Great Bridge PostgreSQL

Open Source Object-Relational Database: Introduction & Overview

White Paper
December 2000

Table of Contents

Introduction

Great Bridge PostgreSQL is a feature-rich open source database that is capable of handling the most challenging demands of today's e-business applications. It is a high-powered object-relational database that offers the stability, scalability and reliability required for mission critical applications. This white paper provides an introduction to Great Bridge PostgreSQL and summarizes its key features and functionality.

The History of PostgreSQL

Great Bridge PostgreSQL is an enhanced distribution of the PostgreSQL 7.0.3 open source database. 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—"post Ingres" (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 formerly served as 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 the PostgreSQL technology. Today, the PostgreSQL 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 7.0.3, has matured into a product capable of significant transaction throughput, complex queries, commercial-grade SQL support, and complex data types.

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 PostgreSQL 7.0.3

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*

PostgreSQL Type	SQL92 or SQL3 Type	Description
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
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

- Multi-Version Concurrency Control

  With better-than-row-level locking, 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. Great Bridge PostgreSQL is currently supported only on the Linux operating system. It is supported on:

- Red Hat Linux 6.1, 6.2 and 7.0
- TurboLinux 6.0
- SuSE Linux 6.4 and 7.0
- Caldera OpenLinux eServer 2.3
- Linux-Mandrake 7.2

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.

Performance

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.

- ANSI SQL Standard Scalable and Portable (AS3AP)
  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**.

- <u>Transaction Processing Performance Council (TPC)</u>
  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

In August 2000 Great Bridge funded the completion of the AS3AP and TPC-C benchmarks comparing PostgreSQL against its leading proprietary and open source database competitors. The results were conclusive: PostgreSQL is the most powerful and scalable open source database available, and outperforms the leading proprietary players in terms of raw transaction power.

On the ANSI SQL Standard Scalable And Portable (AS3AP) benchmark, PostgreSQL 7.0 performed an average of four to five times faster than every other database tested, including two major proprietary DBMS packages, the MySQL open source database, and Interbase, a formerly proprietary product which was recently made open source by Inprise/Borland. (See Exhibit 1) In the tested configuration, Postgres peaked at 1127.8 transactions per second with five users, and still processed at a steady rate of 1070.4 with 100 users. The proprietary leader also performed consistently, with a high of 314.15 transactions per second with eight users, which fell slightly to 288.37 transactions per second with 100 users. The other leading proprietary database also demonstrated consistency, running at 200.21 transactions per second with six users and 197.4 with 100.

The other databases tested against the AS3AP benchmarks, open source competitors MySQL 3.22 and Interbase 6.0, demonstrated some speed with a low number of users but a distinct lack of scalability. MySQL reached a peak of 803.48 with two users, but its performance fell precipitously under the stress of additional users to a rate of 117.87 transactions per second with 100 users. Similarly, Interbase reached 424 transactions per second with four users, but its performance declined steadily with additional users, dropping off to 146.86 transactions per second with 100 users.

In the Transaction Processing Council's TPC-C test, PostgreSQL 7.0 consistently matched the performance of the two leading proprietary database applications. (See Exhibit 2) The two industry leaders cannot be mentioned by name because their restrictive licensing agreements prohibit anyone who buys their products from publishing their company names in benchmark testing results without the companies' prior approval.

Neither Interbase nor MySQL could be tested for TPC-C benchmarks. MySQL could not run the test because the application is not adequately compliant with minimal ANSI SQL standards set in 1992. Interbase 6.0, recently released as open source, does not have a stable ODBC driver yet; while the version 5 ODBC driver for the AS3AP tests was able to be adapted, the TPC-C test would not run.

The benchmark tests were conducted by Xperts Inc. of Richmond, Virginia, an independent technology solutions company, using Quest Software's *Benchmark Factory* application. Xperts ran the benchmark tests on Compaq Proliant ML350 servers with 512 mb of RAM and two 18.2 Gb hard disks, equipped with Intel Pentium III processors and Red Hat Linux 6.1 and Windows NT operating systems. The company ensured the tests' consistency by using the same computers for each test, with each product connecting to the tests through its own preferred ODBC driver. While Benchmark Factory does provide native drivers for some commercial databases, using each product's own ODBC ensured the most valid "apples to apples" comparison.

License

Great Bridge PostgreSQL is distributed under a "Berkeley-style" license similar to those of the various BSD operating systems, fully accepted under the “Open Source Definition” as established by the Open Source Initiative, an open source community group that monitors and tracks open source licensing. This license dates back to the software's history at the University of California at Berkeley, and we believe it to be excellent license that encourages the widespread adoption and use of open source software. Great Bridge PostgreSQL is available without license fee, allowing for free use, redistribution and modification of the software.

