Executive Profile Summary

Tom Maucotel · Data & Insight Analyst | Data Architect Toulouse, France · tom.maucotel@gmail.com · https://www.linkedin.com/in/maucoteltom/ · https://github.com/0-M-T

Short summary, for a profile header

Data analyst turned engineer. PostgreSQL, data governance, production deployment. Advanced practice of AI-assisted engineering.

Long summary, for an “about” section

Three years in a business intelligence team, working on a multi-instance PostgreSQL warehouse and a portfolio of several hundred reports. 181 requests handled, 81 % delivery rate.

My path followed a progression I own: first handling requests (business views, extracts, reports, migrations), then designing internal tools, working on data governance, and putting services into production.

What I built, and can discuss in depth:

- a DDL generator deriving all of its code from database introspection, exposed through three interfaces over a single core, which cut the time to publish an object by a factor of ten;
- connectors for an enterprise data catalogue, covering four families of sources with no native support, with column-level lineage;
- a BI platform monitoring service, taken over and industrialised, which brought failure detection down from several hours to a few minutes across roughly 470 datasets;
- a usage measurement algorithm over an estate of about 9 000 objects, with an explicit “unknown” state so that what could not be measured is never reported as unused;
- an AI-assisted development environment, with 33 encoded procedures, 12 automated hooks and an autonomous agent with adversarial verification.

Two principles that shape how I work: a task is not finished until a figure proves it, and no value is better than a wrong value presented as reliable.

The four flagship projects, problem and result

DDL generator for a reporting layer

Problem. Publishing a transformed object to the reporting layer meant creating two objects by hand, following naming conventions from memory, copying columns and types, and redefining grants. Ten to fifteen minutes per object, several times a week.

Action. A measured baseline first, then a tool deriving the entire DDL from introspection. Three interfaces (graphical, command line, tool server for an agent) over a single core with no duplicated logic. SQL safety at three levels. A dry-run mode faithful to the real execution.

Result. Under a minute per object, conventions guaranteed mechanically rather than remembered, grants validated before writing. 203 tests with no database, including regression tests on an injection flaw and on transaction rollback.

Data catalogue and end-to-end lineage

Problem. No visibility over the data lifecycle: nobody could answer “where does this come from, who consumes it, what breaks if I change it”.

Action. Took over a project started by a colleague, then maintained it (156 commits out of 233, all final merges). Built connectors for four families of sources with no native support. 12 documented fixes to an open-source connector, 9 of them suitable for upstream contribution.

Result. Upstream and downstream impact analysis across the whole chain, from source table to report, with column-level lineage over the relational scope.

BI platform monitoring, put into production

Problem. No native alerting on refresh failure. Outages were discovered by business users, days later.

Action. Took over a service started by a colleague (65 commits out of 97) and industrialised it: secret vault with a short-lived non-persistent secret, resilient HTTP client (rate limiter, circuit breaker, retries on transient errors only), metrics with label-cardinality discipline, three health probes, hardened container, pipeline wired to shared enterprise templates, manual approval before deployment.

Architectural decision I own. Expose metrics rather than write statuses into the warehouse: monitoring that depends on the system it watches cannot report that system's outage.

Result. Roughly 470 datasets monitored, success rate made measurable and therefore steerable, detection down from several hours to a few minutes.

AI-assisted development environment

Problem. An assistant used ad hoc produces uneven work, ignores conventions, and can touch sensitive systems.

Action. 33 encoded procedures (activation conditions, ordered steps, numeric thresholds, stopping conditions) rather than plain instructions. 12 automated hooks for safety and quality, including a secret-protection hook built as a security component in its own right, with its own test suite. 3 tool servers exposing business logic to an agent, defaulting to dry-run with server-side revalidation. An autonomous agent bounded by two independent barriers, with adversarial verification by a judge on a separate model that factually checks what the agent claims to have done.

Result. An environment where expensive rules are enforced by hooks rather than hoped for, and where background work progresses unsupervised with a check against fabricated claims. The configuration was packaged and shared with the team.

Vocabulary worth getting right

French-to-English mappings that are easy to get wrong, and matter in a technical interview.

French	English	Note
entrepôt de données	data warehouse	not "data store"
vue métier	business view	"business" carries the domain sense here
couche de restitution	reporting layer, serving layer	"restitution layer" is not idiomatic
non-régression	regression testing	the French keeps the negation, English does not
mise en production	production deployment, go-live	
astreinte	on-call rotation	
ordonnanceur	scheduler	not "orchestrator", which implies more
coffre-fort de secrets	secret vault, secrets manager	
disjoncteur	circuit breaker	
seau à jetons	token bucket	rate-limiting algorithm
doublure de test	test double	"mock" is only one kind of double
dette technique	technical debt	
analyse d'impact	impact analysis	
recette, validation métier	user acceptance testing	
jeu de données, modèle	dataset, semantic model	depends on the BI platform's own naming
demande, ticket	request, ticket, work item	