PostgreSQL

From Wikipedia, the free encyclopedia

PostgreSQL, often simply **Postgres**, is an object-relational database (ORDBMS) – i.e. an RDBMS, with additional (optional use) "object" features – with an emphasis on extensibility and standards compliance. As a database server, its primary functions are to store data securely and return that data in response to requests from other software applications. It can handle workloads ranging from small single-machine applications to large Internet-facing applications (or for data warehousing) with many concurrent users; on macOS Server, PostgreSQL is the default database;[11][12][13] and it is also available for Microsoft Windows and Linux (supplied in most distributions).

PostgreSQL is ACID-compliant and transactional. PostgreSQL has updatable views and materialized views, triggers, foreign keys; supports functions and stored procedures, and other expandability.[14]

PostgreSQL is developed by the PostgreSQL Global Development Group, a diverse group of many companies and individual contributors.[15] It is free and open-source software, released under the terms of the PostgreSQL License, a permissive free-software license.

Contents

PostgreSQL

Developer(s)	PostgreSQL Global Development Group
Initial release	July 8, 1996[1]
Stable release	9.6.2 / February 9, 2017[2]
Repository	git.postgresql.org/gitweb/?p=postgresql.git (http://git.postgresql.org/gitweb/?p=postgresql.git)
Written in	C (pgAdmin: wxWidgets[3])
Operating system	Cross-platform, i.e. most Unix-like operating systems and Windows
Type	ORDBMS
License	PostgreSQL License[4][5][6]
Website	postgresql.org (http://postgresql.org)

PostgreSQL License[7]

DFSG compatible	Yes[8][9]
FSF approved	Yes[10]
OSI approved	Yes[5]
GPL compatible	Yes
Copyleft	No
Linking from code with a different license	Yes
Website	PostgreSQL License (http://postgresql.org/about/licence)

Name

PostgreSQL's developers pronounce PostgreSQL as /ˈpoʊstɡrɛs ˌkjuː ˈɛl/.[16] It is abbreviated as *Postgres* because of ubiquitous support for the SQL Standard among most relational databases. The community considered changing the name back to Postgres; however, the PostgreSQL Core Team announced in 2007 that the product would continue to use the name PostgreSQL.[17] The name Postgres (Post Ingres) refers to the project's origins in that database which was developed at University of California, Berkeley.[18][19]

History

PostgreSQL evolved from the Ingres project at the University of California, Berkeley. In 1982, the leader of the Ingres team, Michael Stonebraker, left Berkeley to make a proprietary version of Ingres.[18] He returned to Berkeley in 1985, and started a post-Ingres project to address the problems with contemporary database systems that had become increasingly clear during the early 1980s. The new project, POSTGRES, aimed to add the fewest features needed to completely support types.[20] These features included the ability to define types and to fully describe relationships – something used widely before but maintained entirely by the user. In POSTGRES, the database "understood" relationships, and could retrieve information in related tables in a natural way using *rules*. POSTGRES used many of the ideas of Ingres, but not its code.[21]

Starting in 1986, the POSTGRES team published a number of papers describing the basis of the system, and by 1987 had a prototype version shown at the 1988 ACM SIGMOD Conference. The team released version 1 to a small number of users in June 1989, then version 2 with a re-written rules system in June 1990. Version 3, released in 1991, again re-wrote the rules system, and added support for multiple storage managers and an improved query engine. By 1993, the great number of users began to overwhelm the project with requests for support and features. After releasing version 4.2[22] on June 30, 1994 – primarily a cleanup – the project ended. Berkeley had released POSTGRES under an MIT-style license, which enabled other developers to use the code for any use. At the time, POSTGRES used an Ingres-influenced POSTQUEL query language interpreter, which could be interactively used with a console application named `monitor`.

In 1994, Berkeley graduate students Andrew Yu and Jolly Chen replaced the POSTQUEL query language interpreter with one for the SQL query language, creating Postgres95. The front-end program `monitor` was also replaced by `psql`. Yu and Chen announced the first version (0.01) to beta testers on May 5, 1995. Version 1.0 of Postgres95 was announced on September 5, 1995, with a more liberal license that enabled the software to be freely modifiable for any purpose.

In 1996, the project was renamed to PostgreSQL to reflect its support for SQL. The online presence at the website PostgreSQL.org began on October 22, 1996.[23] The first PostgreSQL release formed version 6.0 on January 29, 1997. Since then a group of developers and volunteers around the world have maintained the software as The PostgreSQL Global Development Group.

The PostgreSQL project continues to make major releases (approximately annually) and minor "bugfix" releases, all available under its free and open-source software PostgreSQL License. Code comes from contributions from proprietary vendors, support companies, and open-source programmers at large.

Multiversion concurrency control (MVCC)

PostgreSQL manages concurrency through a system known as multiversion concurrency control (MVCC), which gives each transaction a "snapshot" of the database, allowing changes to be made without being visible to other transactions until the changes are committed. This largely eliminates the need for read locks, and ensures the database maintains the ACID (atomicity, consistency, isolation, durability) principles in an efficient manner. PostgreSQL offers three levels of transaction isolation: Read Committed, Repeatable Read and Serializable. Because PostgreSQL is immune to dirty reads, requesting a Read Uncommitted transaction isolation level provides read committed instead. PostgreSQL supports full serializability via the serializable snapshot isolation (SSI) technique.[24]

Storage and replication

Replication

PostgreSQL includes built-in binary replication based on shipping the changes (write-ahead logs) to replica nodes asynchronously, with the ability to run read-only queries against these replicated nodes. This allows splitting read traffic among multiple nodes efficiently. Earlier replication software that allowed similar read scaling normally relied on adding replication triggers to the master, introducing additional load onto it.

PostgreSQL also includes built-in synchronous replication[25] that ensures that, for each write transaction, the master waits until at least one replica node has written the data to its transaction log. Unlike other database systems, the durability of a transaction (whether it is asynchronous or synchronous) can be specified per-database, per-user, per-session or even per-transaction. This can be useful for work loads that do not require such guarantees, and may not be wanted for all data as it will have some negative effect on performance due to the requirement of the confirmation of the transaction reaching the synchronous standby.

There can be a mixture of synchronous and asynchronous standby servers. A list of synchronous standby servers can be specified in the configuration which determines which servers are candidates for synchronous replication. The first in the list which is currently connected and actively streaming is the one that will be used as the current synchronous server. When this fails, it falls to the next in line.

