PostgreSQL 7.0

Open Source Relational Database

White Paper

Great Bridge, LLC
May 2000

Table of Contents

Introduction

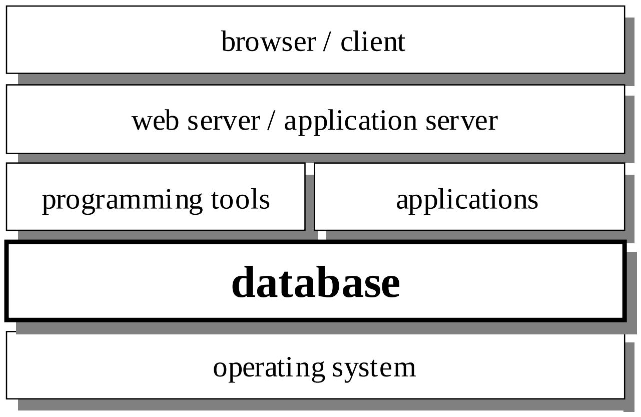
Great Bridge, LLC, was formed in May 2000, with the express purpose of marketing, supporting, and contributing to the development of **open source software**, which we believe will change the world as fundamentally as the Internet itself has. The concept is not a new one, although its widespread application to business systems is. For decades, scientists and technologists have practiced rigorous peer review of their research, writings, and even software code – before the commercial Internet as we now know it existed, programmers would electronically transmit their code across painfully slow first-generation modems, "hacking" each other's work, fixing bugs, and substantially raising the quality of the work product. For an excellent discussion of the history of Open Source, we highly recommend Eric Raymond's essay "The Cathedral and the Bazaar," available at http://www.tuxedo.org/~esr/writings/cathedral-bazaar.

Open source software is freely available, in its raw, uncompiled "source" form, for any and all interested parties to implement, modify, and improve. The Linux operating system is perhaps the most celebrated example of open source development in action; other notable successes include the Apache web server, the Sendmail electronic mail program, both of which are in use in *over half of their respective markets worldwide* – handily trouncing widely available commercial alternatives. There are a variety of licenses under which open source software can be had, which vary with the lineage of the product. Great Bridge supports the Berkeley/BSD-style license under which the PostgreSQL database is offered; we believe it to be the simplest to understand (it's free for any use, with no restrictions on what you can do with the code), and offers the fewest barriers to widespread commercial adoption (some licenses restrict use of the code in certain situations, others "force" any modifications to the code to be contributed back to the larger community). A detailed list of the different varieties of open source licenses can be found at http://www.opensource.org/licenses.

It seems counter-intuitive at first – that a relatively unregulated global community of computer programmers, many of whom work from home, on weekends, around the edges of their “day” jobs, could somehow produce better, more reliable software than a focused internal effort from a multi-billion dollar software company. But the reality behind that seeming disconnect is twofold:

1) Commercial software is more complex, and hence buggier, than ever. While the growth of features in products from industry leaders such as Microsoft and Oracle has been impressive, in some ways, the software has never been of a lower quality.
2) While a software company may have hundreds, even thousands of developers working on a piece of software, open source allows *millions* of skilled programmers to test, fix, and improve the product. In most cases, customers who buy from a proprietary software company do not have access to the source code – and are hence stuck with unfixed “bugs,” slow release cycles, and seemingly interminable waits for the features and fixes that they *know they need*. When the source is freely available to the widest possible community of qualified professionals, the quality and reliability of the software demonstrably improves. Or, as Eric Raymond puts it, “Given enough eyeballs, all bugs are shallow.”

By virtue of our chairman’s association with industry pioneer Red Hat, Great Bridge understands and is fully-committed to the open source development process- and we believed that the database level of the information technology “stack” offered tremendous possibilities for building a support-driven business in the open source software world. After months of research and testing, we came to the overwhelming conclusion that PostgreSQL is, in the widely quoted words of one of its leading developers, “the most advanced open source database anywhere.”



The Information Technology “stack”

This white paper summarizes our initial findings, and offers a high-level overview of the features and functionality of PostgreSQL 7.0.

