PERCONA

WHITEPAPER

Why Choose PostgreSQL?

PERCONA

Why Choose PostgreSQL?

PostgreSQL, commonly referred to as Postgres, is a powerful, 100% free and open source, object-relational database system, with over 30 years of active development. It has a strong reputation for reliability, feature robustness, and performance.

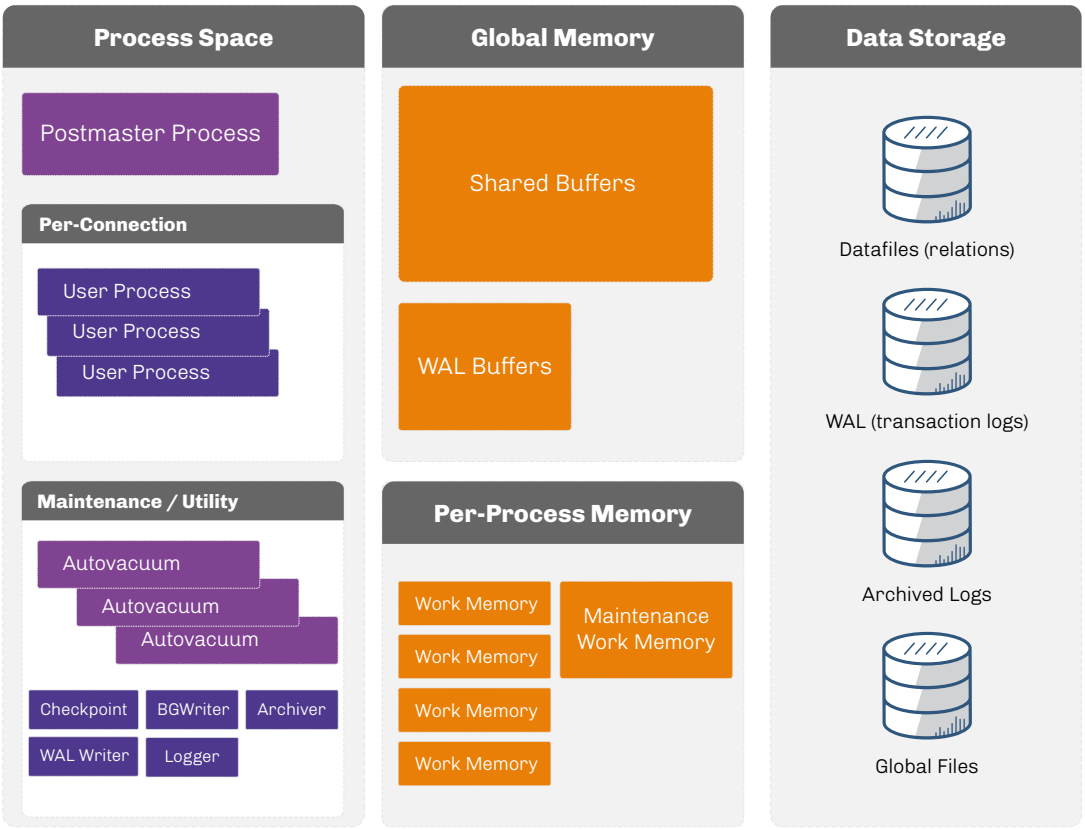
PostgreSQL dates back to 1986 as part of the POSTGRES project at the University of California at Berkeley. Postgres has seen its popularity increase over the years to become the fourth most popular database in the world and is well-supported by its active community.

Postgres is an advanced and powerful SQL-compliant, open-source, object-relational, database management system. Its strong underlying technology makes it popular with companies that perform complex and high-volume data operations. Postgres is often the database of choice for those looking to move away from Oracle's proprietary software.

Postgres is also popular with developers due to features such as:

- Native partitioning
- Parallel query
- Supports foreign data wrappers
- Powerful JSON features
- Streaming and logical replication
- Availability of many open source tools for HA, backups and monitoring

What is Postgres?



Postgres is an object relational database, with a focus on extensibility and technical standards compliance. It sits in the middle ground between classic object databases, like Informix, that represent and store data in objects, and relational databases, like MySQL, that represent and store data in tables. This different set-up, and the fact that Postgres originated in academia, means that it uses slightly different terms and concepts from relational databases.

Relational Term	Postgres Term
Table ir Index	Relation
Row	Tuple
Column	Attribute
Data Block	Page, when block is on disk
Page	Buffer, when block is in memory

In Postgres, the data resides in the database and is manipulated through queries written in SQL. An object relational database bridges the gap between traditional relational databases and the object based modelling technique used in many programming languages, such as Java or .NET.