PostgreSQL is not public domain software. It is copyrighted by the University of California but may be used according to the licensing terms of the copyright below:
POSTGRES95 Data Base Management System (formerly known as Postgres, then as Postgres95).

Copyright (c) 1994-8 Regents of the University of California

Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS

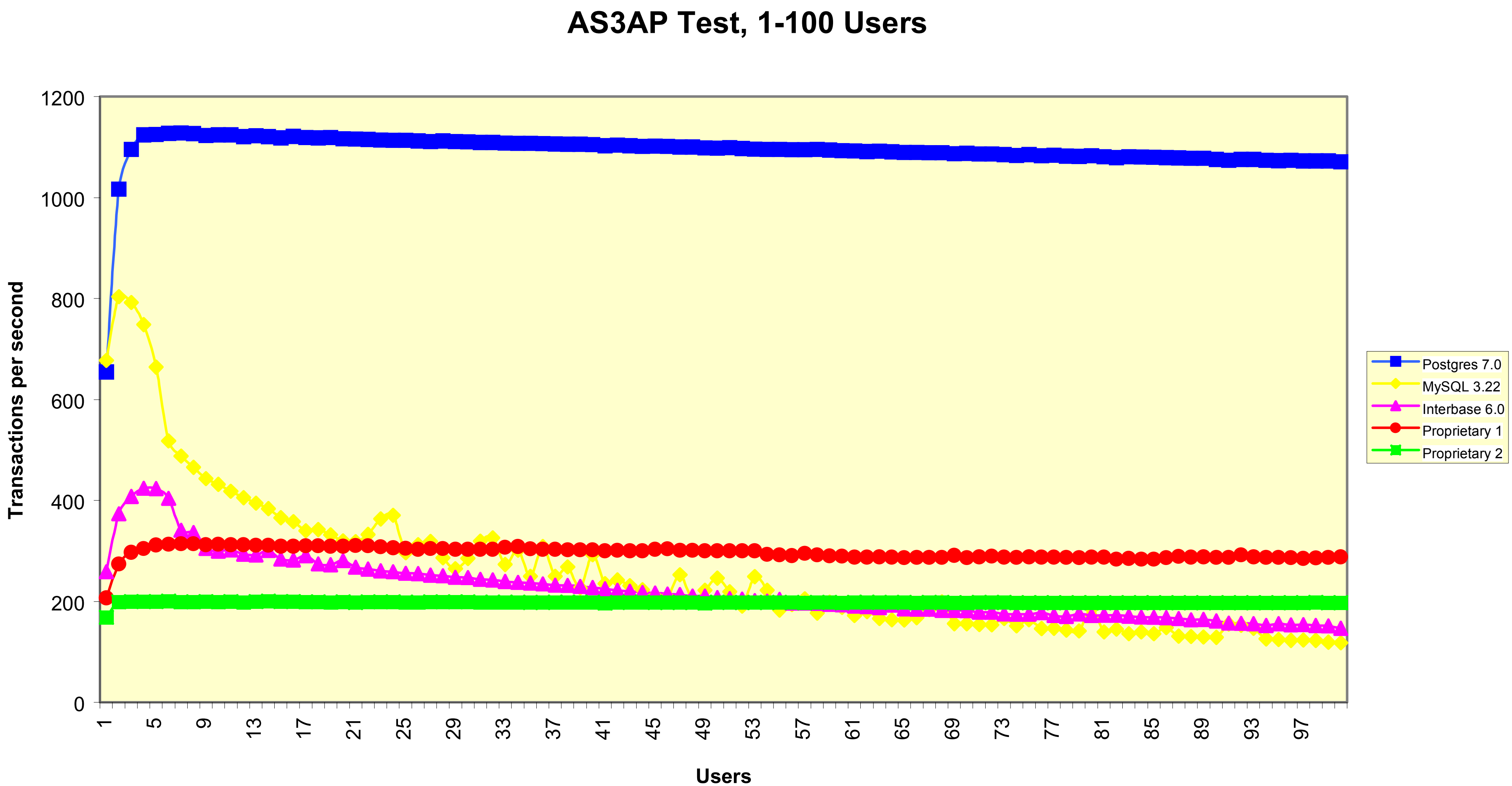
SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

