goes down. The problem is that with most databases this is not automatic, so application programmers need to build their application with clustering in mind. With OpenBase clustering, however, failover is automatic and does not have to be addressed by the application.

## OpenBase SQL database clustering

OpenBase includes clustering at no additional cost with all of its Power Center licenses. OpenBase database clusters can be easily set up using the Database Configure panel in the OpenBase Manager application.

To address the primary key issue, OpenBase SQL automatically generates different ranges of keys for different databases in the cluster. If connections go down, databases can operate safely and independently until communication is restored.

OpenBase SQL failover is embedded into all database APIs. This means that when an application loses the connection to the primary server, the connection automatically establishes communication with the secondary server. As a result, all OpenBase applications automatically fail over, out of the box, so application designers don't have to design their applications with failover in mind.

### ***How do other databases compare?***

Both Oracle and Sybase offer database clustering at huge additional cost. MySQL has a clustered database solution offered by a third-party company. The cost is \$13,000 for the software, not including the support needed to get it running.

FrontBase provides a read-only cluster, allowing satellite sites to see company data, but not to modify it. While this may work fine for select query load balancing, it is not a practical solution for most uses, because updates are only allowed on the primary server.

PostgreSQL does not have a database clustering option.

While Sybase and Oracle both offer auto-failover capability, in both cases, application designers have to actually implement the failover in their program code.

Database clustering is a huge competitive advantage. It can provide complete redundancy and 100% uptime. But if you are not using a database that proactively addresses issues that make database clustering simple and affordable — from initial purchase price, to licensing, through set up and implementation, to day-to-day administration — database clustering can quickly become an ongoing expense and even a liability.

## Database Synchronization

Database synchronization can also be a critical consideration in choosing the right database for your business. Synchronization reconciles and merges differences between the tables of databases that frequently operate offline and independently of one another. Synchronization differs from clustering in that it

does not execute the same SQL on both servers — it compares the data to make them the same.

A good example of an application requiring database synchronization is one used by a mobile sales force. When salespeople are out of the office calling on clients, they use an offline copy of their database on their laptops. When they return, they press a button and any changes are synchronized with the company database.

## **OpenBase SQL database synchronization**

OpenBase uses both time stamps and the same special primary key generation to enable synchronization with the least amount of problems and overhead.

Databases that lack these built-in features for database synchronization force application developers to build synchronization into their applications, which adds unnecessary risk, overhead and cost.

### ***How do other databases compare?***

MySQL, PostgreSQL, FrontBase and Oracle do not have anything similar to the synchronization features built into OpenBase SQL. Sybase offers a form of synchronization, but it requires a significant additional investment and, often, even then, some modification of application software.

OpenBase provides a complete solution that takes the work out of building and managing applications that require database synchronization.

## **Hidden Costs**

Perhaps the most important question to ask in choosing a database is: “What business are the proponents or providers of this database in?”

The answer to this question will often reveal hidden costs that go far beyond the price of the purchase.

There is a very real cost to every break-down your business suffers because of some defect of the database. And there is a very real cost every time you must call in a database consultant to fix or maintain something that should never have broken or required maintenance to begin with.

The truth is that many database providers — and the consultants who recommend their products — have little incentive to deliver a well-designed product. They make their money from the consulting and other services to keep databases and businesses running. Difficult-to-use databases that need maintenance are great for consulting job security and revenues. But they are not in the best interest of the businesses that depend on them.

Not that we have anything against consultants making money. It's just that we know databases can be built so that they do not require ongoing configuration and tweaking just to stay running. Indeed, we work with consultants who make valuable contributions in the areas of software design, application support and business processes and who both recognize and appreciate the long-term value of a database that largely takes care of itself.

When practical issues of licensing, implementation, maintenance and support of the database are considered, it turns out that:

- ◆ Open source databases are *not* free! Lost business data, consulting and support fees cost money!
- ◆ The *actual cost* of Sybase and Oracle goes far beyond the purchase price! Database administrators and consultants are expensive!

## In Conclusion

As different as they may be in some ways, "big iron" and open source databases have one thing in common – they're complicated to set up and maintain – and that costs you money, again and again, in consulting fees and services.

In addition, many open source databases put your data at risk by leaving out important fault-tolerant design features. And the consequences of lost or corrupted data could be very expensive indeed.

For these reasons, we encourage you to think twice before entrusting valuable company data to a seemingly free alternative — or before putting it at the mercy of expensive database administrators.

OpenBase SQL delivers a solution that protects your data, minimizes maintenance and, ultimately, lowers the total cost of running your business.

OpenBase SQL isn't free because we are in the business of delivering the best possible database solutions and being paid fairly for it.

When you factor in database problems and consulting services that ultimately can add up to thousands of dollars, you will see:

- ◆ OpenBase SQL is a good value
- ◆ Consultants who recommend OpenBase SQL are committed to making your solution as stable, reliable and maintenance free as possible

We encourage you to try OpenBase SQL and see the difference for yourself.



## More information

To find out more about OpenBase SQL, contact us at:

### **OpenBase International (USA)**

info@openbase.com

Tel: +1 603-228-3339

Fax: +1 603-228-3637

16 Centre Street

Concord, NH 03301 USA

### **OpenBase Australia**

Frank Falco

frank@openbase.com

Tel: 0418-806-001

PO Box 2455

Kent Town SA 5071 Australia

### **OpenBase Japan**

Kimio Kawamoto

cal@openbase.com

Tel: +81-78-942-1278

Fax: +81-78-944-6157

1200-1 Tsuchiyama, Hiraoka-cho

Kakogawa-city, Hyogo 675-0104 Japan

Or go to: [www.openbase.com](http://www.openbase.com)



OpenBase is a registered trademark of OpenBase International, Ltd. All other product names mentioned herein may be trademarks or registered trademarks of their respective companies. OpenBase International shall not be liable for technical or editorial errors or omissions contained herein. The information in this document is subject to change without notice.

© 2004 OpenBase International, Ltd.