Synchronous multi-master replication is currently not included in the PostgreSQL core. Postgres-XC which is based on PostgreSQL provides scalable synchronous multi-master replication,[26] available in version 1.2.1 (April 2015 version) is licensed under the same license as PostgreSQL. A similar project is called Postgres-XL and is available under the Mozilla Public License.[27] Postgres-R is yet another older fork.[28] Bi-Directional Replication (BDR) is an asynchronous multi-master replication system for PostgreSQL.[29]

The community has also written some tools to make managing replication clusters easier, such as repmgr.

There are also several asynchronous trigger-based replication packages for PostgreSQL. These remain useful even after introduction of the expanded core capabilities, for situations where binary replication of an entire database cluster is not the appropriate approach:

- Slony-I
- Londiste, part of SkyTools (developed by Skype)
- Bucardo multi-master replication (developed by Backcountry.com)[30]
- SymmetricDS multi-master, multi-tier replication

Indexes

PostgreSQL includes built-in support for regular B-tree and hash indexes, and four index access methods: generalized search trees (GiST), generalized inverted indexes (GIN), Space-Partitioned GiST (SP-GiST)[31] and Block Range Indexes (BRIN). Hash indexes are implemented, but discouraged because they cannot be recovered after a crash or power loss. In addition, user-defined index methods can be created, although this is quite an involved process. Indexes in PostgreSQL also support the following features:

- Expression indexes can be created with an index of the result of an expression or function, instead of simply the value of a column.
- Partial indexes, which only index part of a table, can be created by adding a `WHERE` clause to the end of the `CREATE INDEX` statement. This allows a smaller index to be created.
- The planner is capable of using multiple indexes together to satisfy complex queries, using temporary in-memory bitmap index operations (useful in data warehousing applications for joining a large fact table to smaller dimension tables such as those arranged in a star schema).
- *k*-nearest neighbors (*k*-NN) indexing (also referred to KNN-GiST[32]) provides efficient searching of "closest values" to that specified, useful to finding similar words, or close objects or locations with geospatial data. This is achieved without exhaustive matching of values.
- In PostgreSQL 9.2 and later, index-only scans often allow the system to fetch data from indexes without ever having to access the main table.
- PostgreSQL 9.5 introduced Block Range Indexes (BRIN).

Schemas

In PostgreSQL, a schema holds all objects (with the exception of roles and tablespaces). Schemas effectively act like namespaces, allowing objects of the same name to co-exist in the same database. By default, newly created databases have a schema called "public", but any additional schemas can be added, and the public schema isn't mandatory.

A "search_path" setting determines the order in which PostgreSQL checks schemas for unqualified objects (those without a prefixed schema). By default, it is set to "$user, public" ($user refers to the currently connected database user). This default can be set on a database or role level, but as it is a session parameter, it can be freely changed (even multiple times) during a client session, affecting that session only.

Non-existent schemas listed in search_path are silently skipped during objects lookup.

New objects are created in whichever valid schema (one that presently exists) appears first in the search_path.

Data types

A wide variety of native data types are supported, including:

- Boolean
- Arbitrary precision numerics
- Character (text, varchar, char)
- Binary
- Date/time (timestamp/time with/without timezone, date, interval)
- Money
- Enum
- Bit strings
- Text search type
- Composite
- HStore (an extension enabled key-value store within PostgreSQL)
- Arrays (variable length and can be of any data type, including text and composite types) up to 1 GB in total storage size
- Geometric primitives
- IPv4 and IPv6 addresses
- CIDR blocks and MAC addresses

- XML supporting XPath queries
- UUID
- JSON (since version 9.2), and a faster binary JSONB (since version 9.4; not the same as BSON[33])

In addition, users can create their own data types which can usually be made fully indexable via PostgreSQL's indexing infrastructures – GiST, GIN, SP-GiST. Examples of these include the geographic information system (GIS) data types from the PostGIS project for PostgreSQL.

There is also a data type called a "domain", which is the same as any other data type but with optional constraints defined by the creator of that domain. This means any data entered into a column using the domain will have to conform to whichever constraints were defined as part of the domain.

Starting with PostgreSQL 9.2, a data type that represents a range of data can be used which are called range types. These can be discrete ranges (e.g. all integer values 1 to 10) or continuous ranges (e.g. any point in time between 10:00 am and 11:00 am). The built-in range types available include ranges of integers, big integers, decimal numbers, time stamps (with and without time zone) and dates.

Custom range types can be created to make new types of ranges available, such as IP address ranges using the inet type as a base, or float ranges using the float data type as a base. Range types support inclusive and exclusive range boundaries using the [] and () characters respectively. (e.g. '[4,9)' represents all integers starting from and including 4 up to but not including 9.) Range types are also compatible with existing operators used to check for overlap, containment, right of etc.

User-defined objects

New types of almost all objects inside the database can be created, including:

- Casts
- Conversions
- Data types
- Domains
- Functions, including aggregate functions and window functions
- Indexes including custom indexes for custom types
- Operators (existing ones can be overloaded)
- Procedural languages

Inheritance

Tables can be set to inherit their characteristics from a "parent" table. Data in child tables will appear to exist in the parent tables, unless data is selected from the parent table using the ONLY keyword, i.e. `SELECT * FROM ONLY parent_table;`. Adding a column in the parent table will cause that column to appear in the child table.

Inheritance can be used to implement table partitioning, using either triggers or rules to direct inserts to the parent table into the proper child tables.

As of 2010, this feature is not fully supported yet – in particular, table constraints are not currently inheritable. All check constraints and not-null constraints on a parent table are automatically inherited by its children. Other types of constraints (unique, primary key, and foreign key constraints) are not inherited.

Inheritance provides a way to map the features of generalization hierarchies depicted in entity relationship diagrams (ERDs) directly into the PostgreSQL database.

Other storage features

- Referential integrity constraints including foreign key constraints, column constraints, and row checks
- Binary and textual large-object storage

- Tablespaces
- Per-column collation
- Online backup
- Point-in-time recovery, implemented using write-ahead logging
- In-place upgrades with pg_upgrade for less downtime (supports upgrades from 8.3.x and later)

Control and connectivity

Foreign data wrappers

PostgreSQL can link to other systems to retrieve data via foreign data wrappers (FDWs).[34] These can take the form of any data source, such as a file system, another RDBMS, or a web service. This means that regular database queries can use these data sources like regular tables, and even join multiple data-sources together.

