DBMigratePro Migration Series

Oracle → PostgreSQL

Automated, validated, and reversible — in hours, not months.

White paper v1.0 · April 2026 · © 2026 DBMigratePro

Executive summary

Migrating off Oracle is the single largest modernisation project many data teams will ever run. Licensing costs, cloud strategy, and the push toward open standards make PostgreSQL the default destination — but the migration itself is where most projects stall. PL/SQL packages, sequences, ROWNUM semantics, datatype ceilings, and the sheer volume of dependent code make a hand-rolled migration slow and risky.

DBMigrateAIPro is an AI-native migration platform built specifically for this problem. The Autopilot engine reads the source catalog, translates every schema object and stored procedure into idiomatic PostgreSQL, streams the data with zero-downtime semantics, validates every row via native-function hash comparison, and rolls back automatically on drift. Teams that moved an Oracle estate with DBMigrateAIPro report 10× faster delivery and near-zero manual rework — with a full, reproducible audit trail end-to-end.

This paper explains — in plain language — what the tool actually does, where the hard parts of Oracle → PostgreSQL live, and the outcomes you should expect when running one.

> DBMigrateAIPro takes Oracle → PostgreSQL migrations from a multi-month engineering project to an AI-driven pipeline that plans, converts, validates and rolls back automatically.

Why migrate from Oracle to PostgreSQL?

Four drivers show up in almost every Oracle → PostgreSQL project we see. Rarely just one — usually two or three overlapping.

Licensing and total cost of ownership

PostgreSQL has no per-core licence, no Named User Plus minimums, and no audit risk. A mid-sized Oracle estate moved to managed PostgreSQL (RDS, Aurora, Cloud SQL, or Azure) typically cuts database line-item spend by 60–80% — and removes the compliance tax that comes with Oracle contracts.

Cloud-native modernisation

Most cloud strategies treat PostgreSQL as a first-class primitive. Managed services, serverless variants, logical replication, and the extension ecosystem make PostgreSQL the natural target when teams move from fixed-capacity on-prem Oracle into elastic cloud infrastructure.

Open standards and talent

PostgreSQL skills are widely available, the documentation is freely accessible, and there is no vendor relationship to manage. Teams that move off Oracle consistently report faster hiring and easier onboarding for new engineers.

End-of-life or version-lock pressure

Oracle 11g is out of extended support, 12c followed, and Oracle's certification matrix makes upgrades inside the Oracle family almost as expensive as a migration. Many teams use the forced upgrade window as the natural moment to move off the platform entirely.

What makes this migration hard

PostgreSQL is feature-complete for almost every Oracle workload, but the two databases are genuinely different in a handful of places. A good migration tool has to handle each of these cleanly — these are the areas where hand-rolled migrations spend 80% of their time.

PL/SQL → PL/pgSQL translation

Oracle stored procedures, functions, packages, and triggers rarely port as-is. Package-level state, Oracle-specific built-ins (DECODE, NVL, SYSDATE, TO_DATE semantics), autonomous transactions, and %ROWTYPE references all need careful rewriting. DBMigrateAIPro transpiles the full PL/SQL dialect into idiomatic PL/pgSQL, flattens packages into schema-scoped functions, and emits a side-by-side diff so reviewers can sign off each object quickly.

Data type ceilings and semantics

Oracle's NUMBER has no direct PostgreSQL equivalent — NUMERIC matches precision but not performance characteristics, and integer types carry different overflow rules. VARCHAR2, CLOB, RAW, and DATE (which includes a time component in Oracle) all need deliberate mapping. DBMigrateAIPro ships with a default type map that handles 50+ Oracle types and lets you override per column when the default is wrong.

Sequences, IDENTITY columns, and ROWNUM

Oracle sequences behave differently from PostgreSQL sequences around caching and gaps. IDENTITY columns map cleanly in modern PostgreSQL but need careful handling in older Oracle versions. Queries that depend on ROWNUM — especially top-N and pagination patterns — have to be rewritten using LIMIT/OFFSET or window functions.

