PostgreSQL vs. MySQL

A Comparison of Enterprise Suitability

A Fact Sheet by EnterpriseDB

www.EnterpriseDB.com

EnterpriseDB™

Table of Contents

Introduction	3
Performance	3
Multiple Storage Engines and Query Optimization	4
Transactional Support	4
Referential Integrity	5
Procedural Language Support	5
Support for Triggers	5
Supported Authentication Methods	6
Conclusion	6
PostgreSQL vs. MySQL Feature Comparison	7

Introduction

The idea that the open source PostgreSQL database is generally better suited to enterprise environments than its popular competitor, MySQL, is commonplace enough among IT professionals that remarks to that effect often pass without comment. However, for the enterprise architect, developer, or DBA charged with selecting a database for the company's next new application, or for the IT department seeking a truly enterprise-class alternative to an in-production MySQL database that is unable to perform as expected, a consideration of the particular database features and functionality behind that generality is valuable.

This paper discusses a wide range of PostgreSQL features, functionality, and performance that are critical to enterprise database deployments and contrasts them to MySQL's current offering. Topics addressed will include database performance, query optimization, transactional support via ACID, data durability, referential integrity, support for procedural languages and triggers, and support for industry-standard authentication methods.

Many of MySQL's most problematic limitations become apparent only after the database is already in production; however, re-architecting a solution around a more suitable database is especially time-consuming, expensive, and frustrating at that time. As a result, the analysis below will be of particular interest to those who have little or no experience with MySQL in deployment.

Performance

MySQL is considered by some to be a high-performance database, and this can be true for certain classes of read-mostly, Web-based applications. However, acceptable performance is generally considered to be only available from MySQL's default storage engine, MyISAM, which has several important limitations that make it unsuitable for enterprise deployment.

MyISAM is based on IBM's Indexed Sequential Access Method (ISAM) for data storage, which was designed primarily for extremely fast retrieval of keyed data. In this context, MyISAM is considered to provide adequate speed. However, MySQL's implementation of ISAM is known for causing routine and somewhat antiquated database problems, such as:

Data Corruption
Once committed to an enterprise database, data should remain usable and intact. However, MyISAM's data corruption problems are so infamously common that the *myisamchk* utility, which is used to find corruption in MySQL data files, is scheduled as a daily operation on many production MySQL systems. Further, in cases where catalog corruption occurs, it is difficult, if not impossible, to recover successfully.

Lock Contention
Row-level locking is well understood as a foundational requirement of enterprise database operation. MyISAM lasks this feature, and instead locks rows at the less-granular block level. This simpler and less-granular approach causes significant lock contention in multi-user environments.

Offline Management
Because MyISAM does not support multi-versioning, many routine administrative tasks, like adding a column to a table, become impossible to perform during normal

use. This lack of multi-versioning often requires the DBA to take down the database just to perform simple changes.

Some of the problems with MyISAM may be avoided on a table-by-table basis using alternative storage engines, such as InnoDB. However, the MySQL catalog only operates on MyISAM. Because of this limitation, catalog corruption and administrative tasks are still problematic. MySQL developer and co-founder Michael "Monty" Widenius has acknowledged this as a severe limitation, with a partial fix scheduled for MySQL 6.1.

In contrast, PostgreSQL is very differently architected and presents none of these problems. All PostgreSQL operations are multi-versioned using Multi-Version Concurrency Control (MVCC). As a result, common operations such as re-indexing, adding or dropping columns, and recreating views can be performed online and without excessive locking, allowing a DBA to perform routine maintenance transparently to the database's users and to the applications running against the database.

Multiple Storage Engines and Query Optimization

PostgreSQL has supported multiple storage engines since the late 1980's. It wasn't until just recently that MySQL implemented similar functionality via their newly developed Pluggable Storage Engine API. However, MySQL's API is not robust enough to allow for accurate query optimization and makes query planning and tuning difficult, if not impossible, to perform. MySQL's API presents only two optimizer-related function calls to storage engine developers. These calls, if implemented, can assist in costing plans for the query optimizer. However, because these calls do not present the context of the query to the storage engines, the engines themselves cannot accurately return a proper estimate. Often, this results in the generation of slow query plans.

Transactional Support

Enterprise-class databases must include transactional support. In database terms, a transaction is a single unit of work, which may include two or more operations. For example, in a simple debit/credit operation, two operations must be performed. First, an amount is debited from an account. Second, the same amount is credited to another account. What if the first operation succeeded but the second failed? In a database without transaction support, the application would be responsible to notice the failure and correct it. Conversely, in a database that supports transactions, the database would properly undo the debit; this is called atomicity.