The History of PostgreSQL

Like many other software products – open and closed – PostgreSQL has its roots in the academic environment. It began as a project called Ingres, developed at the University of California at Berkeley (1977-1985). The Ingres code was enhanced by Relational Technologies/Ingres Corporation, from which one of the first commercially successful relational database servers was produced. (Ingres Corp. was later purchased by Computer Associates.) Later at Berkeley, Michael Stonebraker led a team to develop an object-relational database server called Postgres (1986-1994). This Postgres code was assimilated by Illustra Corp. and also developed into a commercial product. (Illustra was later purchased by Informix and integrated into Informix's Universal Server; Stonebraker is now CTO of Informix.)

In 1995, two Berkeley graduate students added SQL capabilities to Postgres; a year later, the core steering group for the now-active Postgres project coalesced around open-source veteran Marc Fournier. Today, the Postgres development community consists of hundreds of developers worldwide, guided by a six-member core steering group, similar to the way that Linux creator Linus Torvalds oversees and screens new enhancements for Linux.

PostgreSQL, now at Release 7.0, has matured into a product capable of significant transaction throughput, complex queries, commercial-grade SQL support, and complex data types. The upcoming Release 7.1 promises to be even more robust.

Technical Description of PostgreSQL

PostgreSQL is an Object-Relational Database Management System (ORDBMS), a combination of the best features of both an object-oriented database (OODB) and a relational database management system (RDBMS).

As an ANSI Standard SQL 1992 relational database, PostgreSQL supports the same verbs (commands) and syntax widely used by relational databases throughout the industry. This allows Structured Query Language (SQL) code written on other platforms to run on PostgreSQL with little or no modification. Similarly, since the SQL grammar conforms to the 1992 standard, database programmers knowing SQL can readily become productive in PostgreSQL.

PostgreSQL also supports some of the features of an object-oriented database, thus giving the user additional power and flexibility. This support allows the user to create his or her own datatypes, functions, and operators. In addition, table inheritance allows one table to inherit columns from another table. It is rare to find this combination of features and flexibility in even the most sophisticated commercial offerings.

PostgreSQL is comprised of over 250,000 lines of C code contained in over 850 files. It was developed in a modular fashion since its inception at Berkeley in 1977, and is today the product of a 23 year long collaborative effort among some of the software industry's best developers. There can be little argument that it is the most advanced open-source database server available today.

Compliance

American National Standards Institute (ANSI)

The American National Standards Institute has a standard that describes the minimum Structured Query Language (SQL) and Data Definition Language (DDL) requirements for a database. This is referred to as the ANSI 92 SQL standard, or SQL92. PostgreSQL version 7.0 is compliant with the "entry-level" SQL92 standard. There are some minor features that remain for "full" compliance, all of which are expected to be incorporated into the upcoming 7.1 release.

Open Database Connectivity (ODBC)

Microsoft's Open Database Connectivity (ODBC) specification is a widely accepted application programming interface (API) for database access. It has been implemented on a variety of platforms, including the Macintosh and various Linux/Unix versions; however, not surprisingly, ODBC tends to be most popular as a Microsoft Windows solution. The PostgreSQL ODBC driver complies with the core grammar requirements of the ODBC 3.0 specification.

Java Database Connectivity (JDBC)

The Java Database Connectivity (JDBC) specification is a well-defined interface originally specified by Sun Microsystems and now is supported by the Java Community Press. JDBC defines the objects and methods that all drivers must implement, as well as the objects and methods that drivers *may* optionally implement. PostgreSQL complies with the minimum JDBC definition, but cannot be said to be fully compliant until the remaining SQL92 features referenced above are implemented. Nevertheless, a working JDBC driver is available and is used in many production sites today.

Features and Functions of Version 7.0

Indexes

- Multiple Index Types (B-Tree, R-Tree, and Hash)
  PostgreSQL supports several index types: B-Tree, R-Tree, and Hash. The index type is user-specified allowing the user to pick the most efficient index for each application.
- Unique Indexes
  Unique Indices may be declared which prevent multiple records from having the same key. This ensures data consistency and eliminates the chance of having duplicate records.