Conclusion

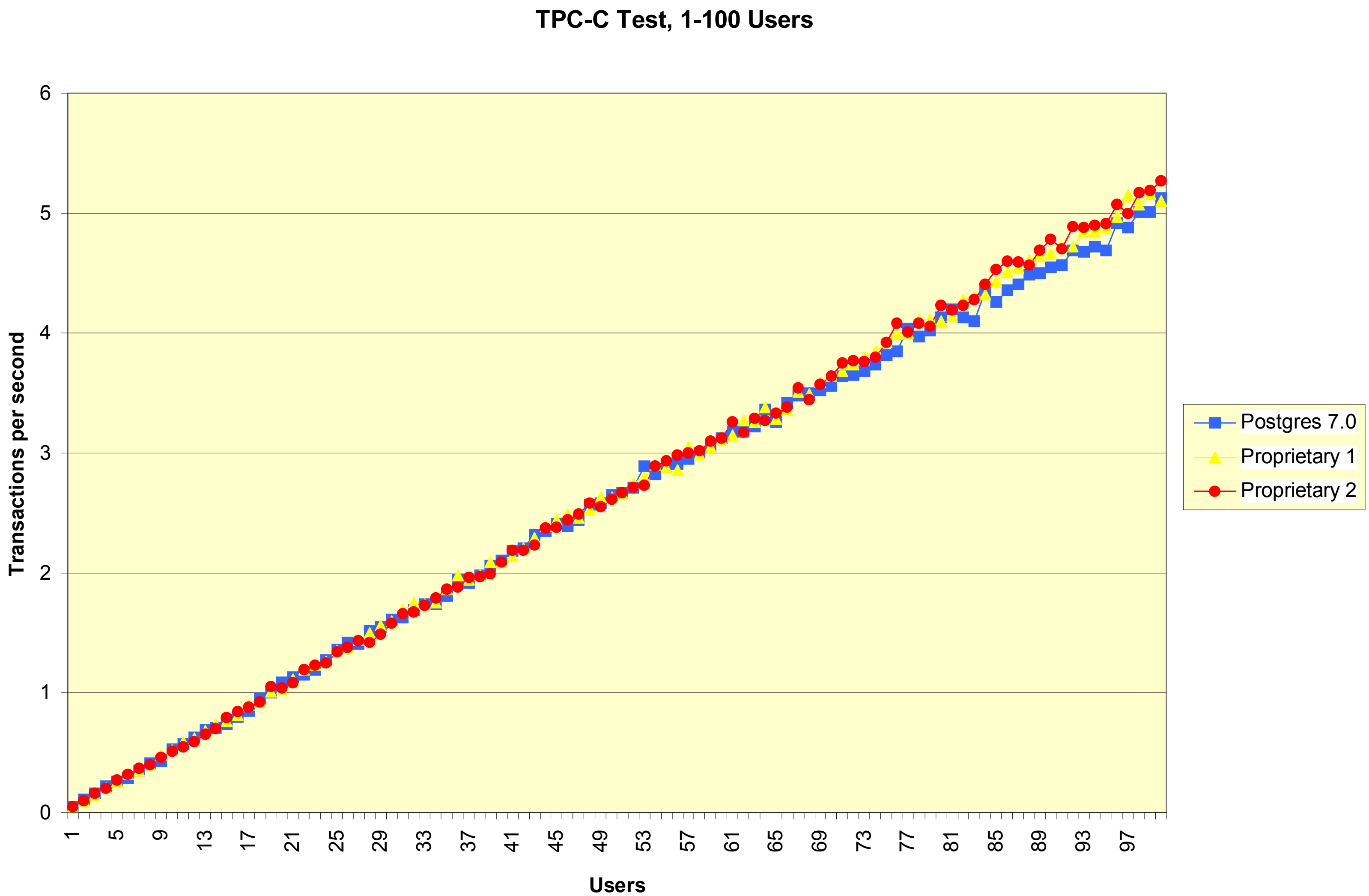
Great Bridge PostgreSQL is the first-ever commercial distribution of PostgreSQL, the most advanced open source database server available. It is a testament to the genius of open-source development that even such a complex product as an Object-Relational Database Management System can be effectively developed by a global community of otherwise unrelated programmers. Great Bridge provides a broad spectrum of professional services and support for Great Bridge PostgreSQL, including technical support packages, developer-to-developer support, VAR partner programs and consulting services.

EXHIBIT 1



AS3AP Test, 1-100 Users
Transactions per second
1200
1000
800
600
400
200
0
1
5
9
13
17
21
25
29
33
37
41
45
49
53
57
61
65
69
73
77
81
85
89
93
97
Users
Postgres 7.0
MySQL 3.22
Interbase 6.0
Proprietary 1
Proprietary 2

EXHIBIT 2


TPC-C Test, 1-100 Users
Transactions per second
0
1
2
3
4
5
6
Users
1
5
9
13
17
21
25
29
33
37
41
45
49
53
57
61
65
69
73
77
81
85
89
93
97
Postgres 7.0
Proprietary 1
Proprietary 2