Postgres is ACID compliant, this means that it supports a set of properties (Atomicity, Consistency, Isolation, and Durability) intended to guarantee data validity even in the event of errors, power failures, etc. This enables Postgres to be transactional, which means that an entire related series of database events either completes or fails in full.

Postgres also supports Stored Procedures that are as powerful as Oracle. This makes it easier for people to convert the business logic (PL/SQL) in an Oracle database to PostgreSQL (PL/pgSQL). In fact, there are several tools and extensions built around Postgres which incorporate many similar features to Oracle. All these make Postgres an appealing option for people looking to migrate away from Oracle.

Postgres is extendable - it provides support for custom views and triggers, and supports stored procedures and functions. It manages concurrency through Multiversion Concurrency Control (MVCC). MVCC basically eliminates the requirements for read locks, which prevent reads on a table while data is being written to it, as well as ensuring ACID compliance.

Postgres enables you to define custom data types, indexes, and more. If you can't fulfil a particular requirement through the standard features of Postgres, you can create custom objects to achieve it. You can create these objects in a wide range of procedural languages, enabling you to move business logic to the database rather than the application.

Postgres users appreciate these innovative database features:

- Parallel query - which allows you to design query plans that leverage multiple CPUs.
- Declarative partitioning - which gives you the ability to specify how data is partitioned.
- Streaming replication - which ensures data consistency across servers.
- Logical replication - which replicates data based on a defined identity property, like a primary key.

Many large companies are running Postgres to support a variety of needs. They use it because it is a highly stable database, it includes high availability and security, and it supports SQL and NoSQL in a single environment.

Migrating from Oracle

Many companies are looking to move into open source software, due to a variety of reasons. One of the strongest motivations is cost savings. Rather than being locked into a proprietary vendor and trapped into paying high license fees, many companies discover they can get the same results, response, and reliability with open source solutions.

Migrating your database can be complicated, moving from Oracle can be difficult, and effectively porting a database application from one backend to another can prove challenging. This is one area where Postgres shines. There are many contributions to Postgres which focus on Oracle compatibility. This makes it much easier and less painful for someone on Oracle to switch to Postgres.

Postgres was designed to follow the same type of data integrity rules as Oracle, it accepts many of the same functions, stored procedures, and customizations. This allows companies to plan and execute a migration to Postgres in far less time, with much less complexity and fewer things to change. This also means less downtime for your business. All of these points are compelling reasons to consider Postgres as your open source database.

Financial Services

Postgres is popular with financial services, often because companies have migrated to Postgres from a legacy Oracle environment. The ACID compliance and support for transactions are key reasons that a financial services company might consider Postgres.

As an example of ACID compliance, we can consider a securities trader and a specific transaction. A “trade” is made up of the sale of some items, and the purchase of others. We cannot record the purchase of certain securities unless we are able to record the sale of the other securities. Even though each of these actions is handled discreetly by the database, the entire set of actions represents a single transaction and must be either committed or rejected as a whole.

Postgres has extensive analytical capabilities, and was one of the first databases to support window based analytic functions. A window function performs a calculation across a set of table rows that are related to the current row. This is comparable to the type of calculation done with an aggregate function, but, unlike regular aggregate functions, use of a window function does not cause rows to become grouped into a single output row. A window function returns each row with its own data and the aggregated information.

In the financial world, this might be used to determine the average portfolio value of a set of customers, and to display the average next to each user’s individual value. If we simply used an aggregate function, we would get the average portfolio value, but we would not see that average as it compares to each user’s value.

If the raw data in the accounts table is:

state	account_id	current_value
MA	617	1287497
PA	508	1011736
PA	412	975284
PA	724	1324799
MA	808	958365

We could write a query

```
SELECT AVG(current_value) GROUP BY state FROM accounts;
```

to determine the average current_value grouped by state:

state	account_id
MA	1122931
PA	1103939.67

A query on the same data using a window function is

```
SELECT state, account_id, current_value, AVG(current_value) OVER
(PARTITION BY state ORDER BY state) FROM accounts;
```

which returns:

state	account_id	current_value	avg(current_value)
MA	617	1287497	1122931
MA	508	1011736	1122931
PA	412	975284	1103939.67
PA	724	1324799	1103939.67
PA	808	958365	1103939.67

If your goal is simply to determine the average portfolio value, an aggregate function is fine. If you want to determine how specific accounts are performing against the average value, then a window based function is needed. The OVER clause is what makes this a window function.

