dbtune

A framework for autonomous databases

Marc Linster and Luigi Nardi
PGDay Paris
26 March 2026

Agenda

- A perspective on autonomy and agentic AI
- Lifecycle stages for a database
- Society of Automotive Engineers and the autonomous database
- Success factors for autonomous systems
- Focus areas for successful autonomous databases
- Current autonomous Postgres projects and capabilities

"Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed."

– Pattie Maes, MIT Media Lab, in Communications of the ACM, 1995

A perspective on achieving autonomy

Autonomy requires

A) Understanding of the desired outcome

B) A self-correcting, feedback-driven, process of

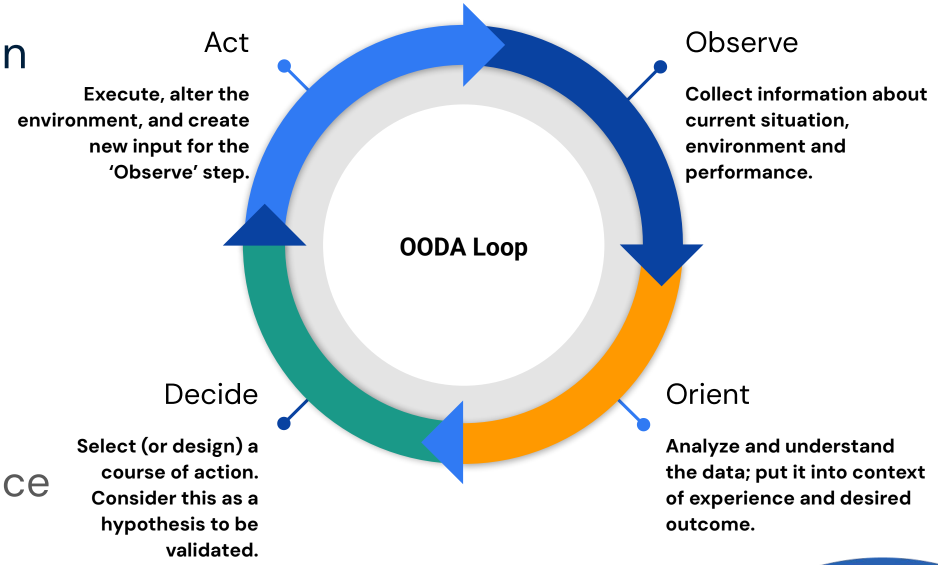
   1. Collection and analysis of information
   2. Decision on how to act to get closer to the desired outcome
   3. Execution of the decision
   4. Observation of the impact

The continuous loop of observe/act, in combination with a desired goal, is key

- Automation alone is not enough
- Observation and analysis are starting points

The *OODA Loop*: A model for goal-directed agile behaviour in complex environments

- Recognized model for data-driven decision making
- Provides a model to differentiate between solutions that
  - Observe and analyze
  - Automate
  - Provide autonomous intelligence