The following properties, which are referred to by the acronym "ACID", are generally understood to be required for the reliable handling of transactions:

- **A**tomicity: guarantees that either all or none of the tasks within a transaction are performed.
- **C**onsistency: ensures that, irrespective of the success or failure of a transaction, the database will remain in a consistent state.
- **I**solation: makes operations in a transaction appear isolated from all other operations.
- **D**urability: guarantees that when a commit succeeds, the transaction will persist and not be undone.

Simply put, PostgreSQL is ACID-compliant, but MyISAM is not, either with respect to the data in the database or with respect to the database metadata. As a result, PostgreSQL reliably handles transactions, but MyISAM does not.

Referential Integrity

Referential integrity, the guaranteed consistency between coupled tables, is another requirement for enterprise-class database operation. An example of referential integrity may be found in an employee database in which employees are linked to their departments using a field. In the employee table, this field would be declared as a foreign key to the department table, which contains a list of all company departments. If referential integrity were not enforced, a department could be dropped from the department table, leaving employees working for a non-existent department.

PostgreSQL maintains referential integrity. In contrast, very few MySQL storage engines support referential integrity, and, because MySQL will quietly accept the syntax for creating referential integrity rules without actually enforcing them, administrators are often forced to double-check their changes.

Procedural Language Support

From business processes to utility functions, procedural languages allow developers and DBAs to implement programmatic logic within the database, speeding up access and response times by reducing network round-trips and by executing more closely to the data. While procedural language support was just recently added to MySQL, PostgreSQL has supported procedural languages for both Tcl and a PL/SQL-like dialect since version 6.3 in 1998.

Because PostgreSQL is an extensible database, developers can write their own procedural language handlers. As a result, PostgreSQL has stable implementations of procedural language handlers for many common programming languages, such as Perl, Tcl, Python, Ruby, and PHP.

Support for Triggers

Support for triggers was also only recently added to MySQL. Unfortunately, MySQL's triggers were only implemented per-statement, lacking the ability to execute per-row. This is a significant omission, as row-level triggers are the most commonly implemented trigger. Row-level triggers have been supported in PostgreSQL since version 6.3 was introduced in 1998.

Supported Authentication Methods

PostgreSQL offers a wide variety of well-known, industry-standard methods to authenticate database users, including trust, password, GSSAPI, SSPI, Kerberos, Ident, LDAP, and PAM. MySQL only supports its own, non-standard, non-pluggable, internal authentication system. This makes enterprise use difficult, because database accounts cannot be centrally provisioned or managed.

Conclusion

The idea that PostgreSQL is better suited for enterprise deployment than MySQL is rooted in concrete differences between the two databases' features, functionality, and performance. MySQL is widely deployed, but its legitimate uses are limited to a narrow range of applications that can tolerate MySQL's inherent limitations. Many enterprise IT departments have optimistically selected MySQL because of its popularity and "hit the wall" once the database is in production. The table following this paper presents an overview of enterprise attributes of PostgreSQL that are absent or limited in MySQL.

PostgreSQL vs. MySQL Feature Comparison

Feature	MySQL	Postgres Plus Advanced Server	Comments
VLDB, Data Warehousing, Business Intelligence			
Bulk Data Loader	**Y**	**Y**	
Direct Path Load	**N**	**Y**	EDB*Loader
Function-based Indexes	**N**	**Y**	
Optimizer Statistics Management	**Y**	**Y**	
Pipelined Table Functions	**N**	**Y**	Use SETOF function
Partitioning	**Y**	**Y**	
Parallel Database	**N**	**Y**	GridSQL provides parallel query
High Availability			
Physical Standby Database	**N**	**Y**	
Online Operations	**N**	**Y**	
Online Backup	**Y***	**Y**	* Dependent on storage engine
Online Reorganization	**N**	**Y**	
Content Management			
Text Data Support / Access	**Y***	**Y**	* InnoDB does not support text
Spatial Data Support	**Y***	**Y**	* InnoDB cannot index
Information Integration			
Capture / Consumedata / Transactions / Events	**Y**	**Y**	Through JMS
Database Features			
ANSI SQL Support	**Y***	**Y**	* Dependent on selected mode
ACID Compliance	**Y***	**Y**	* Dependent on storage engine
Transactions	**Y***	**Y**	* Dependent on storage engine
Nested Transactions	**Y**	**Y**	
ANSI Constraints	**Y***	**Y**	* Dependent on storage engine
Check Constraints	**N**	**Y**	
Synonyms	**N**	**Y**	
Cursors	**Y***	**Y**	* Limited to read-only
Globalization Support	**Y**	**Y**	
Index-organized Tables	**Y**	**Y**	
Instead-of Triggers	**N**	**Y**	
Nested Triggers	**N**	**Y**	
LOB Support	**Y**	**Y**	
User-defined Datatypes	**N**	**Y**	
Domains	**N**	**Y**	
Temporary Tables	**Y**	**Y**	
JDBC Drivers	**Y**	**Y**	
Object-relational Extensions	**N**	**Y**	
Table Collections	**N**	**Y**	
Bulk Binding	**N**	**Y**	
Bulk Collect	**N**	**Y**	
XML Datatype Support	**N**	**Y**	
XML Functions	**Y**	**Y**	
Partial Indexes	**N**	**Y**	
IP Address Datatype	**N**	**Y**	

