credativ

Self-Driving PostgreSQL

Should we allow our database to auto-tune itself while we sleep?

Josef Machytka <josef.machytka@credativ.de>

2026-01-26 - Prague PostgreSQL MeetUp

credativ GmbH

- Founded 1999 in Jülich, Germany
- Close ties to Open-Source Community
- More than 40 Open-Source experts
- Consulting, development, training, support (3rd-level / 24x7)
- Open-Source infrastructure with Linux, Kubernetes, Proxmox
- Open-Source databases with PostgreSQL
- DevSecOps with Ansible, Puppet, Terraform and others
- Since 2025 independent owner-managed company again

Josef Machytka

credativ

- Professional Service Consultant - PostgreSQL specialist at credativ GmbH
- 33+ years of experience with different databases
- PostgreSQL (13y), BigQuery (7y), Oracle (15y), MySQL (12y), Elasticsearch (5y), MS SQL (5y)
- 10+ years of experience with Data Ingestion pipelines, Data Analysis, Data Lake and Data Warehouse
- 3+ years of practical experience with different LLMs / AI / ML including architecture and principles
- From Czechia, living now 12 years in Berlin

- LinkedIn: linkedin.com/in/josef-machytka
- Medium: medium.com/@josef.machytka
- YouTube: youtube.com/@JosefMachytka

- GitHub: github.com/josmac69/conferences_slides
- ResearchGate: researchgate.net/profile/Josef-Machytka

All My Slides:

Recoded talks:

Many Aspects We Do Not Want to Talk About

- The discussion is clouded by hype, misconceptions, disappointment and fear
- We must sincerely discuss ALL the issues related to this topic

Images in slides created by ChatGPT

We Must Keep Asking Questions

- Simon Riggs: The Next 20 Years (PGConf.EU 2023)

```
I encourage everybody not to get too confident
that because PostgreSQL is number one,
therefore we are doing everything right.
That conclusion does not fit.

It just means we're number One, right?
That's the only thing it means.
We still need to ask stupid questions...

Why don't you do it this way?
Why don't you do that?
Why doesn't it work like this?

They're good questions.
```

Image (c) Katerina Machytkova

Heroic Manual Tuning

- Default settings prefer ”safe” compatibility
- Community advice focuses on ”good enough for most”
- Heroic DBAs tried to optimize parameters for a specific workload
- Loop of observe -> research -> test -> apply -> verify
- Time-consuming, error-prone, anecdotal one time fixes
- Production pressure, restart/DDL risk, multi-factor causality
- Not scalable for many instances or dynamic workloads

Images in slides created by ChatGPT

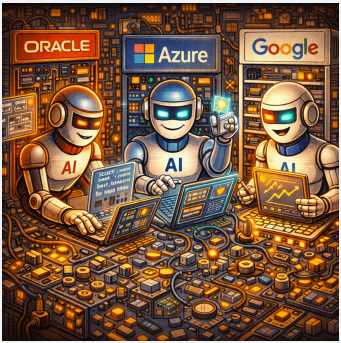
AI Is Quickly Reshaping IT Industry

credativ

- AI-assisted coding is quickly changing software development
- Future is heading towards more AI-assisted work
- With smaller teams & smaller companies doing more complex tasks
- Dynamic startups are already hungry for self-driving PostgreSQL
- Many companies manage dozens, hundreds, or even thousands of DBs
- Optimal manual tuning of these instances is practically impossible
- Adoption of AI-assisted databases is inevitable

Big Database Players and AI

- Oracle AI Database 26ai
  - Real time SQL plan management, AI health checks
  - Automatic indexing, plan regression prevention
- Google Cloud Platform
  - Gemini in databases - Cloud SQL, Spanner, AlloyDB
  - AI assisted troubleshooting and performance tuning
- Microsoft Azure
  - CoPilot in Azure SQL for automatic tuning suggestions
  - Automatic fallback to last known good execution plan
  - SQL queries from natural language, automatic monitoring
- IBM - AI database assistant
- PostgreSQL cannot stay behind in this trend



PostgreSQL And Artificial Intelligence

- How it currently looks like with AI in PostgreSQL ecosystem?