Interfaces

PostgreSQL has several interfaces available and is also widely supported among programming language libraries. Built-in interfaces include libpq (PostgreSQL's official C application interface) and ECPG (an embedded C system). External interfaces include:

- libpqxx: C++ interface
- PostgresDAC: PostgresDAC (for Embarcadero RadStudio/Delphi/CBuilder XE-XE3)
- DBD::Pg: Perl DBI driver
- JDBC: JDBC interface
- Lua: Lua interface
- Npgsql: .NET data provider
- ST-Links SpatialKit: Link Tool to ArcGIS
- PostgreSQL.jl: Julia interface
- node-postgres: Node.js interface
- pgoledb: OLEDB interface
- psqlODBC: ODBC interface
- psycopg2:[35] Python interface (also used by HTSQL)
- pgtclng: Tcl interface
- pyODBC: Python library
- php5-pgsql: PHP driver based on libpq
- postmodern: A Common Lisp interface
- pq: A pure Go PostgreSQL driver for the Go database/sql package. The driver passes the compatibility test suite.[36]
- dpq: D interface to libpq
- epgsql: Erlang interface

Procedural languages

Procedural languages allow developers to extend the database with custom subroutines (functions), often called *stored procedures*. These functions can be used to build triggers (functions invoked upon modification of certain data) and custom aggregate functions. Procedural languages can also be invoked without defining a function, using the "DO" command at SQL level.

Languages are divided into two groups: "Safe" languages are sandboxed and can be safely used by any user. Procedures written in "unsafe" languages can only be created by superusers, because they allow bypassing the database's security restrictions, but can also access sources external to the database. Some languages like Perl provide both safe and unsafe versions.

PostgreSQL has built-in support for three procedural languages:

- Plain SQL (safe). Simpler SQL functions can get expanded inline into the calling (SQL) query, which saves function call overhead and allows the query optimizer to "see inside" the function.
- PL/pgSQL (safe), which resembles Oracle's PL/SQL procedural language and SQL/PSM.
- C (unsafe), which allows loading custom shared libraries into the database. Functions written in C offer the best performance, but bugs in code can crash and potentially corrupt the database. Most built-in functions are written in C.

In addition, PostgreSQL allows procedural languages to be loaded into the database through extensions. Three language extensions are included with PostgreSQL to support Perl, Python and Tcl. There are external projects to add support for many other languages,[37] including Java, JavaScript (PL/V8), R, Ruby, and others.

Triggers

Triggers are events triggered by the action of SQL DML statements. For example, an INSERT statement might activate a trigger that checks if the values of the statement are valid. Most triggers are only activated by either INSERT or UPDATE statements.

Triggers are fully supported and can be attached to tables. Triggers can be per-column and conditional, in that UPDATE triggers can target specific columns of a table, and triggers can be told to execute under a set of conditions as specified in the trigger's WHERE clause. Triggers can be attached to views by using the INSTEAD OF condition. Multiple triggers are fired in alphabetical order. In addition to calling functions written in the native PL/pgSQL, triggers can also invoke functions written in other languages like PL/Python or PL/Perl.

Asynchronous notifications

PostgreSQL provides an asynchronous messaging system that is accessed through the NOTIFY, LISTEN and UNLISTEN commands. A session can issue a NOTIFY command, along with the user-specified channel and an optional payload, to mark a particular event occurring. Other sessions are able to detect these events by issuing a LISTEN command, which can listen to a particular channel. This functionality can be used for a wide variety of purposes, such as letting other sessions know when a table has updated or for separate applications to detect when a particular action has been performed. Such a system prevents the need for continuous polling by applications to see if anything has yet changed, and reducing unnecessary overhead. Notifications are fully transactional, in that messages are not sent until the transaction they were sent from is committed. This eliminates the problem of messages being sent for an action being performed which is then rolled back.

Many of the connectors for PostgreSQL provide support for this notification system (including libpq, JDBC, Npgsql, psycopg and node.js) so it can be used by external applications.

Rules

Rules allow the "query tree" of an incoming query to be rewritten. Rules, or more properly, "Query Re-Write Rules", are attached to a table/class and "Re-Write" the incoming DML (select, insert, update, and/or delete) into one or more queries that either replace the original DML statement or execute in addition to it. Query Re-Write occurs after DML statement parsing, but before query planning.

Other querying features

- Transactions
- Full text search
- Views
  - Materialized views[38]
  - Updateable views[39]
  - Recursive views[40]
- Inner, outer (full, left and right), and cross joins

- Sub-selects
  - Correlated sub-queries[41]
- Regular expressions[42]
- Common table expressions and writable common table expressions
- Encrypted connections via TLS (current versions do not use vulnerable SSL, even with that configuration option)[43]
- Domains
- Savepoints
- Two-phase commit
- TOAST (*The Oversized-Attribute Storage Technique*) is used to transparently store large table attributes (such as big MIME attachments or XML messages) in a separate area, with automatic compression.
- Embedded SQL is implemented using preprocessor. SQL code is first written embedded into C code. Then code is run through ECPG preprocessor, which replaces SQL with calls to code library. Then code can be compiled using a C compiler. Embedding works also with C++ but it does not recognize all C++ constructs.

Security

PostgreSQL manages its internal security on a per-role basis. A role is generally regarded to be a user (a role that can log in), or a group (a role of which other roles are members). Permissions can be granted or revoked on any object down to the column level, and can also allow/prevent the creation of new objects at the database, schema or table levels.

PostgreSQL's SECURITY LABEL feature (extension to SQL standards), allows for additional security; with a bundled loadable module that supports label-based mandatory access control (MAC) based on SELinux security policy.[44][45]

PostgreSQL natively supports a broad number of external authentication mechanisms, including:

- password (either MD5 or plain-text)
- GSSAPI
- SSPI
- Kerberos
- ident (maps O/S user-name as provided by an ident server to database user-name)
- peer (maps local user name to database user name)
- LDAP
  - Active Directory
- RADIUS
- certificate
- PAM

The GSSAPI, SSPI, Kerberos, peer, ident and certificate methods can also use a specified "map" file that lists which users matched by that authentication system are allowed to connect as a specific database user.

These methods are specified in the cluster's host-based authentication configuration file (`pg_hba.conf`), which determines what connections are allowed. This allows control over which user can connect to which database, where they can connect from (IP address/IP address range/domain socket), which authentication system will be enforced, and whether the connection must use TLS.

