credativ

# Migrating Legacy & Proprietary Databases to PostgreSQL

## Digital sovereignty is now more critical than ever (version 3)

Josef Machytka <josef.machytka@credativ.de>

2026-05-05 - PostgreSQL Conference Belgium

# 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


credativ.de

# 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:*

# Table of contents

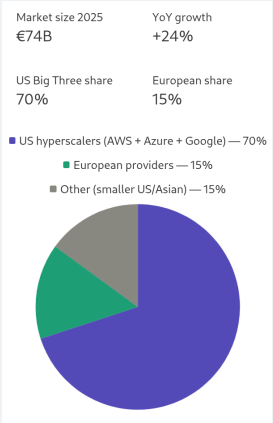
credativ

credativ

# Main Reasons for Migration

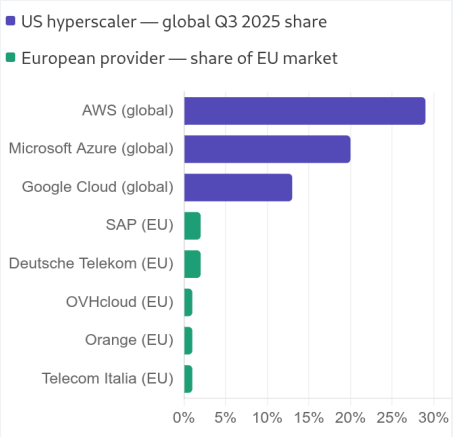
# European Digital Sovereignty

- Non-European hyperscalers control 70% of EU cloud market
- Reducing strategic dependencies is necessary
  - Adoption of open source software by EU institutions
  - Guaranteed exit options and data portability
  - Standardized solutions for monitoring, backup, HA
- Transparent, verifiable auditability of data
  - Be sure what is happening with your data all the time



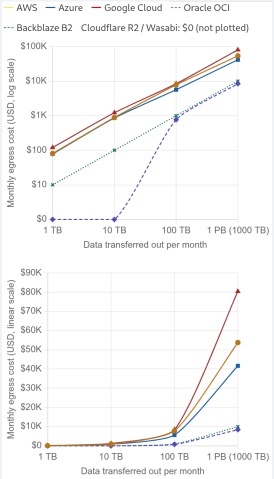
# EU Regulatory Drivers

- General Data Protection Regulation (GDPR)
  - Applies to any organization offering services to EU residents
  - Data Residency & Localization - data must be stored in EU
  - Legal protection from foreign surveillance laws
- U.S. CLOUD Act
  - Requires US providers to hand over data from their servers
  - Regardless of where data is physically stored in the world
  - EU companies can use "Bring Your Own Key" (BYOK) encryption
  - But quantum computers can break encryption in the near future



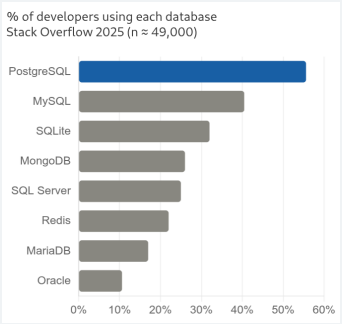
# Vendor Lock-in

- Financial aspects
  - Artificially high switching costs to prevent migrations
  - Egress fees - massive costs for moving data away
  - Should be removed by January 2027 by EU Data Act
  - Growing licensing payments for proprietary databases
  - "Yearly licensing costs higher than salaries of whole Eng team"
  - European capital being drained to the US
- Technical aspects
  - Most often databases dictate the architecture
  - Proprietary database usually leads to proprietary cloud
  - PostgreSQL allows to use any cloud provider
  - Moving from PostgreSQL to another PostgreSQL is much easier



# Aging of Legacy DBs & Hardware

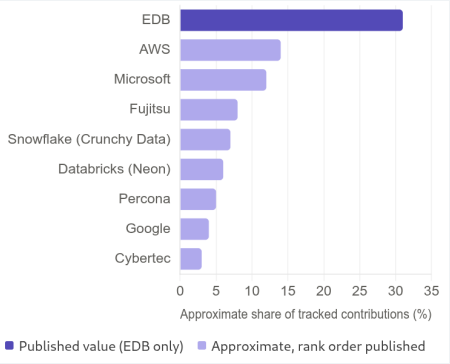
- Stagnant Innovation and Security
  - Lack of support and new features for legacy DBs
  - No plans for future development of some legacy DBs
  - Legacy engines being deprecated
  - Often run on old hardware / outdated OS
- Lack of administrators, shrinking community
  - Hard to find DBAs for legacy systems
  - Real life experience often buried in old discussion forums
  - Internet resources scarce or lost