Concerns: AI as a potential "automated incident generator"

Market Hype: Vendors promising transformative results

Reality Check: Current AI tools have significant limitations

AI as Automated Incident Generator

- Major clouds already strongly depend on AI tools
- AI tools can now quickly mitigate small incidents
- Problems with multiple AI tools interacting unpredictably
  - AWS US-EAST-1 outage caused by AI DevOps tools interacting (article)
  - Cloudflare internet outage caused by AI generated config file (article)
  - User request caused segmentation fault in Vertex AI Agent system (article)

Big Outages of AI Online Tools

- Companies already strongly depend on AI tools
- Big outages disrupted many services relying on them
- Not working coding assistants, monitoring tools, chatbots
- AI availability is now business critical

  - AWS outage made several AI tools unavailable (article)
  - OpenAI telemetry deployment overwhelmed K8s control plane (report)
  - Cloudflare outage disrupted major platforms including ChatGPT (article)

Security And Privacy Incidents

- AI tools can inadvertently leak sensitive data
- Misconfigured AI agents can access unauthorized resources
- New attack vectors exploiting AI vulnerabilities
- ”AI Prompt Injection” attacks in databases
  - Meta AI chatbot exposed users’ prompt and generated content (article)
  - Replit AI coding tool deleted a live production database (article)
  - Prompt injection attacks tricking AI agents to execute commands

Commercial Hype Around AI Tuning Tools

- Companies developing tuning tools are in a tough spot
- Under strong commercial pressure
- Deemed to focus on easy wins, commercially deployable
- Investors want to see quick returns on investment
- Leads to relatively minor performance improvements
- Produces a lot of marketing hype around small gains
  - Breakthrough improvements require more time and effort
  - PostgreSQL community must work together on breakthroughs
  - It is the "next big thing" we can all work towards
  - What about "Community lightweight version" Vs.
  - "Commercial grade version" of AI tuning tools ?
  - Free with lighter features vs. Paid with advanced features ?

Some Early Attempts Already Failed

- OtterTune was a high-profile AI tuning startup
- Founded by CMU database researchers
- Promised significant performance improvements
- Shut down in June 2024 due to financial issues
- Big learning they made - lack of customer retention
- Customers tuned their databases once, then churned
- AI tuning viewed as a one time feature
- Was not seen as a continuous service
- Cloud provides offered similar features for free

AI as Lazy Loser & Clever Hans

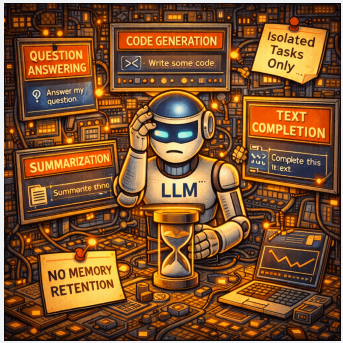
- Without human guidance AI often takes ”the easy way”
- Uses shortcut features which are predictive enough
- Ignores harder to understand causal features
- ”Clever Hans” effect: models exploit strange correlations
- Specification gaming: exploits loopholes in goals and rules
- Reward hacking: finds unintended ways to maximize reward

LLMs Are Not The Right Tool For The Job

- LLMs do not understand how PostgreSQL works
- Can only tell what people wrote about PostgreSQL
- Training data is incomplete, biased, outdated or wrong
- Even before AI, finding information was ”hit or miss”
- Many human errors, misconceptions, misunderstandings
- Leads to plausible-sounding but wrong answers
- LLM alone cannot even calculate basic math correctly
- LLMs can only support experts, not replace them

The Biggest Issues of LLMs

- LLMs are like ”childish parrots” with tunnel vision
- Take each prompt mostly as an isolated task
- Easily forget context of previous interactions
- Can ignore whole parts of the prompt
- Results are unstable even with same prompt
- They excel at small, precisely focused tasks
- Fail with complex, multi-faceted problems
- Excel on creative writing and brainstorming
- Fail on technical accuracy with many details



RAGs Useful For Preserving Knowledge

- Second big AI hype wave - Retrieve Augmented Generation
- Additional domain specific knowledge for LLMs
- Stored and processed in vector databases
- Very useful for customizing LLMs to specific domains
- We can use RAG to provide tuning rules specific to us
- But it just extends the ”parrot” behavior
- Does not make LLMs able to tune database

Agentic AI To The Rescue?