Add-ons

- Apache MADlib: an open source analytics library for PostgreSQL providing mathematical, statistical and machine-learning methods for structured and unstructured data[46]
- Ora2Pg: an OpenSource, GPL licensed Oracle and MySQL migration tool written in Perl

- MySQL migration wizard: included with EnterpriseDB's PostgreSQL installer (source code also available)[47]
- Performance Wizard: included with EnterpriseDB's PostgreSQL installer (source code also available)[47]
- pgRouting: extended PostGIS to provide geospatial routing functionality[48] (GNU GPL)
- PostGIS: a popular add-on which provides support for geographic objects (GNU GPL)
- Postgres Enterprise Manager: a non-free tool consisting of a service, multiple agents, and a GUI which provides remote monitoring, management, reporting, capacity planning and tuning[49]
- ST-Links SpatialKit: Extension for directly connecting to spatial databases[50]
- Citus: an extension to turn Postgres into a distributed system to support real-time workloads on large data sets

Benchmarks and performance

Many informal performance studies of PostgreSQL have been done.[51] Performance improvements aimed at improving scalability started heavily with version 8.1. Simple benchmarks between version 8.0 and version 8.4 showed that the latter was more than 10 times faster on read-only workloads and at least 7.5 times faster on both read and write workloads.[52]

The first industry-standard and peer-validated benchmark was completed in June 2007, using the Sun Java System Application Server (proprietary version of GlassFish) 9.0 Platform Edition, UltraSPARC T1-based Sun Fire server and PostgreSQL 8.2.[53] This result of 778.14 SPECjAppServer2004 JOPS@Standard compares favourably with the 874 JOPS@Standard with Oracle 10 on an Itanium-based HP-UX system.[51]

In August 2007, Sun submitted an improved benchmark score of 813.73 SPECjAppServer2004 JOPS@Standard. With the system under test at a reduced price, the price/performance improved from $84.98/JOPS to $70.57/JOPS.[54]

The default configuration of PostgreSQL uses only a small amount of dedicated memory for performance-critical purposes such as caching database blocks and sorting. This limitation is primarily because older operating systems required kernel changes to allow allocating large blocks of shared memory.[55] PostgreSQL.org provides advice on basic recommended performance practice in a wiki.[56]

In April 2012, Robert Haas of EnterpriseDB demonstrated PostgreSQL 9.2's linear CPU scalability using a server with 64 cores.[57]

Matloob Khushi performed benchmarking between Postgresql 9.0 and MySQL 5.6.15 for their ability to process genomic data. In his performance analysis he found that PostgreSQL extracts overlapping genomic regions eight times faster than MySQL using two datasets of 80,000 each forming random human DNA regions. Insertion and data uploads in PostgreSQL were also better, although general searching capability of both databases was almost equivalent.[58]

Platforms

PostgreSQL is available for the following operating systems: Linux (all recent distributions), Windows (Windows 2000 SP4 and later; compilable by e.g. Visual Studio, now with up to most recent 2015 version), FreeBSD, OpenBSD, NetBSD, OS X (macOS),[13] AIX, HP-UX, Solaris, and UnixWare; and not officially tested: DragonFly BSD, BSD/OS, IRIX, OpenIndiana,[59] OpenSolaris, OpenServer, and Tru64 Unix. Most other Unix-like systems could also work; most modern do support.

PostgreSQL works on any of the following instruction set architectures: x86 and x86-64 on Windows and other operating systems; these are supported on other than Windows: IA-64 Itanium (external support for HP-UX), PowerPC, PowerPC 64, S/390, S/390x, SPARC, SPARC 64, ARMv8-A (64-bit)[60] and older ARM (32-bit,

including older such as ARMv6 in Raspberry Pi[61]), MIPS, MIPSel, and PA-RISC. It is also known to work on Alpha (dropped in 9.5), M68k, M32R, NS32k, and VAX. In addition to these, it is possible to build PostgreSQL for an unsupported CPU by disabling spinlocks.[62]

Database administration

Open source front-ends and tools for administering PostgreSQL include:

psql
: The primary front-end for PostgreSQL is the `psql` command-line program, which can be used to enter SQL queries directly, or execute them from a file. In addition, psql provides a number of meta-commands and various shell-like features to facilitate writing scripts and automating a wide variety of tasks; for example tab completion of object names and SQL syntax.

pgAdmin
: The pgAdmin package is a free and open source graphical user interface administration tool for PostgreSQL, which is supported on many computer platforms.[63] The program is available in more than a dozen languages. The first prototype, named pgManager, was written for PostgreSQL 6.3.2 from 1998, and rewritten and released as pgAdmin under the GNU General Public License (GPL) in later months. The second incarnation (named pgAdmin II) was a complete rewrite, first released on January 16, 2002. The third version, pgAdmin III, was originally released under the Artistic License and then released under the same license as PostgreSQL. Unlike prior versions that were written in Visual Basic, pgAdmin III is written in C++, using the wxWidgets framework allowing it to run on most common operating systems. The query tool includes a scripting language called pgScript for supporting admin and development tasks. In December 2014, Dave Page, the pgAdmin project founder and primary developer,[64] announced that with the shift towards web-based models work has started on pgAdmin 4 with the aim of facilitating Cloud deployments.[65] In 2016, pgAdmin 4 was released.

phpPgAdmin
: phpPgAdmin is a web-based administration tool for PostgreSQL written in PHP and based on the popular phpMyAdmin interface originally written for MySQL administration.[66]

PostgreSQL Studio
: PostgreSQL Studio allows users to perform essential PostgreSQL database development tasks from a web-based console. PostgreSQL Studio allows users to work with cloud databases without the need to open firewalls.[67]

TeamPostgreSQL
: AJAX/JavaScript-driven web interface for PostgreSQL. Allows browsing, maintaining and creating data and database objects via a web browser. The interface offers tabbed SQL editor with auto-completion, row-editing widgets, click-through foreign key navigation between rows and tables, 'favorites' management for commonly used scripts, among other features. Supports SSH for both the web interface and the database connections. Installers are available for Windows, Mac and Linux, as well as a simple cross-platform archive that runs from a script.[68]

LibreOffice/OpenOffice.org Base
: LibreOffice/OpenOffice.org Base can be used as a front-end for PostgreSQL.[69][70]

pgBadger
: The pgBadger PostgreSQL log analyzer generates detailed reports from a PostgreSQL log file.[71]

A number of companies offer proprietary tools for PostgreSQL. They often consist of a universal core that is adapted for various specific database products. These tools mostly share the administration features with the open source tools but offer improvements in data modeling, importing, exporting or reporting.