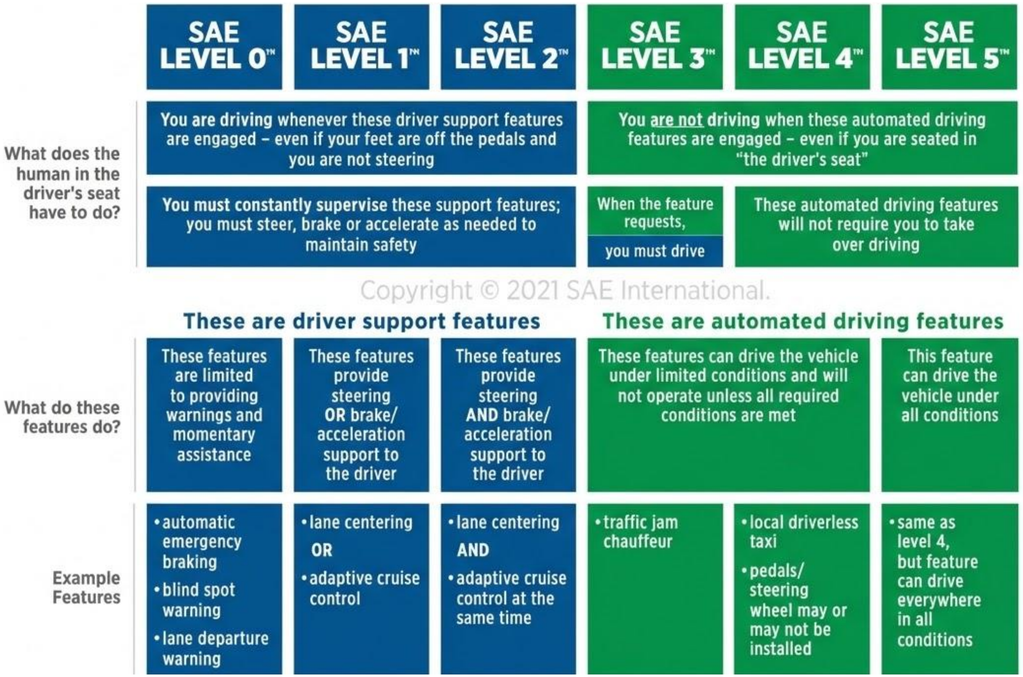


Self-driving cars — an autonomous databases analogy

- Self-driving cars are AI agents. They
  - Act on behalf of a company
  - Get instructions from a user
  - Use AI (ML, LLMs, World Models, ...) to execute the instructions optimally
  - Obey rules (rules of the road, driving style, ...)

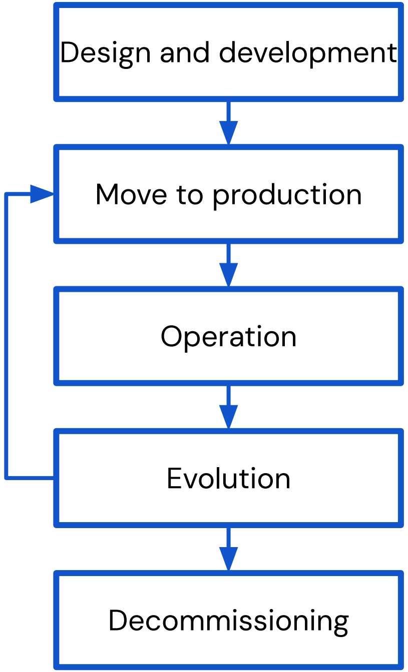
- Comparable to autonomous databases, which
  - Store and manage the company business
  - Execute queries for a user
  - Use AI to "operate" the database
  - Obey rules (no restart, don't modify the schema, ...)

Society of Automotive Engineers autonomy model



- **Blue**: Supports the driver
  - Individual feature focus
  - Hands on steering wheel
- **Green**: Controls the car
  - Automation in narrow situations (e.g., traffic jam), otherwise revert to 'support mode'
  - Automation in well-understood situations (e.g., driving in San Francisco in good weather), otherwise stop to secure
  - Permanent automation

Database lifecycle model



Requirements analysis, Méthode Merise, DDD; Normalization, ...

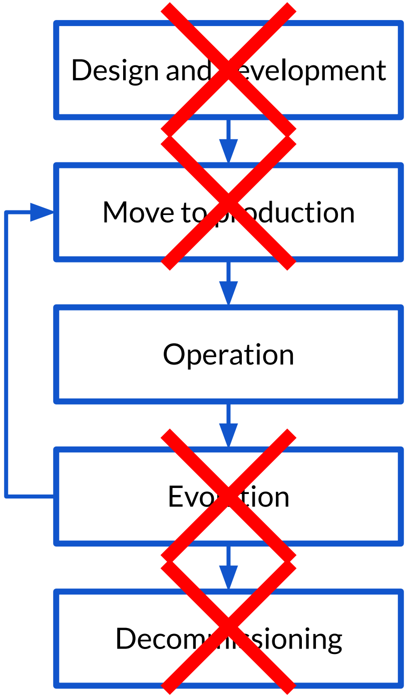
Infrastructure integration, HA, DR, Authentication, Monitoring, ...

Day-to-day operation, performance management, ...

Integration of new features, modernization, replatforming, restructuring, ...

Archiving, deleting, ...

Autonomous cars and autonomous databases



Autonomous cars focus on going from point A to point B, effectively, efficiently and safely in designated areas

They don't:

- Design themselves
- Deploy to production areas
- Modify themselves
- Scrap themselves

… If weather conditions deteriorate to the point that we believe it would affect the safe operation of our cars, Waymo vehicles are ***designed*** *to come to a safe stop until conditions improve — just like a motorcyclist might stop under an overpass in heavy rain …*
https://waymo.com/blog/2019/08/waymo-and-weather

Comparing database lifecycle and SAE model

Database lifecycle stage	**Car lifecycle**	**Automation/AI/Autonomy opportunity in database context**	**Database internal building blocks**
Planning, design, development	New car development process	LLM-supported data modeling; query and index design; intelligent data model review	DDL, DML, TCL, DCL
Move to production	Delivery to customer; commissioning; initial start-up	Hardware dimensioning; initial OS and server parameter setup; HA & DR setup	GUCs
Operation	Driving; basic maintenance	Non-disruptive optimization (indexes, OS & server parameters); patching; failover; auto-scaling; self-securing	GUCs; DDL (index, partitions)
Evolution	Modification	Alignment of new requirements with existing capability	DDL, DML, TCL, DCL