- Can autonomously perform complex tasks
- Integrates LLMs with tools, APIs, and external data
- MCP protocol (JSON) standardizes tools integration
- ”Perception-Action Loop”
  - Observe State (perception / monitoring)
  - Reason about Issue (reasoning & decision making)
  - Formulate Plan (planning & action selection)
  - Execute Tool (action)
  - Verify Outcome (evaluation & learning)
- Inherits all limitations of LLMs
- Can incorporate ML models specialized for PostgreSQL
- Leverages multiple LLMs for enhanced capabilities

AI Coding Assistants

- AI coding tools inherit all LLM limitations
- Excel on small, well-defined coding tasks
- Need complex instructions divided to small steps
- Exploit all gaps in rules and constraints
- Can report success even when code is broken
- Often fall into loops of repeating same mistakes
- Require extensive human supervision and review
- Can add and improve code only under guidance

Multi-Agent Architecture

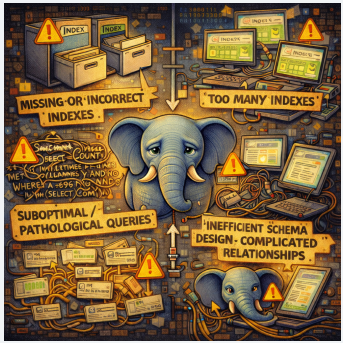
- Multi-Agent Architecture can mimic a team
  - "Detective" - Log Analyzer -> root cause analysis
  - "Architect" - Schema/Index Tuner -> recommendations
  - "Tester" - Testing Agent -> validate changes
  - "Safety Officer" - Policy Enforcer -> safety checks
- Human-in-the-Loop for oversight and approval
- "Autopilot with Oversight" model
- Agents propose plans, humans approve changes
- Incremental set of rules and safety requirements
- Independent safeguards with fallback scenario needed

The Streetlight Effect

- Currently we are like ”Nasreddin looking for lost keys”
- We are looking on well know places, because it’s easy
- But the keys were actually lost somewhere else - in the dark
- Optimizing relatively well known areas is not enough
- Such approaches merely prolong the status quo
- Yes, we can improve performance a bit further
- But we cannot reach any breakthrough this way

Main Known Performance Issues

- Missing indexes / incorrect indexes / too many indexes
- Inefficient schema design / complicated relationships
- Suboptimal queries / pathological queries -> in apps
- Many small queries / N+1 query problems -> in apps
- Connection libraries / ORM unexpected behavior
- Lack of proper partitioning / bad partitioning strategy
- Many short-lived connections / poor connection pooling
- Table bloat / lack of proper vacuuming and maintenance
- Monstrous HW with multiple NUMA nodes, remote memory access
- Insufficient hardware resources
- Poor configuration settings



Workload Related Issues

- Online Transaction Processing (OLTP)
  - High concurrency, many small transactions
  - -> Conservative settings to avoid resource exhaustion
  - -> e.g., low work_mem to prevent OOM under load
- Online Analytical Processing (OLAP)
  - Complex queries, large data scans
  - -> Aggressive settings to maximize throughput
  - -> e.g., high work_mem and parallelism for big joins
- Hybrid Transactional/Analytical Processing (HTAP)
  - Mix of both OLTP and OLAP characteristics
- Specialized workloads - time-series, geospatial, vector search
- Huge ETL jobs with bulk data loads

PostgreSQL Issues We Need To Talk About

credativ

Generic "cargo cult" recommendations ignore specific workload and data

Complex parameter interactions are not understood, many arbitrary settings

PostgreSQL is not in the best shape, with many weaknesses and missing features

AI/ML can help with understanding of issues and support improvements

Cargo Cult of Recommended Settings

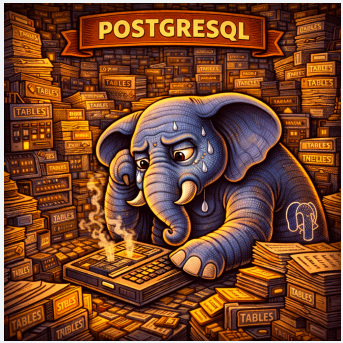
- Default PostgreSQL settings are NOT optimal
- Community provides recommendations ”good enough for most”
- Arbitrary numbers fitting ”average” workloads, HW, data
- Some are very old, but still ”the best we have”
- ”Fossils”, axioms, dogmas, ”cargo cult” practices
- Many settings are ”soft limits” or just guidelines
- We need MUCH BETTER understanding of PostgreSQL settings
- How to fine tune for specific workloads and hardware

Misterious Settings With Arbitrary Values

- Many settings are not understood at all
- Comments say ”measured on arbitrary scale”
- Strongly influence query planner decisions
  - cpu_tuple_cost = 0.01
  - cpu_index_tuple_cost = 0.005
  - cpu_operator_cost = 0.0025
  - parallel_setup_cost = 1000.0
  - parallel_tuple_cost = 0.1
  - mon_parallel_table_scan_size = 8MB
  - mon_parallel_index_scan_size = 512KB

Misterious Settings With Arbitrary Values

- These 2 parameters control costs over all code paths
- cpu_tuple_cost = 0.01
  - Estimated costs of processing each row during a query
- cpu_index_tuple_cost = 0.005
  - Estimated costs of processing each index entry during a query
- Settings ”measured on arbitrary scale”
  - -> Index entry 2x cheaper than table row - Ummmm, always ?
  - -> Table rows in all tables equally expensive - Wow, really ?
  - -> Index entries in all indexes equally expensive - Huh ?
- Maybe tunning on object level is needed here ?