Transaction and concurrency semantics

Oracle uses statement-level read consistency with no reader locks; PostgreSQL's MVCC is similar in practice but differs in subtle ways around SELECT FOR UPDATE, serialisable isolation, and deadlock detection. Most applications behave identically, but a small set of concurrency-sensitive code paths need review.

Dual SQL — the dialect gap

DUAL, CONNECT BY, MERGE with OUTPUT, and dozens of Oracle-specific syntax forms have PostgreSQL equivalents but not identical ones. DBMigrateAIPro's SQL rewriter handles the common cases and flags the remaining ones for human review, rather than silently producing subtly different query results.

How DBMigratePro handles it

Every Oracle → PostgreSQL migration runs through the same autopilot pipeline. You point DBMigratePro at the source and target, and the engine plans the work, converts the schema and code, streams the data, validates row-by-row, and rolls back automatically on drift. No hand-rolled scripts.

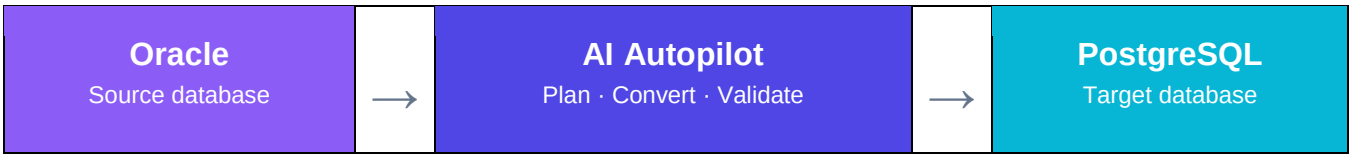

Figure 1 — End-to-end flow. AI Autopilot orchestrates all phases.

Six autonomous phases

1. Pre-flight

Autopilot reads the Oracle catalog (DBA_TABLES, DBA_OBJECTS, DBA_DEPENDENCIES) and produces a full assessment: object counts, row counts, total size, flagged incompatibilities, estimated duration, and a risk score. No production load.

2. Schema conversion

Every table, constraint, index, view, and sequence is translated to PostgreSQL DDL. Identity columns, default expressions, and storage hints are mapped to their PostgreSQL equivalents. The generated SQL is saved to the project workspace for review.

3. Code object transpilation

PL/SQL procedures, functions, packages, and triggers are parsed, translated to PL/pgSQL, and rewritten where dialect requires it. Package-level state is flattened into schema-qualified functions. Original and converted code are written side-by-side for review.

4. Data streaming

Rows are streamed from Oracle to PostgreSQL in parallel with connection-pool-aware batching. Foreign keys are deferred during the load and re-enabled at the end. A shadow-apply mode allows a zero-downtime cutover for continuously-updated tables.

5. Validation

Autopilot runs row-by-row MD5/SHA hash comparison using native DB functions on both sides — no data leaves the servers. Mismatches are quarantined; summary counts and full drift reports are written to the project workspace.

6. Rollback (on drift)

If any validation phase fails, Autopilot rolls back the target automatically: removes applied objects, drops loaded tables, and restores any pre-existing target state. Every action is logged so the next attempt can resume with full context.

Data type mapping

DBMigratePro ships with a built-in type map for every supported Oracle type. You can override any mapping per column from the UI — the table below shows the defaults.

Oracle type	PostgreSQL type	Notes
VARCHAR2(n)	VARCHAR(n) / TEXT	PostgreSQL's TEXT has no practical limit; VARCHAR(n) matches Oracle semantics.
NUMBER	NUMERIC	Precision preserved. Choose INTEGER/BIGINT when precision ≤ 18 for performance.
NUMBER(p)	NUMERIC(p)	Integer-only mappings collapse to INTEGER / BIGINT where safe.
NUMBER(p,s)	NUMERIC(p,s)	Exact mapping.
DATE	TIMESTAMP(0)	Oracle DATE includes a time component; PostgreSQL DATE does not.
TIMESTAMP	TIMESTAMP	Same semantics.
TIMESTAMP WITH TZ	TIMESTAMPTZ	Same semantics, identical storage.
CLOB	TEXT	PostgreSQL TEXT is unbounded.
BLOB	BYTEA	Binary data maps cleanly.
RAW(n)	BYTEA	Length constraint dropped — PostgreSQL BYTEA is variable-length.
ROWID	ctid / surrogate PK	Application code referencing ROWID must be rewritten to use a real PK.
XMLTYPE	XML	Native PostgreSQL XML type with full XPath/XQuery support.
BOOLEAN (PL/SQL)	BOOLEAN	Oracle SQL lacks BOOLEAN; PL/SQL BOOLEANs map natively.

