http://www.devx.com Printed from http://www.devx.com/dbzone/Article/20743

PostgreSQL vs. MySQL vs. Commercial Databases: It's All About What You Need

Can you trust the leading open-source database engines, PostgreSQL and MySQL, to deliver the performance and features that the Oracles, SQL Servers, and DB2s of the world do? Not just yet, but they could offer enough to meet your needs. Find out how they stack up against each other, as well as against the commercial alternatives.

by Tim Conrad

The database server is a fixture in almost every business these days. The common commercial databases, such as Oracle, Microsoft's SQL Server, and IBM's DB2 server, include many features that people have come to rely on to make their database servers "enterprise worthy". These features include advanced database storage, data management tools, information replication, and tools to back it up.

During the past ten years, the open source community has improved the quality of its software, making it more enterprise worthy. As a result, enterprises have shown an interest in migrating from proprietary, commercial software to open-source software in recent years. For example, businesses around the world commonly use Linux, the Perl programming language, the Apache Web server, and the two leading open-source database engines, PostgreSQL and MySQL.

This article compares PostgreSQL and MySQL, both to each other as well as with their commercial counterparts. Rather than examining the MySQL MAX product based on SAP's database engine, it looks at the more widely deployed "original" MySQL.

Questions about MySQL and PostgreSQL often relate to speed. Even though current Postgres releases have gotten much faster, earlier versions were known to be slow. But speed isn't everything when it comes to choosing a good database engine. This comparison is based on features rather than speed. If all you need is raw speed, you can get it in other ways.

How It All Began

History of PostgreSQL

The PostgreSQL relational database system (RDBMS) came from the POSTGRES project at the University of California at Berkley. Professor Michael Stonebraker started the project in 1986 to replace the aging Ingres RDBMS, and DARPA, the National Science Foundation, the Army Research Office, and ESL, Inc. sponsored it. While known as the POSTGRES project, the database assumed various roles in different organizations, including an asteroid tracking database, a financial data analysis system, and an educational tool.

POSTGRES originally used a language called PostQUEL for accessing database information. In 1994, Andrew Yu and Jolly Chen added the POSTGRES SQL interpreter, originally known as Postgres95. Postgres95 was then re-licensed under the Berkley software license and shortly thereafter was renamed PostgreSQL.

Brief History of MySQL

Prior to creating MySQL, the people that wrote it used mSQL to connect to their own low-level data structure. They discovered that mSQL lacked the features and speed they wanted and decided to write their own front

end instead. Thus began the life of MySQL as the product.

The Ins and Outs of Licensing
Both MySQL and PostgreSQL have different licenses, and understanding how they work is important when incorporating these products into enterprise projects. Different licenses fulfill different needs, and they have different requirements.

MySQL AB, the company that owns and produces MySQL, has two licenses available for its database product:

1. **GNU General Public License (GPL) for GPL projects.** If your project is 100 percent GPL in its distribution, you can use this license. To fully comply, you must distribute your application, along with the source code. You also can use this license if you don't intend to ever distribute your project internally or externally.
2. **Commercial License for commercial applications.** An example of the use for this license is when you don't want to distribute the source code for your application. This includes database drivers as well. You can't use the MySQL database drivers with a commercial application unless it's either distributed under the GPL license or you have a Commercial License.

PostgreSQL has a much simpler licensing scheme. It is released under the Berkley License, which allows for any use as long as a copy of the Berkley License is included with it. This means that you can release a commercial product that uses PostgreSQL or is a derivative of PostgreSQL without including source code.

The Features You've Come to Expect
The database comparison boils down to the features that each database engine provides (see Table 1). Database administrators that have worked with commercial database engines such as Oracle, DB2, or MS-SQL have come to rely on a fairly broad feature set. This section compares these commercial databases with the open-source databases.

Data Storage
MySQL has several different data storage mechanisms available. It originally used ISAM/MyISAM, which was then replaced by the more advanced InnoDB. Other storage mechanisms are available, but this discussion focuses primarily on using InnoDB tables because it typically has the most advanced database feature set and is the default table type in MySQL version 4.x. When choosing a MySQL storage mechanism, make sure you read up on all of the features you plan on implementing. While researching this article, I found that some features exist in certain storage mechanisms, but not in others. Most notably, InnoDB and BDB are the only table types that are transaction-safe. PostgreSQL, on the other hand, uses only one data storage mechanism, the aptly named Postgres storage system.