# Big Companies invest in PostgreSQL

- Google lately published huge contribution to PostgreSQL code
  - Enhancements to logical replication
  - Active-Active Replication for multi-region write workloads
  - pg_upgrade Optimization for Large Objects
  - WAL Flush Logic Hardening
  - Parallel pg_dump

- All big hyperscalers invest heavily in PostgreSQL
- They all offer cloud PostgreSQL as a service
- Sell support and additional services
- They also employ many PostgreSQL core developers



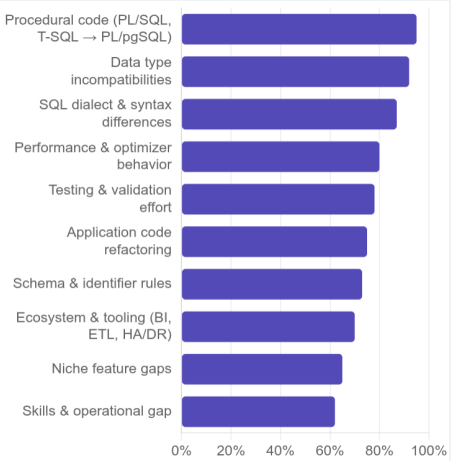
credativ

# Typical Migration Issues

# Typical Migration Issues

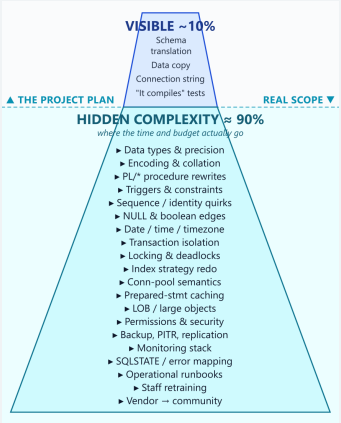
- Volume of data is usually not the main issue
  - More data usually means "just" more time to migrate
  - Not all data needs to be migrated in one go
  - Historical data can be moved later or even deleted
  - Old applications did not have data retention policies
  - Are very old data still needed for business?

- Customers often do not understand real problems
  - Usual requirement: "experience with more than x TB of data"
  - I personally migrated 150 TB - but it was a "simple migration"
  - The data model was clean, basic data types, no triggers



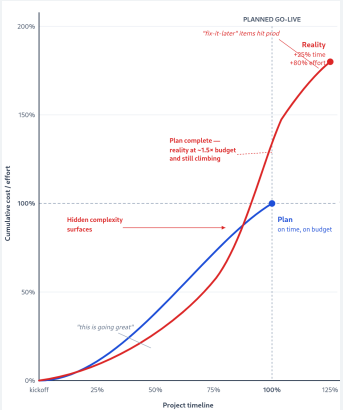
# Typical Migration Issues

- Real problems are lost knowledge & complexity
- Migration is not a simple ”lift and shift”
- Transfer of data is just a small part of migration
- DB-specific features: ROWIDs, query hints, data types
- Different SQL syntax: JOINs, hierarchical queries
- Lost knowledge: “History became legend. Legend became myth.”
- Dead objects in DB - not working code, renamed objects etc.
- Business logic hidden in DB objects



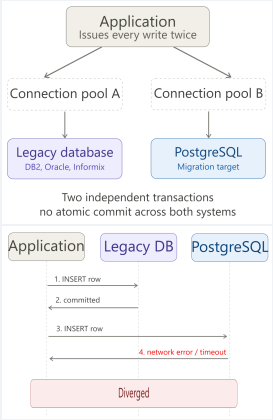
# Typical Migration Issues

- Insufficient analysis / planning / validation
  - Real complexity usually underestimated
  - Hoping that it will be easier than expected
  - "We will fix it later" - never happens
  - Estimates are often too optimistic
  - Budget and schedule often overrun
- Different landscape of tools
  - Proprietary commercial GUIs usually customized
  - Open source tools are not always sufficient
  - Mandatory upskilling / training of staff
  - Different behavior of connection libraries



# Dual-Write - Solution or Anti-Pattern?

- Dual write into both DBs is sometimes recommended
  - The application must write all changes into both DBs
  - Can work in very simple cases
  - Usually badly backfires in complex scenarios
  - And with complicated workload patterns
- Two independent connections into two DBs
  - Cannot be handled by connection poolers
  - Application must handle complex rollback logic
  - -> Race conditions, missed events, broken states
  - "Fire and forget" logic cannot maintain consistency
- CDC / reverse CDC if possible is the best option

