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.

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, Postgres and OpenBase SQL have solid strategies for fully complying in all four A.C.I.D. areas; SQLite and MySQL do not. And, since Real SQL Server is based on SQLite, it also does not fully comply.

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 like you to know. 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. Database systems are more prone to failure under these circumstances, 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 alone cannot solve the problem. While RAID does guard against a complete hard drive failure, RAID often just duplicates corrupted data — giving database owners a false sense of security.

Good database durability starts with the assumption that random access files will be corrupted; and that, when they do, the database software needs to be able to detect problems and 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 a master and working copy of database data, along with a realtime journal, which tracks all changes. Changes are

flushed to the journal as transactions commit, followed soon thereafter by batched flushing of changes to the working random access files.

Since an incomplete write can corrupt 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 the database back to its most recent state. In this way, OpenBase SQL provides a redundant and automatic system that keeps data safe.

In addition to maintaining data integrity, this journaling approach 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 of which 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 are open source databases. MySQL was originally designed without any mechanism to prevent or correct data corruption due to operating system failures. But as customers began to lose data — as well as their confidence in MySQL — file mirroring was added. While mirroring provides backup benefits similar to soft-RAID, it also significantly degrades performance. And while mirroring reduces 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 an affordable option.

Postgres has added a write-ahead log to address reliability issues. 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.

SQLite is a free, open-source database, which lacks most of the features of a relational database, including protections 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 built into SQLite.

In an effort to add some durability to SQLite, Real SQL Server now offers an SQL log so that the database can be restored from a backup if the main files become corrupt.

FileMaker does not have any type of log or journaling and is prone to file corruption. Loss of data is a frequently heard complaint from FileMaker users.

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 SQL 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.

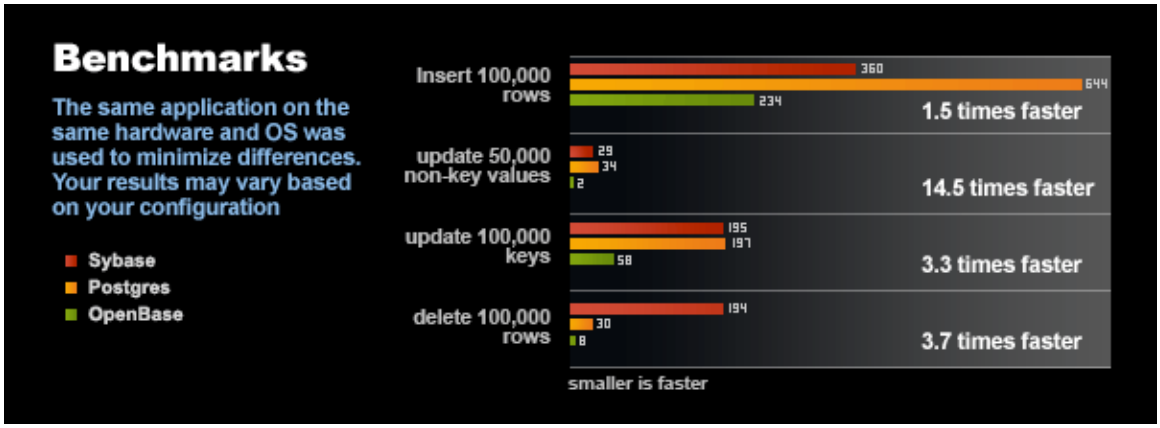
Sybase, Oracle, Postgres and OpenBase SQL offer fault-tolerant transactions on this level.

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 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 improve 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 database services and capabilities must be assessed and evaluated against requirements 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.

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 open source 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 important, in supporting a business and achieving its 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, reads are smaller, 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 with 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 promotes their ability 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 any downtime required. 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 assessing database performance and scalability. Multi-threading enables a database server to perform multiple operations at exactly the same time and is essential for most applications with multiple users — or any application expected to grow to support a multiple users over time.

A multi-threaded database server enables a database to take full advantage of multiple processors and multiple processor cores. For example, it can run multiple queries simultaneously on different processors and cores at the same time.

Some databases create separate processes to try to improve scalability. But because each request must still be queued up and performed serially, one after another, non-multi-threaded databases are virtually unusable for multi-user systems or for Web sites receiving any type of significant traffic. Without the ability to process multiple requests simultaneously, a long query made by one user will block shorter queries and the database becomes a major bottleneck.

In a multi-threaded environment, however, 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. Deadlocks can occur when, in the process of serializing multi-threaded operations 'impossible situations' develop as different threads wait for one another to release resources. The result? Clients left hanging in a deadlocked state. Deadlocks are a common problem in database design and not easily solved.

How do databases compare?

Sybase, Oracle and OpenBase SQL all provide multi-threaded access. Oracle, however, offers it only in their higher priced licenses. Postgres is not multi-threaded, but creates a completely new process for each thread it needs to execute at the same time.

MySQL, SQLite 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 different strategies for handling serialization and avoiding deadlocks. Oracle is has a reputation 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 what customization which will work for each situation. Sybase and Postgres address the challenge in a similar way.

