Database Design Best Practices -

(Understanding common mistakes)

Jobin Augustine

PERCONA

1.

Poor Selection of Data types

- Common mistake when migrating Other database systems to PostgreSQL
- **40%** performance penalty is common.
- Space wastage
- Costly Indexes.

Data Type Best Practice.

- **PostgreSQL has 40+ main data types**
- **Use of NUMERIC instead of INTEGER or BIGINT is the common mistake**
- **Select data types based on:**
  - Takes the least amount of space and complexity to hold the data
  - Avoid runtime data type conversions in.
  - Look for built in functionality (example inet,macaddr)

2.

Unused Indexes

- **Indexes are created based on assumptions or carried over from other database systems**
- **Never monitored or assessed**
- **PostgreSQL tables are heap structures. Cheap to scan. Index scans are not always better.**
- **Underestimates the impact of unused Indexes**
  - Each Transaction slows down
  - Write amplification.
  - More WAL is generated
  - Occupies shared buffers / cache
  - Impact on the Autovacuum.

3.

PostgreSQL Defaults

- **The defaults are set to get a minimal functional system in a smallest possible hardware.**
- **Tuning is mandatory**
- **Tuning must consider**
  - Available hardware resources (CPU & Memory)
  - Type of workload. OLTP vs OLAP
  - Connections
  - Storage.

4.

Poor Defaults by DBaaS

- DBaaS solutions are found to set very odd values
- Assumption of Cloud providers gives a best tuned system is plain wrong.

5.

Poor connection Management

- **Misunderstands the overhead of connections. poor connection management.**
  - Large value for max_connections
  - Large number of idle connections and idle in transactions

PERCONA

connection Management

- Each connection creates a backend process
- Not much point in having connections greater than 10x of CPU count (A generic rule of thumb)
- high max_connection than what host can accommodate is an open door for DoS attacks.
- Creation of new database connection is costly.
- There is huge overhead in maintaining connections (snapshots, lock management and memory)
- Importance of connection pooler

6.

Parameters

Frequent Areas of outages and poor performance

- Autovacuum, workers, vacuum_cost_limit
- Work_mem & maintenance_work_mem
- checkpointing and bgwriter

Parameters has to be adjusted according to host machine and workload.

PERCONA

7.

Multi-Tenancy

- Multi-tenancy is commonly used in other database system to reduce the licence cost.
- Not a valid reason for PostgreSQL.
- Single fatty database is a bad idea.
- Database softwares has many contention points including the concurrency control.
- Noisy neighbour problem in multi-tenancy is hard to tackle.

8.

Quick decision - DBaaS

- **Jumping to DBaaS Solutions as the provisioning is easy.**

PERCONA

DBaaS Challenges

- **How do we upgrade?**
  - In-place upgrade of a big size database
  - pg_upgade with hardlink option
  - predictable downtime?
- **Troubleshooting.**
  - Dashboard is not enough. Linux tools are very essential.
  - Indepth analysis of CPU / IO utilization.
- **Can we restore**
  - Specific WALs?
  - Backups which are not restorable to the way we want is as good as **No backup**
- **Tablespaces**
- **Segregation of Disks**
  - A database system is mostly IO centric.

DBaaS Challenges

- **Extensions of choice**
  - columnar store, sharding, time series data, etc.
  - plpython
- **Custom Development of extensions**
- **DR capability testing - compliance requirements**
- **Custom Dictionary for text searches**
- **Logical replication slot continuity**
  - What is possible with Patroni.
- **Costs high**
  - 2x, upto 5x are reported : https://news.ycombinator.com/item?id=24608538

9.

Usage of “PostgreSQL” like software

- **PostgreSQL derivatives and “Compatible” Softwares**

Want to edit, but don't see an edit button when logged in? Click here.

PostgreSQL derived databases

A list of PostgreSQL derived forks and rebranded distributions in alphabetical order.