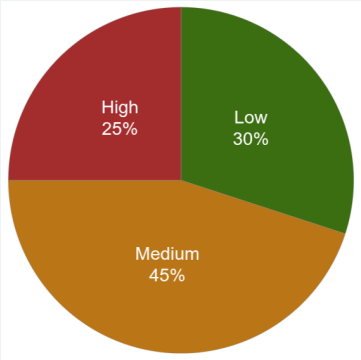


credativ

# Practical Decision Framework

# Complexity Estimation Matrix

- **Low Complexity:**
  - Standard data types, minimal procedural code
  - ORM-driven applications (Hibernate, Entity Framework)
  - Well-documented data models
- **Medium Complexity:**
  - Some proprietary types (e.g., XML, spatial)
  - Moderate use of triggers and stored procedures
  - App heavily relies on specific SQL dialect (e.g., T-SQL, PL/SQL)
- **High Complexity:**
  - Heavy business logic embedded in database
  - Cross-database queries, extensive use of DB links
  - Undocumented legacy systems (”Lost Knowledge”)
  - Missing original developers



# Strategy Selection: Offline vs Online vs Hybrid

credativ

- **Offline Migration (Dump & Restore)**
  - Best for: Smaller databases, high downtime tolerance
  - Pros: Simplest approach, lowest cost, easy to validate
  - Cons: Requires significant maintenance window
- **Online Migration (CDC)**
  - Best for: Mission-critical apps, large data volumes (TBs)
  - Pros: Close-to-zero downtime
  - Cons: High complexity, expensive CDC tools, complex validation
- **Hybrid Approach**
  - Best for: Legacy systems where CDC isn't possible (e.g., old Informix)
  - Pros: Balances cost and downtime, migrates historical data offline
  - Cons: Requires careful orchestration of cutover

# Online Migration - Change Data Capture (CDC)

credativ

- Point in Time snapshot of source DB, new data synced via CDC

- **Trigger-Based CDC - legacy approach**
  - triggers write changes to a shadow table or audit log
  - significant overhead on the source database
- **Query-Based (Timestamp/Polling) CDC**
  - periodical querying of source tables based on last_modified
  - struggles to track DELETE operations
- **Log-Based CDC**
  - standard for enterprise online migration
  - Reads source database transaction log asynchronously
  - Virtually zero additional overhead

# Risk & Tolerance Assessment

- **Downtime Tolerance**
  - Can the business survive a weekend outage? If yes, go offline.
  - If max downtime is 15 minutes, prepare for a complex CDC setup.
- **Data Loss Tolerance**
  - Is a loss of 5 minutes of transactions acceptable?
  - Influences backup strategies and cutover safety measures.
- **Performance Degradation Risk**
  - Initial post-migration performance might be worse.
  - Plan for extensive post-migration tuning and query optimization.
- **Reversibility Requirements**
  - Can we roll back easily?
  - Dual-write (if possible) or reverse CDC are costly but sometimes mandatory.

credativ

# Validation

# Pre-migration Steps - Migration Readiness Report

credativ

- Audit of objects - total count, sizes, count of lines of code
- Objects directly mapped to PostgreSQL with automated tools
- Objects requiring custom mapping / manual tweaks
  - -> Differences in syntax, feature gaps
- Objects without direct mapping
  - -> Special SQL syntax / procedural code / proprietary features
- Validation scope & Acceptance criteria
- Business constraints / Risk Appetite
  - Downtime tolerance, Data loss tolerance
  - Tolerance of Performance Degradation
  - Tolerance of Feature Discrepancies
  - Reversibility requirements

# Post-Migration Validation

credativ

- **Volume checks**
  - Row counts
  - External files sizes (if any)
- **Structural Checks**
  - Schema checks, data type checks
  - Index, constraints comparisons
- **Content Checks**
  - Checksums, Hashing Sampling
  - Check stats - sum(), avg(), min(), max(), count() per column
  - Count of NULLs per column
- **Application Checks**
  - Regression testing
  - Performance testing

credativ

# Experiences with Migrations

# IBM DB2

credativ

- First released in 1983, based on System R project
- System R was the source of SQL and relational DBs
- Licenses very expensive, especially for z/OS
- Pay for processors, for caches, for features, for support
- Runs on Linux, Unix, Windows (LUW), and mainframes (z/OS)
- LUW and z/OS are different products with specific features
- Syntax & Data types very close to PostgreSQL, closer than Oracle
- db2fce - a DB2 compatibility extension for PostgreSQL
- Implements objects like SYSIBM.SYSDUMMY1 - to avoid rewriting SQL