In contrast, OpenBase SQL takes a unique approach to avoiding deadlocks. It uses a mechanism that 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 multiple servers — has many benefits. Database clustering can be used to provide complete redundancy and 100% uptime, with automatic failover of applications to a second database when a primary database goes down. Mirroring a database in two different locations provides uninterrupted database access if a site goes down. Remote database mirroring can also be used to provide faster database access in locations with slow or unreliable communication links.

Clustering is a powerful concept, but not all implementations of database clustering are the same. Indeed, there are as many clustering approaches as there are databases. Initial costs vary widely — and for those database providers that leave many of the practical aspects of clustering up to application developers, the on-going costs and complexity can be considerable. Simply put: any clustering issue not addressed in the design of the database will require the application designer to do the necessary programming to make it work.

Primary key generation, for instance, can be a serious problem in clustered databases. A primary key is a value generated by the database to uniquely identify a record. If the connection between clustered databases goes down and they continue to operate independently, each database may generate the same primary key and use it to insert a different record at each location. If this happens, the cluster will fail completely when the link comes back up and the keys conflict. OpenBase SQL database clustering addresses this issue. Other databases don’t.

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. For most databases this is not automatic, so application programmers need to build failover into their applications. With OpenBase SQL database 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 SQL database clusters are 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 makes failover transparent and consistent across applications by embedding automatic failover into all OpenBase SQL database APIs. As a result, when an application loses the connection to the primary database server, the interface automatically establishes communication with the secondary server. As a result, all OpenBase SQL database applications will automatically fail over, out-of-the-box, and 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, but at a huge additional cost. Both also offer failover, but, in both cases, application designers have to implement the failover in their program code. It is not automatic.

MySQL offers 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. Postgres does not have a database clustering option.

Database clustering, with its full redundancy and potential to delivery 100% uptime, can be critical to protecting your data and your business. But unless your database proactively addresses issues to make clustering simple and affordable, the costs — from initial purchase price, to licensing, through set up and implementation, to day-to-day administration — can quickly add up and even become a liability.

Database Synchronization

Database synchronization can be another 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 operation 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 special primary key generation to synchronize databases with minimal effort and overhead.

How do other databases compare?

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

MySQL, Postgres and Oracle do not offer anything like the synchronization features built into OpenBase SQL. Sybase offers a form of synchronization, but it requires a significant additional costs 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. The primary key system in OpenBase makes it possible to run databases independently without causing conflicts when they synchronize.

Database Security

Database security is a growing concern for companies with mobile users and remote office locations. While most databases have some level of access control built in, communication security is often overlooked.

As a result, many companies hire consultants to set up a virtual private network (VPN) or use signed certificates to strengthen communication security. In addition to added expense, these approaches become problematic if users need to access the database from public places with limited VPN access.

OpenBase SQL Security

OpenBase SQL addresses the issue through always-on encryption for secure database access, from anywhere, anytime. OpenBase uses the Diffie-Hellman (D-H) key exchange cryptographic protocol, which allows communications to be established securely over an insecure public network. D-H key exchange provides the basis for a variety of authentication protocols, including SSL and SSH. OpenBase uses this same trusted protocol to provide secure access to data — without having to configure a VPN.

How do other databases compare?

Most databases offer user password protection or some sort of encryption of data on the server. Other than OpenBase, however, none of the databases compared in this paper have encryption turned on by default. Postgres does offer SSL encryption as an option.

Hidden Costs

When comparing databases, it is important to remember that in addition to purchase price, there is a very real cost to any 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 in the first place.

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 purchase price.

The truth is that many database providers — and the consultants who recommend their products — have little incentive to deliver a database that is self-maintaining. They make money from ongoing configuration, tweaking, administration and other services just to keep databases running. Difficult-to-use databases may be great for consulting job security and service revenues, but they are not in the best interest of the businesses that depend on them.

We hasten to add that there are many ways that consultants can — and do — add tremendous value. Indeed, we work with consultants all the time who make critical contributions in areas such as software design, application support and business process improvement. In fact, these consultants recognize and appreciate the advantages of a database that takes care of itself — and use it to help their clients shift their investment from database maintenance to business innovation.

What's the real price?

When you consider the potential business impact of database problems — along with the practical issues of database licensing, implementation, maintenance and support — it turns out that:

- Open source databases are *not* free! Lost data, consulting and support fees cost money!
- The *actual cost* of Sybase, Oracle and Postgres go far beyond licensing! 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.

What's more, many open source databases put your data at risk through a lack of important fault-tolerant design features — and the consequences of lost or corrupted data can be very expensive indeed.

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

With OpenBase SQL, it is possible to choose an affordable, proven database that protects your data, minimizes maintenance and, ultimately, lowers the total cost of running your business. OpenBase SQL is open source, but it isn't free. That's because our business is to design and deliver the best possible database solution — and be paid fairly for it.

When you consider the impact of database problems on your business and the ongoing costs of data base administration, consulting and support on your bottom line, we believe you will see that:

- *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*

Experience the difference. Try OpenBase SQL today.

More information

To find out more about OpenBase SQL, visit: http://www.openbase.com