Postgres supports all standard window functions and allows for the creation of custom window functions. These analytic query statements provide deep insight into your data.

Scientific Analysis

Postgres is a great choice for scientific analysis due to its ability to process unstructured data. In many cases, the data for analysis is coming from disparate sources that have their own field names, content, and structure. An object relational database like Postgres is well-positioned to ingest, process, store, and query that data.

Some of the data might be best queried with NoSQL syntax, that is generated by an application. Postgres can handle queries using either SQL or NoSQL syntax, providing analysts with an extensive range of possibilities for how they query the data.

This is another area where data integrity is critical. If you are tracking data on drug testing, it is important that you are certain that the results being returned reflect complete and current data. Analytic functions, including the window functions mentioned previously, are important in this arena.

Managing Web Traffic

When users visit a website, they have certain expectations regarding performance. No one likes to be kept waiting while a page loads slowly, or to discover that a page doesn't fully load.

PostgreSQL handles concurrency by using multiple versions of the same record within table. This means that PostgreSQL requires less locking. It allows Postgres to handle the high concurrency requirements of web applications, and can successfully scale to meet growing demand.

Another reason PostgreSQL excels is the API first design. PostgreSQL and its ecosystem provides for the easy development of REST APIs on top of Postgres. Please take a look at the postgREST (http://postgrest.org/en/v6.0/) project for more information.

The process of building out and displaying a web page is no longer simple. Gone are the days where a web page remains static. Now even public websites are geo-location aware, and data-driven. An example of a geo-location aware, data-driven website is a news website, which includes weather predictions based on the user's location. The prediction should be displayed, and updated in real-time. The website may ALSO customize based on the user's personal preferences and interests. An API-driven development approach meets this need and PostgreSQL would be a natural choice.

Postgres is known for scaling up with vertically with increased resources. It can also scale horizontally using the hot standby (standby allows READS) feature, to accommodate a high level of transactions in high traffic environments.

Meeting Geospatial Requirements

As mentioned in the section above, the processing of geospatial queries is an area of strength for Postgres. This is normally accomplished through the use of the PostGIS extension. This extension is open source and can be integrated into a new or existing Postgres environment.

An example of where this might be used is when prepaying for time at a designated parking meter. When a user request is issued, one of the key components of the request is the desired location of the space. Obviously, it's not great if the space is too far from your destination!

Different spaces may incur different costs, so the user needs to gather all relevant information to make the best possible decision. Required data includes:

- The distance from the space to end location
- Time of availability
- Cost for parking
- Distance from current location.

With PostGIS, this location-based information can be accurately and quickly determined. Other information, such as the parking costs for a specific location, do not require location-based information, so standard queries can be issued to return that information.

The company that offers this service needs to perform extensive analysis of usage data, that may also have a location component. For example, if you determine that the primary users of your parking come from within ten miles of a certain location, you can target your advertising to focus on people in that area. You would also want to conduct time-based queries to determine periods of higher usage, to potentially charge a higher price at those times. Postgres and PostGIS enable those queries to be run quickly and effectively.

PERCONA

How Percona Can Help Your PostgreSQL Deployment Succeed

Percona Distribution for PostgreSQL

To create Percona Distribution for PostgreSQL we took the best and most critical enterprise-level components from the open-source community, designed and tested to work together in one single source.

Percona Distribution for PostgreSQL 11 delivers a single source that provides an enterprise-grade, open-source installation of PostgreSQL Core Distribution. You no longer need to find solutions for common requirements, like high availability and backup, on your own. Installing Percona Distribution for PostgreSQL provides a complete package for PostgreSQL that meets the needs of all companies, large and small. This distribution is available without charge from Percona.

Percona PostgreSQL Support and Consulting Services

Whether you're running Percona Distribution for PostgreSQL or another PostgreSQL distribution, Percona offers a comprehensive, responsive, and cost-effective Support plan to help your organization's deployment succeed. Percona also offers expert PostgreSQL Consulting services to help with your deployment and ensure optimal ongoing performance

Conclusion

Postgres is continuing to grow in popularity. As discussed, this is partly due to companies considering open source alternatives to costly Oracle installations.

The ability to scale a Postgres environment means it can handle any traffic requirements you have. Extensive customizations and extensions enable you to utilize tools that help your business develop and grow. You can feel confident that Postgres is able to meet your evolving database needs, now and in the future.

Contact Us

Percona can help you maximize your database performance with open source database support, managed services, and consulting services.

For more information, contact us at +1-888-316-9775 (USA), +44 203 608 6727 (Europe) or have us reach out to you.