# IBM DB2 – Case Study

credativ

- Migration of 6 TB big database of clients and their payments
- Source - mainframe IBM z15, z/OS, DB2 v12
- Target - PostgreSQL on Debian
- Direct connection to DB2 impossible on current infrastructure
- IBM licensing is very strict, limits for z/OS CPU cores
- Online migration would require additional licensing for IBM software
- No open source simulator of IBM DB2 z/OS exists - only payed one


- Offline migration from DDL SQL export and CSV files
- Structures dumped using GEN tool, CSV using UNLOAD utility

# Informix Dynamic Server (IDS)


- Created by Informix Software in 1980s
- Acquired by IBM in 2001, development now by HCL
- Version 15 released in November 2024


- Syntax is closer to PostgreSQL than Oracle
- Conversion of data types and code very straightforward
- Migrator can convert most of the code automatically
- Can migrate data using UNL & clob*/blob* dump files
- Problems mostly with performance limits
- License can limit size of buffer, HW can be slow

# Informix - Case Study

- Informix database of 100,000s products - 2 TB data
- Big text descriptions in CLOB columns
- PDF leaflets in BLOB columns

- Very outdated hardware, performance limited by license
- Tests of direct batch migration were not successful
- Only very small batches of rows can be processed at a time
- Hybrid solution - metadata read from Informix
- Data imported from UNL & clob*/blob* dump files
- Migrator internally converts UNL dump to CSV
- Subsequent load with COPY command very quick

# Sybase Adaptive Server Enterprise (ASE)

credativ


- Created 1987 as Sybase, acquired by SAP in 2010
- SAP does not improve it anymore, sells SAP HANA as replacement
- Dominated telecommunications, banking, insurance
- Last version 16.1 (2022), no new features, only fixes
- Special ”Tabular Data Stream” (TDS) communication protocol
- Distinct Transact-SQL (T_SQL) dialect - similar to MS SQL Server

- Migrator converts data model & data with 100% success rate
- Problems only with mix of character sets in old data
- Proc with implicitly streamed data sets might need adjustments

# Sybase ASE - Case Study

credativ



- Multiple migrations of legacy systems in last year
- Sybase ASE 15 + 16, target PostgreSQL 16 + 17
- First cases just data models with data
- All logic embedded in applications
- Issues with multiple character sets in old data
- Conversion of charsets would require deep analysis of each record
- Client decide against conversion, would be very time consuming
- New migrations including stored procedures and triggers
- New TSQL parser works very well, but objects require manual adjustments
- Multiple implicitly exported data sets from functions
- Migrator extract structures of data sets from system tables

# Microsoft SQL Server

credativ


- Created as a fork of Sybase ASE in 1989, later rewritten
- T-SQL dialect of SQL, with special constructs
- Basic data types very similar, bigger limits
- Problems in migration similar to Sybase ASE
- Advanced features mostly replaced by PG extensions
- Typically many cross-database references

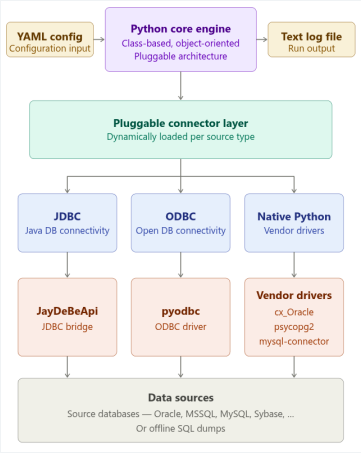
- Big migration currently in discussion
- Similar issues as Sybase migration

credativ

# Meet credativ-pg-migrator

# Meet credativ-pg-migrator

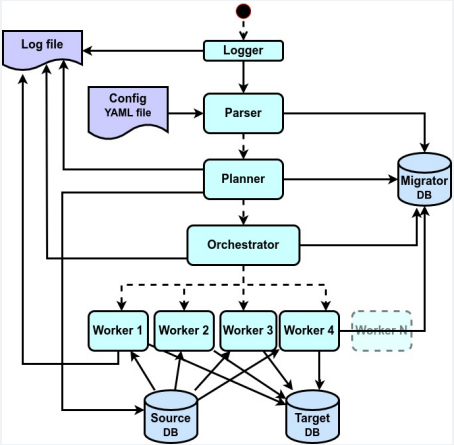
- Offline migrations from source DBs or dumps
- Written in Python for maximum compatibility
  - Modern languages have limited support for older DBs
  - Written in classes, dynamically pluggable connectors
- Uses well documented and stable libraries
  - Pyodbc, JayDeBeApi, cx_Oracle, psycopg2, mysql, etc.
  - Connectivity via JDBC, ODBC, or Python native DB access
  - YAML configuration file, text log file