Comparing database lifecycle and SAE model

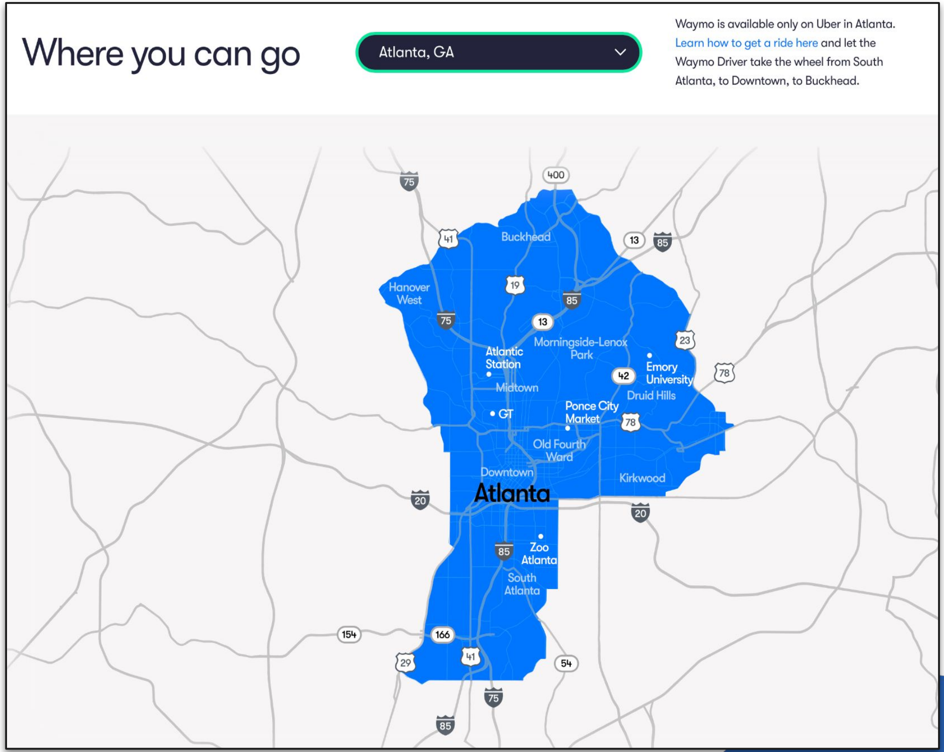
SAE standards focus

Database lifecycle stage	**Car lifecycle**	**Automation/AI/Autonomy opportunity in database context**	**Database internal building blocks**
Planning, design, development	New car development process	LLM-supported data modeling; query and index design; intelligent data model review	DDL, DML, TCL, DCL
Move to production	Delivery to customer; commissioning; initial start-up	Hardware dimensioning; initial OS and server parameter setup; HA & DR setup	GUCs
Operation	Driving; basic maintenance	Non-disruptive optimization (indexes, OS & server parameters); patching; failover; auto-scaling; self-securing	GUCs; DDL (index, partitions)
Evolution	Modification	Alignment of new requirements with existing capability	DDL, DML, TCL, DCL

Key success factor: Clear scope

- Self-driving cars are 'autonomous' within a specific scope
  - City/geo
  - Weather
  - All systems are go (no flat tires, …)

- Databases can use a similar model: Autonomy during 'normal' operation. Excludes, for example:
  - Schema optimization
  - Corruption repair
  - Integration of new capabilities
  - …



Key success factor: Componentized architecture

- Added Waymo Driver to mass produced vehicle

- Autonomous Databases
  - Add-on to proven platform
  - Multiple AI capabilities
    - Machine Learning
    - LLMs
    - Optimization
    - Neuro-symbolic AI
    - ...