Prominent users

Prominent organizations and products that use PostgreSQL as the primary database include:

- Yahoo! for web user behavioral analysis, storing two petabytes and purportedly the largest data warehouse, using a heavily modified version of PostgreSQL with an entirely different column-based storage engine and different query-processing layer. While in terms of performance, storage, and query the database bears little resemblance to PostgreSQL, the front-end maintains compatibility so that Yahoo can use many off-the-shelf tools already written to interact with PostgreSQL.[72][73]
- In 2009, the social-networking website MySpace used Aster Data Systems's nCluster database for data warehousing, which was built on unmodified PostgreSQL.[74][75]
- Geni.com uses PostgreSQL for their main genealogy database.[76]
- OpenStreetMap, a collaborative project to create a free editable map of the world.[77]
- Afilias, domain registries for .org, .info and others.[78]
- Sony Online multiplayer online games.[79]
- BASF, shopping platform for their agribusiness portal.[80]
- Reddit social news website.[81]
- Skype VoIP application, central business databases.[82]
- Sun xVM, Sun's virtualization and datacenter automation suite.[83]
- MusicBrainz, open online music encyclopedia.[84]
- The International Space Station – for collecting telemetry data in orbit and replicating it to the ground.[85]
- MyYearbook social-networking site.[86]
- Instagram, a mobile photo-sharing service.[87]
- Disqus, an online discussion and commenting service.[88]
- TripAdvisor, travel-information website of mostly user-generated content.[89]
- Yandex, a Russian internet company switched from Oracle to Postgres for its email offering.[90]
- AWS Redshift, a columnar OLAP system based on ParAccel's Postgres modifications.

Service implementations

Some major vendors offer PostgreSQL as software as a service:

- Citus Cloud, is a database-as-a-service provided by Citus Data. Built by the initial engineering team that built Heroku Postgres it focuses on horizontally scaled out Postgres.[91] It offers continuous monitoring, minutely backups, point-in-time-recovery, and the ability to elastically scale your database.
- Heroku, a platform as a service provider, has supported PostgreSQL since the start in 2007.[92] They offer value-add features like full database "roll-back" (ability to restore a database from any point in time),[93] which is based on WAL-E, open-source software developed by Heroku.[94]
- In January 2012, EnterpriseDB released a cloud version of both PostgreSQL and their own proprietary Postgres Plus Advanced Server with automated provisioning for failover, replication, load-balancing, and scaling. It runs on Amazon Web Services.[95]
- VMware has offered vFabric Postgres (also known as vPostgres[96]) for private clouds on vSphere since May 2012.[97]
- In November 2013, Amazon.com announced the addition of PostgreSQL to their Relational Database Service offering.[98][99]
- In November 2016, Amazon Web Services announced the addition of PostgreSQL compatibility to their cloud-native Amazon Aurora managed database offering.[100]