# credativ-pg-migrator - Architecture

- Modular structure
  - Parser, Planner, Orchestrator, Workers
  - Runs parallel workers, one reader and writer per table
  - Checks row counts after migration of table

- Creates and fills rich migration protocol tables
  - Protocol tables contain all details about migrated objects
  - Outputs detailed INFO and DEBUG messages



# credativ-pg-migrator - Source DBs

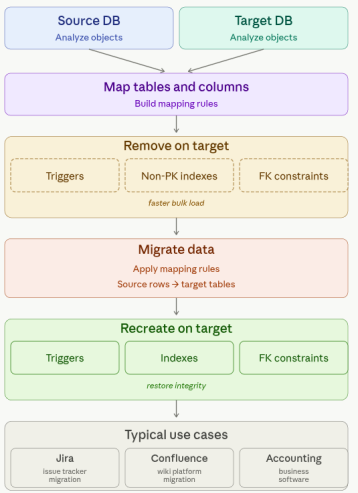
- Target database is always PostgreSQL
- Supports 8 different source databases
  - Oracle
  - Informix
  - MS SQL Server
  - Sybase ASE
  - SQL Anywhere
  - IBM DB2 z/OS
  - IBM DB2 LUW
  - MySQL/MariaDB
  - + PostgreSQL

# credativ-pg-migrator - Standard Workflow

- Reads metadata from source DB or SQL dump
- Converts to PostgreSQL format
- Creates target objects in PostgreSQL
- Migrates complete data models
- Tables, data, indexes, constraints, views
- Allows multiple custom adjustments

# credativ-pg-migrator - Mapping Workflow

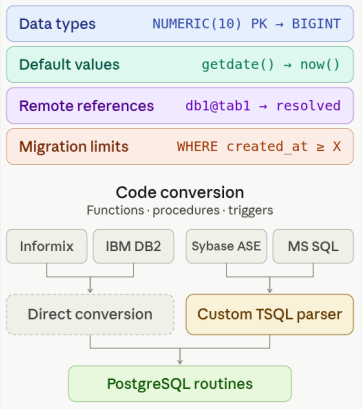
- Transfer of data between existing data models
- Analyzes objects in source and target DBs
- Maps tables and columns between DBs
- Removes triggers, non PK indexes and FK constraints
- Migrates data according to mapping rules
- Recreates triggers, indexes and FK constraints
- Examples: Jira, Confluence, accounting SW



# credativ-pg-migrator - Other Features

- Custom defined adjustments:
  - Replace Data types
  - Replace Default values
  - Substitutions of Remote objects references
  - Limitations for Data migrations

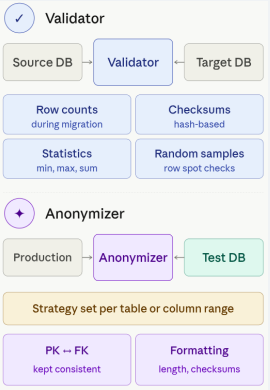
- Code conversion for some databases
  - Informix, IBM DB2 - direct conversion
  - Sybase ASE, MS SQL - custom TSQL parser
  - Migrator converts Functions, Procedures, Triggers
  - Success rate of code conversion 60 to 90%
  - Errors mostly due to missing tables, renamed objects/columns
  - Some statements may require manual adjustments



# Validator & Anonymizer

- Both currently under development
- Validator run of migrator
  - Migrator validates row counts during migration
  - Validator adds checksum-based data validation
  - Comparison of statistical results from tables
  - Direct comparisons of random data samples

- Anonymizer for GDPR sensitive data
  - Necessary for creating demo or testing databases
  - Implements multiple strategies for anonymization
  - Users select the method per range of tables / columns
  - Anonymizer ensures consistency of PK - FK
  - Preserves formatting / mathematical checks



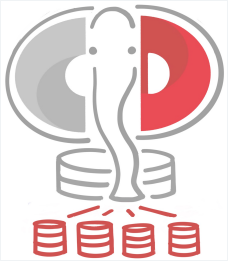
## Open Source Version 0.13.0

credativ

Released under the GNU General Public License, version 3 (or any later version)

GitHub repository - github.com/credativ/credativ-pg-migrator

PyPi - pypi.org/project/credativ-pg-migrator/



# Thank you for your attention!

credativ


Questions?