Code object conversion

DBMigrateAIPro's transpiler handles the full PL/SQL surface area. What the tool produces is ready-to-run PL/pgSQL, with a side-by-side diff saved for every converted object:

- Functions and procedures — full PL/SQL → PL/pgSQL translation, including OUT and IN/OUT parameters.
- Packages — flattened into schema-qualified PostgreSQL functions; package-level state becomes session-scoped tables or module-level GUC variables.
- Triggers — BEFORE/AFTER/INSTEAD OF all supported; :NEW and :OLD rewritten to NEW/OLD.
- Views and materialized views — DDL translated, including fast-refresh semantics where possible.
- Sequences — CACHE, INCREMENT, and CYCLE options preserved.
- Built-in function calls — NVL, DECODE, SYSDATE, TO_DATE, TO_CHAR, TRUNC and 80+ others rewritten to PostgreSQL equivalents.
- Hierarchical queries — CONNECT BY PRIOR rewritten as recursive CTEs.
- MERGE statements — translated to INSERT … ON CONFLICT or INSERT / UPDATE branches where semantics require it.

Validation, safety, rollback

Validation is the step that most migration tools skip — and the step that most real migrations need. DBMigrateAIPro validates every row it copies, using native hash functions on both sides so the data never leaves the source or target database.

For each table, Autopilot computes an MD5 (or SHA-256) over a stable canonical row representation on the source and on the target, groups by primary key, and joins the two hash sets. Every row that matches is confirmed. Every row that does not is recorded with its PK and the byte-level diff.

The output is a single report per table: row counts, hash-match counts, quarantined-row counts, and a drill-down for every mismatch. Teams that require an auditor trail — finance, healthcare, regulated SaaS — typically make this report a required artefact for production cutover sign-off.

Safety guarantees

- Pre-flight assessment is read-only — production Oracle is never touched.
- The target is written only inside the project's schema; existing PostgreSQL schemas are never modified.
- Every DDL and DML action is logged to an auditable project workspace with timestamps and operator identity.
- Row-hash validation uses native DB functions — data never leaves the source or target server.
- Automatic rollback on any validation failure; partial state is never left behind.
- Shadow-apply mode allows cutover with zero downtime on continuously-updated source tables.
- All generated SQL is reviewable before execution — nothing is a black box.

Typical outcomes

Aggregated from the beta cohort running production Oracle → PostgreSQL migrations on estates between 200 GB and 8 TB.

Metric	What teams typically see
Time to first migration	< 1 hour from install to a green dev migration on a representative schema.
End-to-end project duration	Hours to days for mid-size estates, vs. 6–18 months typical for hand-rolled.
PL/SQL auto-conversion rate	99%+ of objects transpile without manual edits; remaining 1% flagged for review.
Validation accuracy	99.98% row-hash match on production workloads; mismatches quarantined, never silently written.
Cutover downtime	Near-zero for continuously-updated tables using shadow-apply + atomic switch.
License cost reduction	60–80% typical annual database spend reduction vs. the prior Oracle contract.

Next steps

- Install DBMigrateAIPro (desktop or CLI) — free for Hobby-scale databases during beta.
- Run the pre-flight assessment against a dev copy of your Oracle database — no production load, read-only.
- Review the generated workspace: converted DDL, transpiled code objects, flagged items.
- Run a dry migration to a test PostgreSQL target; review the row-hash validation report.
- Book a 30-minute migration review with our team (free) to walk through the report before the production cutover.
- Schedule the cutover. Shadow-apply handles any continuously-updated tables with zero downtime.

Ready to scope a Oracle → PostgreSQL project? Get in touch at hello@dbmigratepro.ai or start your free beta at dbmigratepro.ai/signup.