Data Integrity
One of the critical features of any database engine is data integrity. ACID (Atomic, Consistent, Isolated, Durable) compliance is a qualification that assures data integrity. ACID essentially means that when a transaction is performed within a database, either the whole transaction is successful and the information is written to the database, or nothing is written. Both PostgreSQL and MySQL support ACID-compliant transaction functionality.

Both databases also support partial rollbacks of transactions, and they know how to deal with deadlocks. MySQL uses traditional row-level locking. PostgreSQL uses something called Multi Version Concurrency Control (MVCC) by default. MVCC is a little different from row-level locking in that transactions on the database are performed on a snapshot of the data and then serialized. New versions of PostgreSQL support standard row-level locking as an option, but MVCC is the preferred method.

The Advanced Features
PostgreSQL has many of the database features that Oracle, DB2, or MS-SQL has, including triggers, views, inheritance, sequences, stored procedures, cursors, and user-defined data types. MySQL's development version, version 5.0, supports views, stored procedures, and cursors. MySQL's future version, version 5.1, will support triggers. MySQL does, however, support the advanced feature of data partitioning within a database. PostgreSQL does not.

Stored Procedures and Triggers

While PostgreSQL has had support for stored procedures and triggers for quite some time now, MySQL has support for these only in development versions 5.0 and beyond. PostgreSQL's query language, PL/pgSQL, is very similar to Oracle's PL/SQL. In addition, PostgreSQL's procedures and triggers can be written in other languages as well, such as PL/TCL, PL/perl, and PL/python. These additional languages come in two basic flavors, safe and unsafe. Safe allows only for use of things in the programming language that don't affect the host system negatively, such as direct access to the file system.

Indexes
Oracle is known for the amount of tweaking it allows for databases, especially when it comes to indexing. Overall, experienced Oracle users will probably find the indexing strategies employed by these open-source databases quite primitive. Both PostgreSQL and MySQL support single column, multi-column, unique, and primary key indexes. MySQL supports full text indexes out of the box, and PostgreSQL can support full text indexes with some changes to the database that are included with the source.

Data Types
Databases hold data, and the types of data that a database can hold are called data types. Both PostgreSQL and MySQL support most standard data types. In the past few years, large object support has become increasingly popular, and both databases support this as well. PostgreSQL supports user-defined data types, while MySQL does not. MySQL and PostgreSQL also both support the storing of geographic features, known as GIS (Geographic Information System). PostgreSQL additionally has network-aware data types that recognize Ipv4 and Ipv6 data types.

Replication
Another major feature of enterprise-level databases is support for replication. Both MySQL and PostgreSQL have support for single-master, multi-slave replication scenarios. This base level of replication is included with the distributions of the software, and the source code is open. PostgreSQL offers additional support for multi-master, multi-slave replication from a third-party vendor, as well as additional replication methods.

Platform Support
While both Oracle and DB2 run on multiple platforms, Microsoft's SQL Server is limited to Windows. Both MySQL and PostgreSQL support many different platforms, including Windows, Linux, FreeBSD, and MacOSX. MySQL uses a threaded model for server processes, wherein all of the users connect to a single database daemon for access. PostgreSQL uses a non-threaded model where every new connection to the database gets a new database process.

Database Interface Methods
PostgreSQL and MySQL both support ODBC and JDBC for network connectivity, as well as native database access methods. These native methods provide access via the network in both plain text methods and, for a higher level of security, SSL-encrypted methods.

Another important part of database interface methods is authentication for the database. MySQL uses a simple method to store all of its authentication information inside a table. When users attempt to access a database, MySQL compares their credentials against this database, verifying from which machines the users can connect and to what resources they have access.

PostgreSQL can use a similar method, but it also has some others. For example, it can use a hosts file for database access to define which remote users can connect to which database. It can also use the local authentication systems for database access (e.g., your Unix password would also be your PostgreSQL password).

A number of programming methods also provide ways to access these databases. Both PostgreSQL and MySQL support access via C/C++, Java, Perl, Python, and PHP. PostgresSQL also has internal programming languages for writing stored procedures and triggers, among them are pl/pgsql, pl/tcl, and pl/perl.