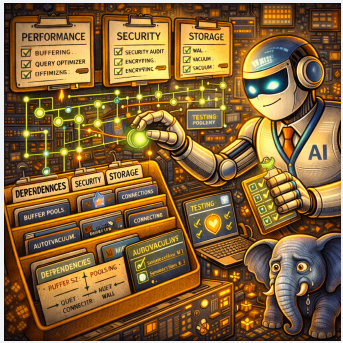
Dissecting PostgreSQL Settings

credativ

- Better tuning is not a trivial task at all
- Visible parameters can be checked via pg_settings view
- PostgreSQL code contains other hidden parameters
  - PG 9.5 - 253 params in view, 300+ with hidden params
  - PG 13 - 332 in view, 400+ total
  - PG 18 - 397 in view, 500+ total
- Growth reflects new features and finer control
- But also increased complexity of tuning & interactions

Categorizing PostgreSQL Settings

- Settings relevant for tuning
  - Logging & Monitoring settings: ~25 params
  - Query Planner/Optimizer settings: ~20
  - Write Ahead Log (WAL) & Checkpoints: ~15
  - Memory Usage: ~10
  - Autovacuum settings: ~10
  - Kernel & OS Resources: ~10
  - Lock Management & Concurrency: ~5
  - Replication & Recovery: ~10
- How many of them we really understand to tune them precisely?



PostgreSQL Weaknesses

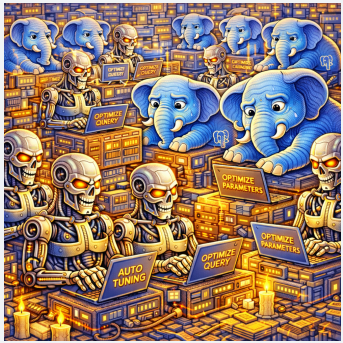
credativ

- PostgreSQL has known problems and limitations
- Community is cautious about big changes
  - 1 GB internal strings limit - too small
  - Is 8KB page size still relevant ?
  - only limited NUMA support
  - Query planner seems to be not fully understood
  - Clustered indexes, Global indexes on partitions?
  - TOAST storage overhead - can we do better?
  - Transaction ID Wraparound Risk - 64-bit XIDs?
  - Session-based architecture forever?
- AI / ML can significantly help here
  - Analyze complicated code paths
  - Suggest improvements in code base
  - Generate test cases and run them

Vision of Auto Tuning Tools

credativ

- Multi-Agentic System Architecture
  - "System Observer Agent": database metrics, logs, query plans, OS stats
  - "Application Agent": focuses on application-level patterns
  - "Analyzer Agent": identifies bottlenecks, patterns, anomalies
  - "Tuning Agent": suggests configuration, schema, query changes
  - "Validator Agent": tests changes in safe environment
  - "Executor Agent": applies validated changes to production
  - "Learner Agent": learns from past tuning successes/failures
- RAG (Retrieval Augmented Generation) vector storage:
  - Up-to-date best practices, community knowledge
  - Internal documentation, architecture diagrams
- Safety Mechanisms:
  - Rollback plans for any changes
  - Staged deployments (canary releases)
- Human-in-the-Loop must oversee and approve critical changes



Preserving Community Knowledge for AI Tuning

credativ

- Community has deep expertise in PostgreSQL tuning
- But much of this knowledge is tacit, not documented
- Knowledge not shared is lost
- We must collect this knowledge systematically
- This knowledge could enhance AI tuning tools dramatically
- Necessary for RAG part of AI tuning tools
- Best strategy? Format? Anonymization?
- Maybe exchange of knowledge for access to AI tuning tools?

Data Imagineering Future For DBAs

- Term ”Imagineering” was coined by Walt Disney
- Combining imagination and engineering to create innovations
- In data context - creatively designing, managing, utilizing data
- Encompasses data modeling, architecture, visualization, storytelling
- Goal: Transform raw data into meaningful insights, engaging narratives
- Blending technical skills with creativity to solve complex data challenges
- Less experts solving more complex problems with data

Thank you for your attention!

credativ

All my slides

Recorded talks