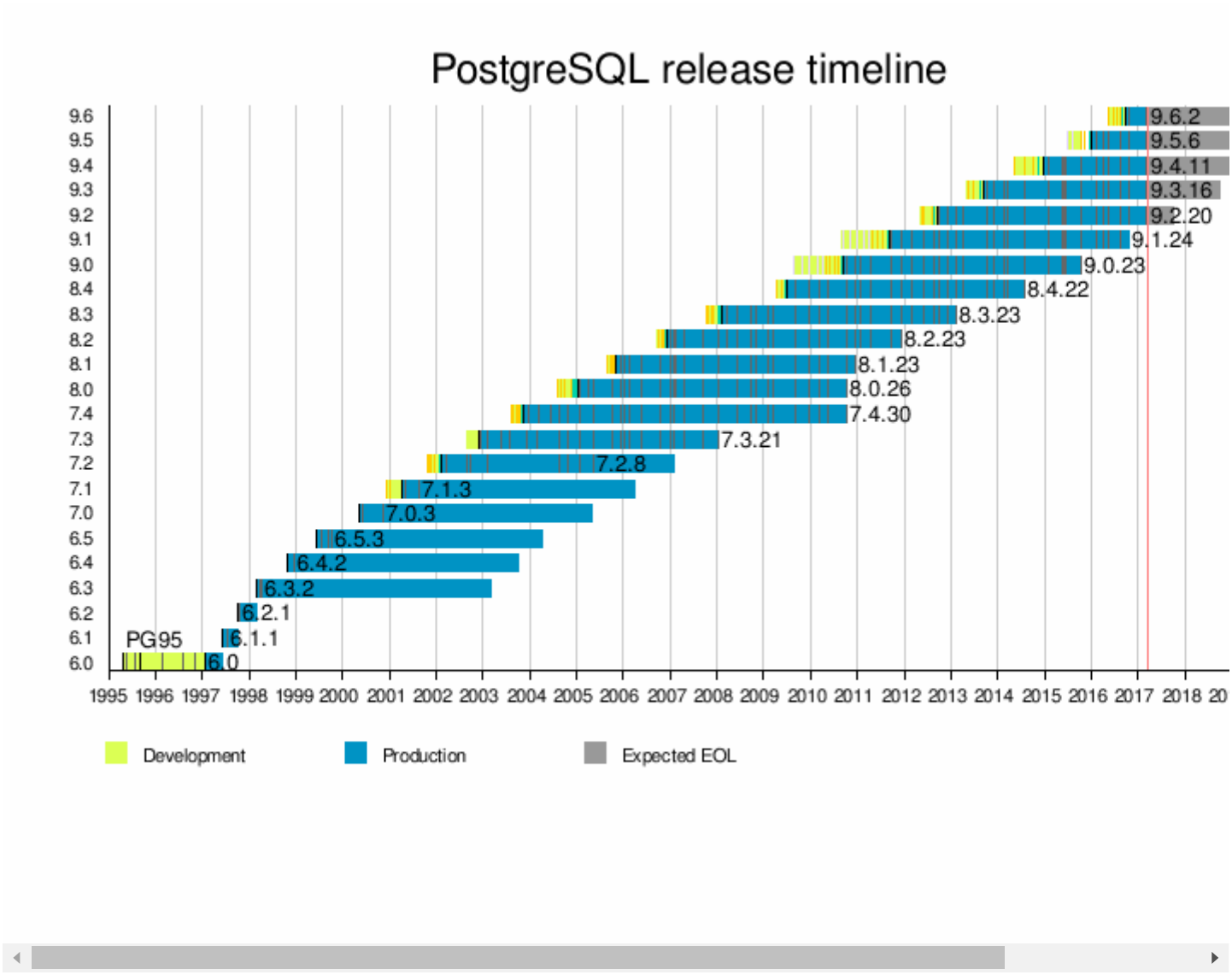
Release history

Release	First release	Latest minor version	Latest release	End of Life	Milestones
6.0	1997-01-29	–	N/A	N/A	First formal release of PostgreSQL, unique indexes, pg_dumpall utility, ident authentication
6.1	1997-06-08	6.1.1	1997-07-22	N/A	Multi-column indexes, sequences, money data type, GEQO (GEnetic Query Optimizer)
6.2	1997-10-02	6.2.1	1997-10-17	N/A	JDBC interface, triggers, server programming interface, constraints
6.3	1998-03-01	6.3.2	1998-04-07	2003-04	SQL92 subselect capability, PL/pgTCL
6.4	1998-10-30	6.4.2	1998-12-20	2003-10	VIEWs (then only read-only) and RULEs, PL/pgSQL
6.5	1999-06-09	6.5.3	1999-10-13	2004-06	MVCC, temporary tables, more SQL statement support (CASE, INTERSECT, and EXCEPT)
7.0	2000-05-08	7.0.3	2000-11-11	2004-05	Foreign keys, SQL92 syntax for joins
7.1	2001-04-13	7.1.3	2001-08-15	2006-04	Write-ahead log, outer joins
7.2	2002-02-04	7.2.8	2005-05-09	2007-02	PL/Python, OIDs no longer required, internationalization of messages
7.3	2002-11-27	7.3.21	2008-01-07	2007-11	Schema, table function, prepared query[101]
7.4	2003-11-17	7.4.30	2010-10-04	2010-10	Optimization on JOINs and data warehousing functions[102]
8.0	2005-01-19	8.0.26	2010-10-04	2010-10	Native server on Microsoft Windows, savepoints, tablespaces, point-in-time recovery[103]
8.1	2005-11-08	8.1.23	2010-12-16	2010-11	Performance optimization, two-phase commit, table partitioning, index bitmap scan, shared row locking, roles
8.2	2006-12-05	8.2.23	2011-09-26	2011-12	Performance optimization, online index builds, advisory locks, warm standby[104]
8.3	2008-02-04	8.3.23	2013-02-07	2013-12	Heap-only tuples, full text search,[105] SQL/XML, ENUM types, UUID types
8.4	2009-07-01	8.4.22	2014-07-24	2014-07	Windowing functions, column-level permissions, parallel database restore, per-database collation, common table expressions and recursive queries[106]
9.0	2010-09-20	9.0.23	2015-10-08	2015-09	Built-in binary streaming replication, hot standby, in-place upgrade capability, 64-bit Windows[107]
9.1	2011-09-12	9.1.24	2016-10-27	2016-09	Synchronous replication, per-column collations, unlogged tables, serializable snapshot isolation, writeable common table expressions, SELinux integration, extensions, foreign tables[108]
9.2	2012-09-10	9.2.20	2017-02-09	2017-09	Cascading streaming replication, index-only scans, native JSON support, improved lock management, range types, pg_receivexlog tool, space-partitioned GiST indexes
9.3	2013-09-09	9.3.16	2017-02-09	2018-09	Custom background workers, data checksums, dedicated JSON operators, LATERAL JOIN, faster pg_dump, new pg_isready server monitoring tool, trigger features, view

					features, writeable foreign tables, materialized views, replication improvements
9.4	2014-12-18	9.4.11	2017-02-09	2019-12	JSONB data type, ALTER SYSTEM statement for changing config values, refresh materialized views without blocking reads, dynamic registration/start/stop of background worker processes, Logical Decoding API, GiN index improvements, Linux huge page support, database cache reloading via pg_prewarm
9.5	2016-01-07	9.5.6	2017-02-09	2021-01	UPSERT, row level security, TABLESAMPLE, CUBE/ROLLUP, GROUPING SETS, and new BRIN index[109]
9.6	2016-09-29	9.6.2	**2017-02-09**	**2021-09**	Parallel query support, PostgreSQL foreign data wrapper (FDW) improvements with sort/join pushdown, multiple synchronous standbys, faster vacuuming of large table
10		10	**TBD** (2017)	**TBD**	

Legend: Old version | Older version, still supported | **Latest version** | Latest preview version | Future release

Community-supported
Community support ended[110]



See also

- Comparison of relational database management systems
- List of databases using MVCC

References

1. "Happy Birthday, PostgreSQL!". PostgreSQL Global Development Group. July 8, 2008.
2. "PostgreSQL 9.6.2, 9.5.6, 9.4.11, 9.3.16 and 9.2.20 released!". *PostgreSQL*. The PostgreSQL Global Development Group. 2017-02-09. Retrieved 2017-02-10.
3. "Debian -- Details of package pgadmin3 in jessie". Retrieved 2017-03-10.
4. "PostgreSQL licence approved by OSI". Crynwr. 2010-02-18. Retrieved 2010-02-18.
5. "OSI PostgreSQL Licence". Open Source Initiative. 2010-02-20. Retrieved 2010-02-20.
6. "License". PostgreSQL Global Development Group. Retrieved 2010-09-20.
7. "License".
8. "Debian -- Details of package postgresql in sid". *debian.org*.
9. "Licensing:Main". *FedoraProject*.
10. "PostgreSQL". *fsf.org*.
11. "OS X Lion Server — Technical Specifications". 2011-08-04. Retrieved 2011-11-12. "Web Hosting [..] PostgreSQL"
12. "Lion Server: MySQL not included". 2011-08-04. Retrieved 2011-11-12.
13. "Mac OS X packages". The PostgreSQL Global Development Group. Retrieved 27 August 2016.
14. "What is PostgreSQL?". *PostgreSQL 9.3.0 Documentation*. PostgreSQL Global Development Group. Retrieved 2013-09-20.
15. "Contributor Profiles". PostgreSQL. Retrieved December 17, 2011.
16. Audio sample, 5.6k MP3 (http://www.postgresql.org/files/postgresql.mp3)
17. "Project name – statement from the core team". archives.postgresql.org. 2007-11-16. Retrieved 2007-11-16.
18. Stonebraker, M; Rowe, LA (May 1986). *The design of POSTGRES* (PDF). Proc. 1986 ACM SIGMOD Conference on Management of Data. Washington, DC. Retrieved 2011-12-17.
19. "PostgreSQL: History". PostgreSQL Global Development Group. Retrieved 27 August 2016.
20. Stonebraker, M; Rowe, LA. *The POSTGRES data model* (PDF). Proceedings of the 13th International Conference on Very Large Data Bases. Brighton, England: Morgan Kaufmann Publishers. pp. 83–96. ISBN 0-934613-46-X.
21. Pavel Stehule (9 June 2012). "Historie projektu PostgreSQL" (in Czech).
22. "University POSTGRES, Version 4.2". 1999-07-26.
23. Page, Dave (2015-04-07). "Re: 20th anniversary of PostgreSQL ?". *pgsql-advocacy* (Mailing list). Retrieved 9 April 2015.
24. Dan R. K. Ports; Kevin Grittner (2012). "Serializable Snapshot Isolation in PostgreSQL" (PDF). *Proceedings of the VLDB Endowment*. **5** (12): 1850–1861. doi:10.14778/2367502.2367523.
25. *PostgreSQL 9.1 with synchronous replication* (news), H Online
26. *Postgres-XC project page* (website), Postgres-XC
27. *Postgres-XL product page* (website), TransLattice
28. "Postgres-R: a database replication system for PostgreSQL". Postgres Global Development Group. Retrieved 27 August 2016.
29. "Postgres-BDR". 2ndQuadrant Ltd. Retrieved 27 August 2016.
30. Marit Fischer (2007-11-10). "Backcountry.com finally gives something back to the open source community" (Press release). Backcountry.com.
31. Bartunov, O; Sigaev, T (May 2011). *SP-GiST – a new indexing framework for PostgreSQL* (PDF). PGCon 2011. Ottawa, Canada. Retrieved 2016-01-31.
32. Bartunov, O; Sigaev, T (May 2010). *K-nearest neighbour search for PostgreSQL* (PDF). PGCon 2010. Ottawa, Canada. Retrieved 2016-01-31.
33. Geoghegan, Peter (March 23, 2014). "What I think of jsonb".
34. Obe, Regina; Hsu, Leo S. (2012). "10: Replication and External Data". *PostgreSQL: Up and Running* (1 ed.). Sebastopol, CA: O'Reilly Media, Inc. p. 129. ISBN 978-1-4493-2633-3. Retrieved 2016-10-17. "Foreign Data Wrappers (FDW) [...] are mechanisms of querying external datasources. PostgreSQL 9.1 introduced this SQL/MED standards compliant feature."
35. "PostgreSQL + Python | Psycopg". *initd.org*.
36. "SQL database drivers". *Go wiki*. golang.org. Retrieved 22 June 2015.
37. "Procedural Languages". postgresql.org. 2016-03-31. Retrieved 2016-04-07.
38. "Add a materialized view relations.". 2013-03-04. Retrieved 2013-03-04.
39. "Support automatically-updatable views.". 2012-12-08. Retrieved 2012-12-08.