Name	Vendor	License	Availability	Notes
AgensGraph	Bitnine	Apache2	2016-	PostgreSQL + Graph Model features (Support graph storage and Cypher query language)
Aster Data	Teradata	Proprietary	2005-....	PostgreSQL + Map/Reduce
BDR	2ndQuadrant	BSD	2014-	PostgreSQL Multi Master, contributed actively back to Core PG
Bizgres	Greenplum	BSD	2005-2007	PostgreSQL + BI features
Cybercluster	Cybertec	BSD	2007-2010	Clustering (pgCluster fork)
Greenplum Database	Greenplum	Apache2	2005-....	PostgreSQL + BI features (formerly known as "Bizgres MPP") [1]
ExtenDB	ExtenDB	Proprietary	2003-2007	PostgreSQL + BI Features [2]
FUJITSU Enterprise Postgres	Fujitsu	proprietary	2006-....	Full PostgreSQL compatibility with additional functionality [3]
GresCube	NTT DATA	Proprietary	2012-....	Database appliance solution based on PostgreSQL [4]
GridSQL	EnterpriseDB	GPL	2007-2010	PostgreSQL + BI Features (formerly ExtenDB) [5]
Great Bridge PostgreSQL	Great Bridge LLC	BSD	1999-2001	PostgreSQL re-distribution
HadoopDB	Yale University	Apache License V2.0	2009-....	PostgreSQL + shared-nothing cluster + Hadoop [6]
Hadapt	Teradata	Proprietary	2011-....	HadoopDB fork
Mammoth	Command Prompt	BSD	2005-2010	PostgreSQL + proprietary replication + extensions
Netezza	IBM	proprietary	2002-....	Appliance based on PostgreSQL SQL engine
NuSphere UltraSQL	NuSphere	proprietary	2002-2003	Native Win32 port of PostgreSQL
ParAccel	Actian	proprietary	2005-....	PostgreSQL + BI features [7]
Pervasive PostgreSQL	Pervasive	BSD	2005-2006	PostgreSQL re-distribution
pgCluster	SRA	BSD	2002-2005	Clustering (Share Nothing)
pgCluster-II	SRA	BSD	2006-2007	Clustering (Shared Disk)
pgPool-II	pgPool GDG	BSD	2006-....	Clustering (Connection Pooling / Replication / Load-Balancing)
PipelineDB	PipelineDB	GPL v3	2015-....	Streaming SQL
PostgresForest	NTT DATA	BSD	2006-2010	Clustering / PostgresForest is a fork of the JDBC driver, not from the backend code.
EDB Postgres Advanced Server	EnterpriseDB	proprietary	2008-....	PostgreSQL + Oracle compatibility + security + performance tools + developer tools + DBA tools, formally Postgres Plus Advanced Server / EnterpriseDB AS [8]
Postgres Pro Enterprise	Postgres Professional	proprietary	2016-...	PostgreSQL + enterprise features [9]
Postgres-R	PGDG	BSD	2006-2010	Clustering
Postgres-X2	PGX2DG	BSD	2015-	Clustering (formerly Postgres-XC)
Postgres-XC	PGXCDG	BSD	2010-2013	Clustering [10]
Postgres-XL	PGXLDG	BSD	2014-....	Clustering

PERCONA

What is PostgreSQL

- What you get from postgresql.org is PostgreSQL
- Stability and Availability.
- Performance.
- Quality of code.

- “XYZ PostgreSQL” or “ABC PostgreSQL” are not PostgreSQL. Expect bugs and problems.
- Performance claims, be sceptical. test with your load.
- Lock-in

10.

Extensions

- **Major cause of crashes reported to us**
- **Performance overhead.**

Extensions

- **Use when it is not avoidable**
- **unload / remove when not in use**
- **Native features like partitioning is powerful enough.**

Antoine de saint-exupéry

Perfection is achieved,

not when there is nothing more to add,

but when there is nothing left to take away.

PERCONA

11.

High Availability

- Assess the real objectives and business value.
- What is at minimum required.
- Take the most widely used / popular in the community.
- Keep the recommendations a check.

Perceived HA is Not HA

Story of 8 hour outage.

PostgreSQL -> PgPool -> HAproxy with Zookeeper

- Automatic pgpool starting
- Zookeeper managing the DCS

Availability 9s are to be validated by actual data

- Simple systems are found to give better actual availability.
- Measurement and Interpretation is important.
  - https://en.wikipedia.org/wiki/High_availability
  - Outage due to any dependent component also accounted for the unavailability by very definition.
- Unacceptable performance also accounted for unavailability.

Mean time between failures also need to be measured and can make more sense than 9s.

12.

Poor use of Indexes

- **Partial Indexes are seldom used. many are not even aware**
- **Expression indexes**

PERCONA

13.

Data Retention

Anything ever growing in size is not sustainable. But Cancerous

Data Retention

- **Partitioning**
- **Archive tables.**
- **Purging.**

PERCONA

14.

Decision making

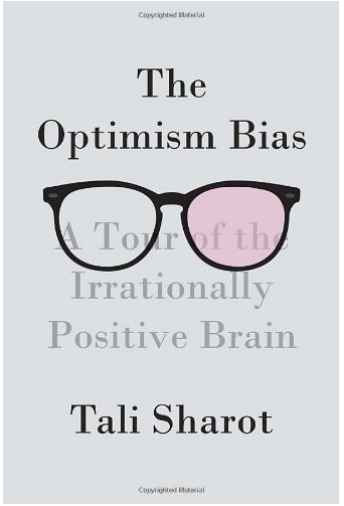
DEV

- Excitement
- Quick Decisions
- Advantages

Vs

Ops

- Skepticism
- Critical
- What are the objectives



The inclination to overestimate the likelihood of encountering positive events and underestimating the likelihood of encountering negative events.

PERCONA

Test of Time

Those who walks in the front, steps on the land mine

- Assess how popular is certain technologies
- Get unbiased information (collect plus and minus)
- Test the waters - non-critical environment

PERCONA

Ex:
Version Upgrade

- Dev and Test environments first - Incremental
- Watch and report issues - Assess possible problems
- Demand End-to-End testing.
- Wait for second minor version - Reduce the risk
- Three trial runs for Main production upgrade and prepare runbook.

Q & A