Feature	MySQL	Postgres Plus Advanced Server	Comments
Distributed			
Basic Replication	**Y**	**Y**	
Oracle Replication (To and From)	**N**	**Y**	
Distributed Queries	**Y**	**Y**	
Distributed Transactions	**Y**	**Y**	
Heterogeneous Connectivity	**N**	**Y***	* db links over ODBC planned for future release. Currently supports db links to Oracle, PostgreSQL, and EnterpriseDB.
Networking			
Connection Manager	**Y**	**Y**	
Multiprotocol Connectivity	**Y**	**Y**	
Connection Pooling	**Y**	**Y**	
System Management			
Tablespace Support	**Y**	**Y**	
Online Backup and Recovery	**N***	**Y**	* Dependent on storage engine
GUI for Performance Management	**Y**	**Y**	
GUI Framework for Database / Network Management	**Y**	**Y**	
Security			
Virtual Private Database	**N**	**Y***	Security policies for row-level security (have to download Veil)
Fine-grained Auditing	**N**	**N**	
Roles or Groups	**N**	**Y**	
Enterprise User Security	**N**	**Y**	
Password Management	**Y**	**Y**	
Encryption	**Y**	**Y**	
PAM Authentication	**N**	**Y**	
LDAP Support	**N**	**Y**	
Development			
Precompiler Support	**Y**	**Y**	
OCI Support	**N**	**Y**	OCL-compatible layer
PL/SQL Stored Procedures	**N**	**Y**	
PL/SQL Functions	**N**	**Y**	
PL/SQL Packages	**N**	**Y**	
PL/SQL Triggers	**N**	**Y**	
Java Stored Procedures	**N**	**Y**	
Perl Stored Procedures	**N**	**Y**	
TCL Stored Procedures	**N**	**Y**	
Python Stored Procedures	**N**	**Y**	
Ruby Stored Procedures	**N**	**Y**	
PHP Stored Procedures	**N**	**Y**	
.NET Connector	**Y**	**Y**	
ODBC	**Y**	**Y**	
JDBC	**Y**	**Y**	

Feature	MySQL	Postgres Plus Advanced Server	Comments
PHP	**Y**	**Y**	
C API	**Y**	**Y**	

Migration

GUI Tool to Assist	**Y**	**Y**	
Command Line Tool	**N***	**Y**	* Can be self-scripted
SQL Server Migration	**Y**	**Y**	
Sybase Migration	**N**	**Y**	
Oracle Migration	**Y**	**Y**	
- data	**Y**	**Y**	
- schema	**Y**	**Y**	
- stored procedures	**N**	**Y**	EnterpriseDB Migration Toolkit
- functions	**N**	**Y**	EnterpriseDB Migration Toolkit
- triggers	**N**	**Y**	EnterpriseDB Migration Toolkit
- packages	**N**	**Y**	
Oracle-like Tools	**N**	**Y**	EDB*Plus, EDB*Loader

About the Postgres Plus Family

EnterpriseDB's Postgres Plus products are ideally suited for transaction-intensive applications and deliver the performance, scalability, and reliability required for enterprise-class workloads. Postgres Plus is an open source distribution of the PostgreSQL database and includes significant performance benefits and important ease-of-use capabilities for developers and DBAs. Bundled into a one-click, cross-platform installer, Postgres Plus is targeted at developers of next-generation applications and sets a new standard for commercial distributions of open source databases.

Postgres Plus Advanced Server is a commercially licensed product that adds advanced capabilities to Postgres Plus, including robust Oracle compatibility, dynamic performance tuning, and sophisticated management and monitoring.

Customer case studies detailing leading organizations' deployments of PostgreSQL in response to MySQL's limitations are available on EnterpriseDB's website. Please visit www.enterprisedb.com to learn more.

CORPORATE HEADQUARTERS
EnterpriseDB Corporation
499 Thornall Street, Suite 200
Edison, NJ 08837-2210, USA
T +1 732 331 1300
F +1 732 331 1301

EMEA HEADQUARTERS
EnterpriseDB Ltd
90 Long Acre, Covent Garden
London WC2E 9RZ, United Kingdom
T +44 (0) 207 849 3093
F +44 (0) 207 849 3200