40. "Add CREATE RECURSIVE VIEW syntax". 2013-02-01. Retrieved 2013-02-28.
41. Momjian, Bruce (2001). "Subqueries". *PostgreSQL: Introduction and Concepts*. Addison-Wesley. ISBN 0-201-70331-9. Retrieved 2010-09-25.
42. Bernier, Robert (February 2, 2006). "Using Regular Expressions in PostgreSQL". O'Reilly Media. Retrieved 2010-09-25.
43. "A few short notes about PostgreSQL and POODLE". *hagander.net*.
44. "SEPostgreSQL Documentation".
45. "NB SQL 9.3".
46. "MADlib". *madlib.incubator.apache.org*. Retrieved 2016-04-09.
47. "Postgres Plus Downloads". *Company website*. EnterpriseDB. Retrieved November 12, 2011.
48. *pgRouting*, PostLBS
49. "Postgres Enterprise Manager". *Company website*. EnterpriseDB. Retrieved November 12, 2011.
50. *ST Links*
51. Josh Berkus (2007-07-06). "PostgreSQL publishes first real benchmark". Retrieved 2007-07-10.
52. György Vilmos (2009-09-29). "PostgreSQL history". Retrieved 2010-08-28.
53. "SPECjAppServer2004 Result". SPEC. 2007-07-06. Retrieved 2007-07-10.
54. "SPECjAppServer2004 Result". SPEC. 2007-07-04. Retrieved 2007-09-01.
55. "Managing Kernel Resources". *PostgreSQL Manual*. PostgreSQL.org. Retrieved November 12, 2011.
56. Greg Smith (15 October 2010). *PostgreSQL 9.0 High Performance*. Packt Publishing. ISBN 978-1-84951-030-1.
57. Robert Haas (2012-04-03). "Did I Say 32 Cores? How about 64?". Retrieved 2012-04-08.
58. Khushi, Matloob (June 2015). "Benchmarking database performance for genomic data.". *J Cell Biochem*. **116**: 877–83. doi:10.1002/jcb.25049. PMID 25560631.
59. "oi_151a Release Notes". OpenIndiana. Retrieved 2012-04-07.
60. "AArch64 planning BoF at DebConf". *debian.org*.
61. Souza, Rubens (17 June 2015). "Step 5 (update): Installing PostgreSQL on my Raspberry Pi 1 and 2". *Raspberry PG*. Retrieved 27 August 2016.
62. "Supported Platforms". PostgreSQL Global Development Group. Retrieved 2012-04-06.
63. "pgAdmin: PostgreSQL administration and management tools". *website*. Retrieved November 12, 2011.
64. "pgAdmin Development Team". *pgadmin.org*. Retrieved 22 June 2015.
65. Dave, Page. "The story of pgAdmin". *Dave's Postgres Blog*. pgsnake.blogspot.co.uk. Retrieved 7 December 2014.
66. phpPgAdmin Project (2008-04-25). "About phpPgAdmin". Retrieved 2008-04-25.
67. PostgreSQL Studio (2013-10-09). "About PostgreSQL Studio". Retrieved 2013-10-09.
68. "TeamPostgreSQL website". 2013-10-03. Retrieved 2013-10-03.
69. oooforum.org (2010-01-10). "Back Ends for OpenOffice". Retrieved 2011-01-05.
70. libreoffice.org (2012-10-14). "Base features". Retrieved 2012-10-14.
71. Greg Smith; Robert Treat & Christopher Browne. "Tuning your PostgreSQL server". *Wiki*. PostgreSQL.org. Retrieved November 12, 2011.
72. Eric Lai (2008-05-22). "Size matters: Yahoo claims 2-petabyte database is world's biggest, busiest". Computerworld.
73. Thomas Claburn (2008-05-21). "Yahoo Claims Record With Petabyte Database". InformationWeek.
74. Emmanuel Cecchet (May 21, 2009). *Building PetaByte Warehouses with Unmodified PostgreSQL* (PDF). PGCon 2009. Retrieved November 12, 2011.
75. "MySpace.com scales analytics for all their friends" (PDF). case study. Aster Data. June 15, 2010. Archived (PDF) from the original on November 14, 2010. Retrieved November 12, 2011.
76. "Last Weekend's Outage". *Blog*. Geni. 2011-08-01.
77. "Database". *Wiki*. OpenStreetMap.
78. *PostgreSQL affiliates .ORG domain*, AU: Computer World
79. *Sony Online opts for open-source database over Oracle*, Computer World
80. *A Web Commerce Group Case Study on PostgreSQL* (PDF) (1.2 ed.), PostgreSQL
81. "Architecture Overview". *Reddit software wiki*. Reddit. 27 March 2014. Retrieved 2014-11-25.
82. "PostgreSQL at Skype". Skype Developer Zone. 2006. Retrieved 2007-10-23.
83. "How Much Are You Paying For Your Database?". Sun Microsystems blog. 2007. Retrieved 2007-12-14.
84. "Database – MusicBrainz". MusicBrainz Wiki. Retrieved 5 February 2011.
85. Duncavage, Daniel P (2010-07-13). "NASA needs Postgres-Nagios help".
86. Roy, Gavin M (2010). "PostgreSQL at myYearbook.com" (talk). USA East: PostgreSQL Conference.
87. "Keeping Instagram up with over a million new users in twelve hours". Instagram-engineering.tumblr.com. 2011-05-17. Retrieved 2012-07-07.
88. "Postgres at Disqus". Retrieved May 24, 2013.
89. Matthew Kelly (27 March 2015). *At The Heart Of A Giant: Postgres At TripAdvisor*. PGConf US 2015. (Presentation video (https://www.youtube.com/watch?v=YquXmwZNnfg))
90. "Yandex.Mail's successful migration from Oracle to Postgres [pdf] | Hacker News". *news.ycombinator.com*. Retrieved 2016-09-28.

