PERCONA

10 Deadly PostgreSQL Mistakes

Every company is different. The missteps they make when designing their PostgreSQL database, however, are often not. Below, we've compiled **the 10 most common mistakes** we've seen from our years of helping companies successfully use PostgreSQL.

#10 Tool Selection

Tools are supposed to help us do more with less effort. However, deploying the wrong tools or misunderstanding how they work can have the opposite effect.

tip: **Make sure you are using the right tools for the job. Check the tool versions you are using; an old tool can hamper productivity.**

#9 Lax Security Practices

Organizations don't always implement database security best practices. Common mistakes include not using SSL to encrypt traffic, over-granting permissions or not using effective roles, and storing everything in the public schema.

tip: **Getting database security right is crucial. Regularly audit your activity and check that security best practices are followed.**

#8 Improper Use of Vacuum

Vacuuming compacts and optimizes dead space, which can greatly speed up performance. But when a vacuum is running, it can often cause a performance hit.

tip: **Manage and plan ahead. Schedule vacuuming for periods with lower traffic volumes, or auto-schedule when data volumes are enough to make it worthwhile.**

#7 Poor Connection Management

A large number of open connections in PostgreSQL can lead to severe performance degradation over time. Every connection is a new process, and there is a high overhead for new connections.

tip: **Thinking through connection management in real-world situations can help fine-tune performance.**

#6 Relying on the PostgreSQL Default Settings

PostgreSQL is NOT tuned out of the box, and its default settings will not help your database perform at its best.

tip: **Check the default settings on your database and determine whether they are right for your application and use case.**

#5 Over-/Under-indexing

Creating an index of data within your database makes it easier and faster to return results when a query takes place. But over- and under-indexing can create additional work for the database.

tip: **Understand the available index models and apply the correct one based on your needs.**

#4 Extension Foo

Extensions are wonderful, but you need to be careful - PostgreSQL can be extended by things that will break it.

tip: **Make sure you understand the extensions you want to add to your implementation. Stick with reliable sources, and disable any odd or new ones if you are experiencing problems.**

#3 No Backup or High Availability Plan

Many companies are not prepared to recover from an outage, whether that is failover or from backup.

tip: **Remember to check your WAL settings, use automated failover, and have a local backup and remote. Your retention minimum should be seven days.**

#2 Not Tuning the Database for the Workload

Developers tend to use the same database and settings for every application they create. However, they should be tuning the database to support a specific application workload every time.

tip: **Deploy an observability and analysis tool to get deep insight into your database and see if patterns emerge.**

#1 Poor Database Design and Architecture

Typical design architecture mistakes we see include bad data type selection, poor schema design, overreliance on ORM frameworks for SQL, undersized or oversized hardware, and storing more data than is needed for longer than needed.

tip: **A complicated database design doesn't mean an optimized database. Understand how you need your database to work so you can choose the right hardware and applications to support it.**

For more information please contact us at **+1-888-316-9775** (USA), **+44 203 608 6727** (Europe), or via email at sales@percona.com

PERCONA

27062023