Backups
When it comes to backups, open-source databases may not completely fulfill your needs. Both databases come with scripts to facilitate a simple text dump of your database data and its schema. Both database solutions also provide methods for doing a hot-database backup, or backing up your database without shutting it down. Many commercial backup tools, such as Vertias NetBackup or Tivoli TSM, have agents that provide online backups of commercial databases. A quick Web search returned only a few vendors that create agents for PostgreSQL and MySQL. The overall coverage appears limited.

Backups also include simple database recovery from soft failures, such as database crashes or unexpected power failures. PostgreSQL uses a system called Write Ahead Logging to provide database consistency checking. MySQL has database consistency checking only under InnoDB table types.

GUI Tools
Many people use GUI tools to manage their databases. Many such tools—both open source and commercial—are available for MySQL and PostgreSQL. These tools can be either applications that run natively on your operating system or Web-based tools. Many of these tools are closely modeled after tools available to commercial databases.

Data Migration
Both MySQL and PostgreSQL have database migration utilities to help migrate data from commercial databases. These utilities are available from third parties as either open-source or commercial tools. PostgreSQL also comes with tools to help migrate data from Oracle and MySQL. Obviously, the more complex your schema, the more difficult the conversion will be, and some of these tools may not completely migrate everything perfectly.

Training and Support
The issue of support has mitigated acceptance for open-source software in the enterprise. Many do not realize that support is available for many open-source products—beyond Web sites and mailing lists. MySQL AB provides support for MySQL, and several companies, including Command Prompt, Inc. and PostgreSQL, Inc., provide support for PostgreSQL. These offerings include support levels that rival commercial databases, many providing 365x24 support.

Training is also available on a wide variety of topics for both PostgreSQL and MySQL. MySQL AB provides training in cities around the world, with topics ranging from administration to writing Web-based applications using MySQL. PostgreSQL training is also available from dbExperts and Big Nerd Ranch.

Who Else Uses Them?
A number of large companies use both open-source databases in various ways. Both database engines have somewhat large database installations in use. I use the word *somewhat* because data storage is a relative term. Oracle and DB2 can scale to terabytes of data storage fairly easily. MySQL and PostgreSQL are known to run well into the hundreds of gigabytes, but few companies use the databases above that range.

Cox Communications uses MySQL to manage information related to its cable modem business. NASA uses MySQL to store information about public contracts. Slashdot, a widely read online publication, uses MySQL to store all of the information related to its site. The Associated Press uses MySQL to serve various types of information, including access to the U.S. Census and Olympic results.

You probably use PostgreSQL indirectly on a fairly regular basis. Afilias, which manages the .ORG registration, uses PostgreSQL to store all of .ORG registry information. The American Chemical Society uses PostgreSQL to store documents that exist only within that database. BASF uses PostgreSQL in a shopping platform for its agriculture products. The World, a media company, has built much of its infrastructure around the use of PostgreSQL.

If It Ain't Broke, Don't Extend it
An axiom of open-source software is that developers write it to "scratch an itch." This is a good practice when the itch is something like a failing disk, which compels the developer to improve the tools that work on the disk. However, in regards to databases, the itch flares up only when data surpasses certain limits, such as size or complexity. PostgreSQL and MySQL boast widespread use for relatively small databases (under 100GB, for example). Once the data grows larger than 100GB or so, the number of users drops off drastically. At that point, working through large-database-related issues becomes more of a problem.

The itch axiom also applies when working in some of the more "buzzword-complaint" areas. Some of the more advanced features in the open-source databases (such as replication) are nowhere near what you'd find on commercial alternatives. Quite simply, most users don't need replication at the levels that Oracle, DB2, or MS-SQL offer; therefore, PostgreSQL and MySQL developers don't get the itch to improve it.

The great thing about open source software, though, is that it's pretty easy to try out and has lots of freely available online documentation to help you learn the products. While these databases may not be optimal for every project, they work very well for others. If you're curious—and this article hasn't answered your particular usage questions, take MySQL or PostgreSQL out for a spin and see if they meet your needs.

Tim Conrad *lives and works in New York City, where he supports the network infrastructure at an educational services company. In his spare time he likes to go to concerts, drink beer, play guitar, and spend quality time with his computers. Email Tim.*