91. **Cite error: The named reference `CitusCloud` was invoked but never defined (see the help page).**
92. Alex Williams (1 April 2013). "Heroku Forces Customer Upgrade To Fix Critical PostgreSQL Security Hole". TechCrunch.
93. Barb Darrow (11 November 2013). "Heroku gussies up Postgres with database roll-back and proactive alerts". GigaOM.
94. Craig Kerstiens (26 September 2013). "WAL-E and Continuous Protection with Heroku Postgres". Heroku blog.
95. "EnterpriseDB Offers Up Postgres Plus Cloud Database". Techweekeurope.co.uk. 2012-01-27. Retrieved 2012-07-07.
96. O'Doherty, Paul; Asselin, Stephane (2014). "3: VMware Workspace Architecture". *VMware Horizon Suite: Building End-User Services*. VMware Press Technology. Upper Saddle River, NJ: VMWare Press. p. 65. ISBN 978-0-13-347910-2. Retrieved 2016-09-19. "In addition to the open source version of PostgreSQL, VMware offers vFabric Postgres, or vPostgres. vPostgres is a PostgreSQL virtual appliance that has been tuned for virtual environments."
97. Al Sargent (15 May 2012). "Introducing VMware vFabric Suite 5.1: Automated Deployment, New Components, and Open Source Support". VMware blogs.
98. Jeff (14 November 2013). "Amazon RDS for PostgreSQL – Now Available". Amazon Web Services Blog.
99. Alex Williams (14 November 2013). "PostgreSQL Now Available On Amazon's Relational Database Service". TechCrunch.
100. "Amazon Aurora Update – PostgreSQL Compatibility | AWS Blog". *aws.amazon.com*. Retrieved 2016-12-01.
101. Vaas, Lisa (2002-12-02). "Databases Target Enterprises". *eWeek*. Retrieved 2016-10-29.
102. Krill, Paul (November 20, 2003). "PostgreSQL boosts open source database". *InfoWorld*. Retrieved 2016-10-21.
103. Krill, Paul (January 19, 2005). "PostgreSQL open source database boasts Windows boost". *InfoWorld*. Retrieved 2016-11-02.
104. Weiss, Todd R. (December 5, 2006). "Version 8.2 of open-source PostgreSQL DB released". *Computerworld*. Retrieved 2016-10-17.
105. Gilbertson, Scott (February 5, 2008). "PostgreSQL 8.3: Open Source Database Promises Blazing Speed". *Wired*. Retrieved 2016-10-17.
106. Huber, Mathias (July 2, 2009). "PostgreSQL 8.4 Proves Feature-Rich". *Linux Magazine*. Retrieved 2016-10-17.
107. Brockmeier, Joe (September 30, 2010). "Five Enterprise Features in PostgreSQL 9". *Linux.com*. Linux foundation. Retrieved 2017-02-06.
108. Timothy Prickett Morgan (12 September 2011). "PostgreSQL revs to 9.1, aims for enterprise". *The Register*. Retrieved 2017-02-06.
109. Richard, Chirgwin (7 January 2016). "Say oops, UPSERT your head: PostgreSQL version 9.5 has landed". *The Register*. Retrieved 2016-10-17.
110. "Versioning policy". PostgreSQL Global Development Group. Retrieved 2012-01-30.

Cite error: A list-defined reference named "SQL_standard" is not used in the content (see the help page).
Cite error: A list-defined reference named "Conformance" is not used in the content (see the help page).
Cite error: A list-defined reference named "Git" is not used in the content (see the help page).

Further reading

- Obe, Regina; Hsu, Leo (July 8, 2012). *PostgreSQL: Up and Running*. O'Reilly. ISBN 1-4493-2633-1.
- Krosing, Hannu; Roybal, Kirk (June 15, 2013). *PostgreSQL Server Programming* (second ed.). Packt Publishing. ISBN 978-1-84951-698-3.
- Riggs, Simon; Krosing, Hannu (October 27, 2010). *PostgreSQL 9 Administration Cookbook* (second ed.). Packt Publishing. ISBN 1-84951-028-8.
- Smith, Greg (October 15, 2010). *PostgreSQL 9 High Performance*. Packt Publishing. ISBN 1-84951-030-X.
- Gilmore, W. Jason; Treat, Robert (February 27, 2006). *Beginning PHP and PostgreSQL 8: From Novice to Professional*. Apress. p. 896. ISBN 1-59059-547-5.
- Douglas, Korry (August 5, 2005). *PostgreSQL* (second ed.). Sams. p. 1032. ISBN 0-672-32756-2.
- Matthew, Neil; Stones, Richard (April 6, 2005). *Beginning Databases with PostgreSQL* (second ed.). Apress. p. 664. ISBN 1-59059-478-9.
- Worsley, John C; Drake, Joshua D (January 2002). *Practical PostgreSQL*. O'Reilly Media. p. 636. ISBN 1-56592-846-6.

External links

- Official website (https://www.postgresql.org)
- PostgreSQL wiki (https://wiki.postgresql.org/wiki/)

Wikimedia Commons has media related to

- https://www.depesz.com/
  - https://explain.depesz.com/ – PostgreSQL's explain analyze made readable – documented in Understanding EXPLAIN (http://www.dalibo.org/_media/understanding_explain.pdf)
- PostgreSQL (https://www.dmoz.org/Computers/Software/Databases/PostgreSQL/) at DMOZ

PostgreSQL.

Wikibooks has a book on the topic of: ***Postgres RDBMS***

Retrieved from "https://en.wikipedia.org/w/index.php?title=PostgreSQL&oldid=770090629"

Categories: Client-server database management systems | Cross-platform software | Free database management systems | Free software programmed in C | ORDBMS software for Linux | RDBMS software for Linux | PostgreSQL | Software that uses wxWidgets

---