- Multi-column Indexes

  PostgreSQL supports concatenated (or compound) keys (indices which consist of multiple columns such as LastName+FirstName+MiddleInitial). Although this functionality is beneficial in many applications, it is particularly useful in mining data from data warehouses.
- Clustered Indexes

  Each table in a PostgreSQL database can support a clustered index. This high-powered index physically sorts the data in the same sequence as specified by the index. A clustered index allows for the fastest possible data retrieval time.

DataTypes

Datatypes describe the content, capacity, and format of the data stored within a column in a database table. In building a table, choosing the appropriate datatypes can greatly increase the efficiency of the resulting database. PostgreSQL supports many more datatypes than are required by the 1992 ANSI SQL Standard. These include Character, Numeric, Data/Time, Logical, Geometric, and Network data types.

PostgreSQL Type	**SQL92 or SQL3 Type**	**Description**
Bool	Boolean	logical boolean (true/false)
Box		rectangular box in 2D plane
char(n)	character(n)	fixed-length character string
Cidr		IP version 4 network or host address
Circle		circle in 2D plane
Date	Date	calendar date without time of day
Decimal	decimal(p,s)	exact numeric for p <= 9, s = 0
float4	float(*p*), *p* < 7	floating-point number with precision *p*
float8	float(*p*), 7 <= *p* < 16	floating-point number with precision *p*
Inet		IP version 4 network or host address
int2	Smallint	signed two-byte integer
int4	int, integer	signed 4-byte integer
int8		signed 8-byte integer
Interval	Interval	general-use time span
large object		binary large object
Line		infinite line in 2D plane
Lseg		line segment in 2D plane
Money	decimal(9,2)	US-style currency
Numeric	numeric(p,s)	exact numeric for p == 9, s = 0

PostgreSQL Type	SQL92 or SQL3 Type	Description
Path		open and closed geometric path in 2D plane
Point		geometric point in 2D plane
Polygon		closed geometric path in 2D plane
Serial		unique id for indexing and cross-reference
Time	Time	time of day
Timetz	time with time zone	time of day, including time zone
Timestamp	timestamp with time zone	date/time
varchar(n)	character varying(n)	variable-length character string

Data Integrity

- Row-level locking

  By locking data at the row level, rather than at the page level (which may lock many rows), PostgreSQL reduces the contention between users for the same data.

- Transaction Logging

  In addition to being written to the target database, modifying transactions (Inserts, Deletes, and Updates) are also written to a log file. This duplication allows transactions to be recovered (repeated) following an abnormal shutdown of the database. Logging thus reduces the possibility of losing data in the event of a power interruption or hardware failure.

- Commit/Rollback

  Certain database functions require the grouping of transactions such that if any part of a function fails, the entire function fails. An example would be the transfer of money from one account to another; if the deposit (insert) fails, the previous withdrawal (update) needs to be backed out as if it never happened. PostgreSQL supports this concept with the implementation of Commit and Rollback. Transactions can be isolated in such a way that they can all be rolled back in the event of a failure of any of the other isolated transactions.

- Checkpoints

  A database achieves a great deal of its speed by caching individual transactions in high-speed Random Access Memory (RAM) and later writing them, en masse, to disk. Although caching transactions to RAM improves speed, it is risky in that RAM is not persistent memory and its contents can be lost in the event of a power outage or hardware failure. A "Checkpoint" forces all completed transactions to be written from RAM to disk, which is a persistent (safe) storage device. PostgreSQL supports checkpointing which helps to ensure the integrity of the data within the database.

- Triggers
  Triggers are user-written procedures which can be configured to fire off in the event of a transaction against a table. Typically triggers are used to ensure referential integrity (consistency) between tables in a database.
- Constraints
  In addition to referential integrity as enforced through the use of triggers, PostgreSQL also supports Declarative Referential Integrity (DRI) as enforced by constraints placed on a table when it is created. This ensures that every child record has a parent record in another table and prevents orphaned records with no relationship to data anywhere else in the database.
- On-line Backup
  PostgreSQL allows for on-line, or "hot" backups. Hot backups can be run without users having to log out of the database. This allows for archiving a database without any disruption to those using the database.