- Multiple, task-specific AI capabilities, integrated with multiple sensors
  - Sensor Fusion Encoder (radar, lidar, cameras)
  - Driving VLM for complex semantic reasoning
  - Waymo's World Decoder for prediction of other road users behaviors

  https://waymo.com/blog/2020/03/designing-5th-generation-waymo-driver

Key success factor: Feature complete within scope

- Autonomous systems are not a loose assembly of disparate systems
  - Map
  - Radar
  - Lidar
  - Vision
  - Breaking model
  - Pedestrian avoidance model
  - ...
- To be safe, reliable, and usable by the greater public, it must be
  - Feature complete within the scope (e.g., San Francisco to San Jose)
  - Recognize its limits: When reaching limits, it hands over or transition to a safe state

Our perspective

- Initially, autonomy will come from a layer outside of Postgres
- Over time, Postgres will become more 'self-driving friendly'
  - More API-driven real-time visibility to the inner processes, e.g., which queries use which indexes on which table
- Eventually, proven autonomy components will migrate into the database
  - E.g., intelligent vacuum control
- This will be a gradual process
  - Cars started with speed controllers, lane-change warning, parking assistants, … until we got Level 4 autonomous cars
  - Waymo-style cars took 15+ years to be accepted
  - The same happened with automatic database failover!
- Making Postgres easier is critical to Postgres' success
  - We have outgrown the supply of qualified Postgres DBAs
  - AI-assisted Engineering/Vibe Coding creates databases faster than any DBA can support them

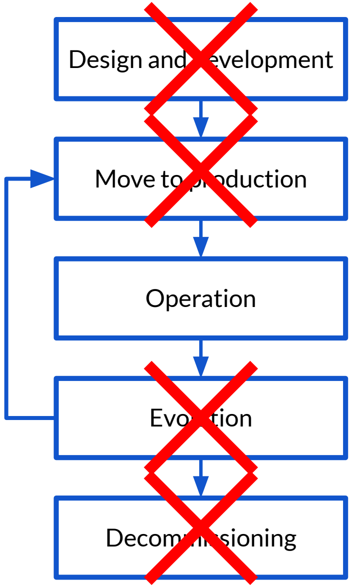
Autonomous database focus area

- **Goal**: Find lifecycle stages and situations where hands-off autonomy is possible in the near future (SAE Level 4)
- **Recommendation**: Operational phase (aligns with 'self-driving')
- **Key challenges**
  1) Adaptation to changing environment: users, queries, data volumes
  2) Deal with patches and upgrades
  3) Handle outages and failover
  4) Identify threats and defend
  5) Do it without additional functional or non-functional verification

Why focus on the operations phase

- Databases spend 95% of their time in this phase
- It’s the first time we see the real environmental conditions
- Goals (AQR, TPS, TPS/$, CPU, IO, …) can be clearly expressed
- The environment during Operations is readily observable
- Meaningful actions are ‘semantically safe’

  => GUCs, OS parameters, index tuning, partitioning, …



The question of trust and guardrails

- Autonomous databases need to be semantically safe
  - Can impact performance
    - AQR, latency, resource consumption, …
    - Strong and fast acting guardrails required!
  - Cannot impact outcomes
    - Results of queries change
    - GUCs that cause a restart to fail
- Actions need to be trustworthy
  - Based on data
  - Understandable and /or explainable
- Human in the loop
  - AI must be able to work hand-in-glove with DBAs
  - Establish trust and confidence
  - Gradual move from “AI advises” to “AI acts”

Current Postgres-related initiatives in industry

AlloyDB Autopilot	Vacuum, index and memory advice
AWS DevOps Guru	Anomaly detection (load, long-running queries, …)
Azure Postgres Autonomous Tuning	Index and statistics recommendations Vacuum and checkpoint optimization
CNPG	Self-healing: failure detection, failover, and redirection of traffic Patching: Automates patch application, but not autonomously Scaling: Automates addition of read-replicas, but not autonomously
DBtune	Agentic AI-based tuning of PostgreSQL GUCs
Dexter/HypoPG	Identifies slow queries; HypoPG for index candidates. Dry-run or auto-create
Ottertune (defunct)	ML-based tuning recommendations
pganalyze	Monitoring focus on performance; recommends indexes
PGTune	Templatized calculator for GUCs
Postgres.AI	Monitoring with LLM-focused data output; Integrates with LLM to suggest fixes
PoWA/pg_qualstats	Collect, aggregate, and analyze statistics

List excludes academic initiatives, e.g. Peloton or Cosine, and redirected ventures, e.g., CrystalDB

Current Postgres-related initiatives (OODA context)

	Observe	Orient	Decide	Act
AlloyDB Autopilot	✓	✓	✓	✓
AWS DevOps Guru	✓	✓	✓	
Azure Postgres AT	✓	✓	✓	✓
CNPG – Failover	✓			✓
CNPG – Patching				✓
CNPG – Scaling				✓
DBtune	✓	✓	✓	✓
Dexter/HypoPG	✓	✓	✓	✓ (grey)
pganalyze	✓	✓	✓ (grey)	
PGTune		✓	✓	
Postgres.AI			✓	
PoWA/pg_qualstats	✓	✓		