Functions

In order to increase the productivity of database programmers, PostgreSQL has placed often-used snippets of code into "functions" which can be called by the programmer. The most often used of these functions are referred to as "aggregates" because they operate on a number of rows simultaneously. These aggregates include Count(), Sum(), Max(), Min(), and Avg(); they are used to count the number of rows in a table, to find the sum of the numbers in a column, and to find the maximum, minimum, and average values of a column, respectively.

In addition to these aggregates, PostgreSQL also supports numerous other functions including Mathematical, String Manipulation, Date/Time Conversion, Formatting, and Geometric functions.

Even more flexibility is offered to the database programmer through PostgreSQL's support of User-Defined Functions. This capability allows a developer to encapsulate the code required to perform user-specific tasks. Once incorporated into a function, these routines can be easily called to perform work.

Platforms

Operating Systems

PostgreSQL is quite portable. It supports numerous operating systems, both open and proprietary. These platforms include:

BSDI	DG-UX	FreeBSD	HP/UX	IBM AIX
Linux Alpha	Linux Sparc	Linux x86	NetBSD	Nextstep
OSF/1 Alpha Digital Unix	SGI Irix 5.3	Solaris Sparc	Solaris x86	SunOS 4 (Sparc)
System V R4.2	Ultrix 4	Windows 9x (client only)	Windows NT	

Programming Languages

In addition, PostgreSQL supports many Programming Languages including:

C, C++, and Embedded C	Java
Perl 4 and Perl 5 (OO)	PHP
Python	Tcl

Other languages and development environments may access the database through its support of the Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC) specifications.

<u>Performance</u>

Benchmarks

Out of a need to be able to compare different databases, running under different operating systems, and on different hardware platforms, over time the industry has embraced several benchmark tests which can level the playing field – allowing databases to be compared “apples to apples”. Two of the more prominent tests used today to evaluate database performance are the AS3AP and the TPC-C.

- <u>ANSI SQL Standard Scalable and Portable (AS3AP)</u>
  This database benchmark was designed with several goals in mind: to be scalable enough that it can be used on both large and small platforms (which are comprised of the hardware, database, and operating system); to indicate an “equivalent database size” which is the maximum database size capable of running this benchmark for the indicated platform in under 12 hours; and to determine a “cost per megabyte” for the “equivalent database”. The transactions performed by this benchmark give a good assessment of the **raw transaction-processing power** of the platform and its **ability to scale**.

- Transaction Processing Performance Council (TPC)
  Probably the most accepted Database Benchmark in the industry is the TPC-C. TPC-C denotes Version C of the benchmark designed by the TPC. Unlike some of the other benchmarks which merely judge processing power (speed), Version C of the TPC benchmark is meant to emulate the on-line transaction processing (OLTP) typical in a real-world business environment. In particular, it mimics an order-entry system. Like a real-world order-entry system, the transactions processed in this benchmark mimic the management, selling, and distribution of a product or service typical in any industry. This benchmark results in two metrics: the first is the total number of transactions the database accomplished in a minute (a measure of throughput); the second metric is the number of transactions per minute divided by the Total Cost of Ownership (TCO) of the database, operating system, hardware, maintenance, and licenses for a total of five years. This metric is thus a measure of price performance (efficiency).

Testing

Currently Great Bridge is in the process of running the AS3AP and TPC-C benchmarks on PostgreSQL. These benchmarks are being run under both proprietary and open operating systems. Though the full results are not yet available, the initial findings are very impressive. From a price performance perspective, the results of running an open-source database on an open-source operating system (both of which serve to reduce the platform's Total Cost of Ownership) are even more compelling.

Following the completion and certification of these benchmarks by outside third parties, their results will be published by Great Bridge in a forthcoming whitepaper.

Conclusion

PostgreSQL is the most advanced open source database server available. It is a testament to the genius of open-source that even such a complex product as an Object-Relational Database Management System can be effectively developed by a global community of otherwise unrelated programmers. Great Bridge believes that PostgreSQL 7.0 is an important milestone in the development of open source software, and we look forward to keeping the momentum behind this exciting product going in succeeding releases.