Level 4 autonomous Postgres databases: A near-term perspective

AI-augmented operations — requires 'hands on the wheel'

- Observation, analysis, and recommendation
  - Upgrades, updates — including advice on when to do it with least impact
  - Schema improvements
  - Partitioning and sharding
- Automation
  - Upgrades, updates
  - Partitioning & sharding

Autonomous operations — self-driving database under normal circumstances

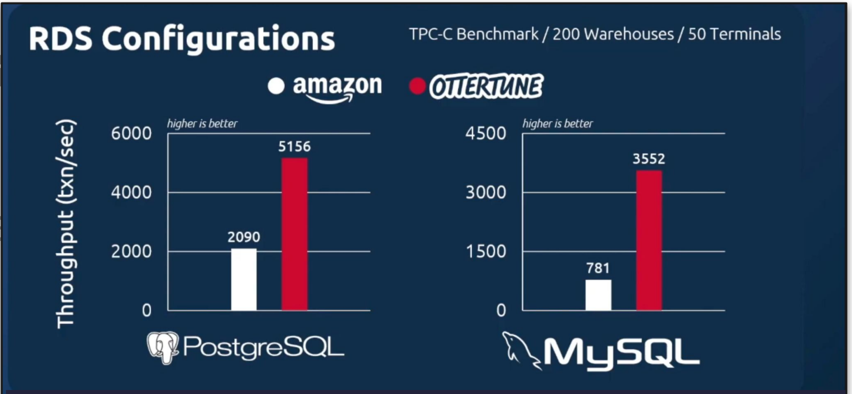
- Server parameter tuning
- OS parameter tuning
- Vacuum/bloat tuning, including recovery of free space
- Index tuning, including addition and removal
- Horizontal read scaling (up and down)
- Failover
- Securing (learning and controlling of safe/unsafe access patterns)

Why do we believe this is feasible?

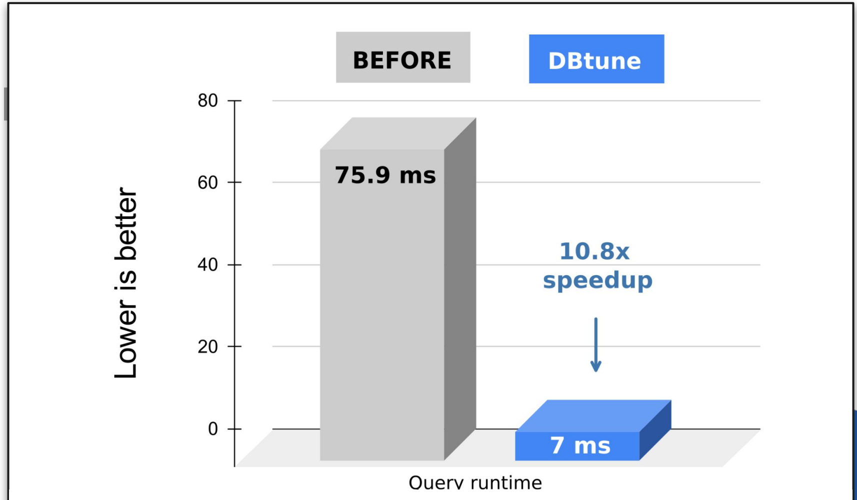
- Postgres APIs are providing more real-time visibility
  - pg_stat_statements
  - pg_stat_tuples
  - ...
- Sufficient compute power
  - The overhead for AI requests and commands has become negligible
  - AI/ML technology at scale is readily available
- We have proof points
  - Results
  - Technology

Why do we believe this is fea

- Postgres APIs are providing more re
  - pg_stat_statements
  - pg_stat_tuples
  - ...
- Sufficient compute power
  - Postgres supports the additional requests and commands
  - AI and ML technology at scale is
- We have proof points
  - Results
  - Technology



"You are Overpaying Jeff Bezos for Your Databases"; (OtterTune 2021)



"How Midwest Tape achieved a 10x performance boost with PostgreSQL tuning on AWS RDS" (DBtune 2026)

Summary

- Focus on the operations phase — just like self-driving cars
- Define guardrails and develop trust — just like self-driving cars
- Sensor data is key — just like self-driving cars
- Right balance between doing it inside the database and outside the database
- Integrated solutions are key — just like self-driving cars
- AI isn’t just LLMs — just like self-driving cars
- Defining the Autonomous Postgres is equal part database and AI — just like self-driving cars

Thank you

www.dbtune.com