DBMigratePro Migration Series

PostgreSQL → BigQuery

Separate analytics from operations — serverless, petabyte-ready, ML-native.

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

Executive summary

BigQuery is Google Cloud's serverless data warehouse. Its per-query pricing, integration with Vertex AI and BigQuery ML, native support for GIS and JSON, and automatic scaling have made it a default analytics target for teams already on GCP (and increasingly for teams not on GCP at all).

Most PostgreSQL → BigQuery migrations are not replacements — they're replications. PostgreSQL stays as the operational database; BigQuery becomes the analytics layer where data scientists, analysts, and ML pipelines run their workloads without competing with production traffic.

DBMigrateAIPro automates the initial migration and flags what needs re-thinking (PG functions have no direct BigQuery equivalent; BigQuery's clustering and partitioning replace PG's B-tree indexes; standard SQL has subtle differences from PG SQL). This paper explains what the tool does, where PostgreSQL → BigQuery genuinely differs, and the outcomes to expect.

DBMigrateAIPro replicates PostgreSQL into BigQuery — converting schema, streaming data via Cloud Storage staging, and validating every row — so your analytics team gets elastic, serverless query while the operational database keeps serving the application.

Why migrate from PostgreSQL to BigQuery?

Teams move PostgreSQL analytical workloads to BigQuery for a consistent set of reasons.

Separate analytics from operational load

Running heavy analytical queries on the same PostgreSQL instance that serves the application is a recipe for latency spikes and contention. Replicating to BigQuery gives analysts a separate workspace where query time doesn't compete with production transactions — and compute scales automatically for the heaviest reports.

Serverless, per-query pricing

BigQuery has no servers to size, no cluster to keep running, no idle capacity to pay for. You pay per-query (bytes scanned) or for a flat-rate slot commitment. Teams with bursty analytics workloads often see substantial cost improvements vs. a fixed-capacity PostgreSQL DW.

ML and AI integration

BigQuery ML lets you train and run ML models with SQL. Vertex AI integration is native. For teams building analytical or predictive products on top of their data, the gap between storage and ML training is much smaller in BigQuery than in PostgreSQL.

Petabyte-scale without rearchitecture

BigQuery transparently handles datasets from GB to PB with the same interface. Teams that expect data growth to outstrip PostgreSQL's operational ceiling use BigQuery as the long-term analytics home while keeping PG for the operational layer.

What makes this migration hard

BigQuery is columnar, distributed, and serverless — a different paradigm from PostgreSQL. The tool handles mechanical translation and flags parts that need architectural review.

Standard SQL dialect differences

BigQuery uses Google Standard SQL. Most PostgreSQL SELECT queries port cleanly, but there are differences: BigQuery has no UPDATE/DELETE quotas on high-churn tables (it has DML quotas), array and struct types are first-class, regex syntax follows RE2, and some window functions differ. DBMigrateAIPro rewrites the common cases and flags the rest.

No indexes — partitioning and clustering

BigQuery has no user-managed B-tree indexes. Queries are fast because data is columnar and queries scan only needed columns. Partitioning (by date or integer range) and clustering (up to 4 columns) replace most PG index patterns. The tool proposes partition/cluster keys based on observed PG indexes.

Stored procedures and functions

BigQuery supports stored procedures (Standard SQL scripting) and JavaScript UDFs, but the surface is smaller than PL/pgSQL. Complex procedural logic often moves to an external orchestrator (dbt, Dataflow, Cloud Run) instead. The tool translates what maps cleanly and flags the rest.

Data types

PostgreSQL types map mostly cleanly — NUMERIC, TIMESTAMP, BOOLEAN, BYTES, STRING, DATE all have BigQuery equivalents. JSONB maps to BigQuery JSON (first-class since 2022). Arrays map to ARRAY. Custom PostgreSQL types (ENUMs, composite types, domains) are flattened to their base types.

Ongoing replication vs one-time load

Most PG → BigQuery migrations are ongoing. DBMigrateAIPro supports one-time bulk loads via GCS staging (fastest for initial migration) and incremental sync via Debezium-style CDC if you need near-real-time replication.

How DBMigratePro handles it

Every PostgreSQL → BigQuery 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.

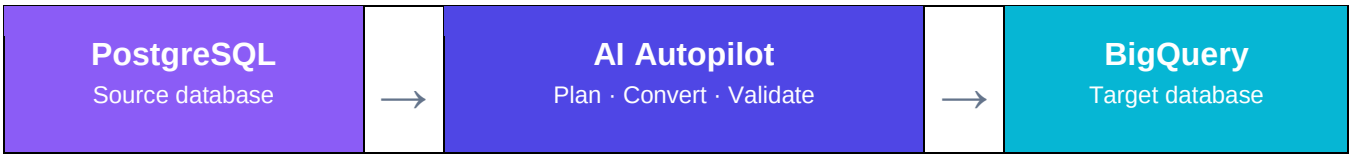

Figure 1 — End-to-end flow. AI Autopilot orchestrates all phases.

Six autonomous phases

1. Pre-flight

Reads pg_catalog, counts rows and objects, flags PG-specific features (custom types, extensions, complex PL/pgSQL) that won't port directly.

2. Schema conversion

Translates tables, constraints, and views to BigQuery DDL. Indexes become partition + clustering proposals. ENUMs and custom types flattened.

3. Code object transpilation

PL/pgSQL procedures translated to BigQuery Standard SQL scripting where possible; complex logic flagged for external rewrite.

4. Staged data copy

Rows exported to a GCS bucket in compressed columnar format, then LOAD JOB'd into BigQuery per table in parallel.

5. Validation

Row-by-row hash comparison using md5() on both sides. Mismatches quarantined with full diff detail.

6. Rollback (on drift)

Validation failure triggers automatic cleanup of the target dataset. All actions logged for resumable retries.

Data type mapping

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

PostgreSQL type	BigQuery type	Notes
BOOLEAN	BOOL	Same semantics.
SMALLINT / INTEGER / BIGINT	INT64	BigQuery has a single 64-bit integer type.
NUMERIC(p,s)	NUMERIC / BIGNUMERIC	NUMERIC for p≤38,s≤9; BIGNUMERIC for wider precision.
REAL / DOUBLE PRECISION	FLOAT64	Same semantics.
VARCHAR(n) / TEXT	STRING	BigQuery STRING is UTF-8, unbounded.
BYTEA	BYTES	Binary data maps cleanly.
DATE	DATE	Same semantics.
TIMESTAMP	DATETIME	No timezone storage in either.
TIMESTAMPTZ	TIMESTAMP	BigQuery TIMESTAMP stores UTC; semantics preserved.
TIME	TIME	Same semantics.
INTERVAL	INTERVAL	BigQuery INTERVAL is less general than PG INTERVAL; precision preserved for common cases.
UUID	STRING	BigQuery has no UUID type; stored as STRING.
JSON / JSONB	JSON	BigQuery JSON type is first-class.
ARRAY	ARRAY<T>	PG arrays map to BigQuery ARRAY with the element type.
ENUM	STRING + CHECK	Enumerated types flattened; application-level check.
Composite / ROW	STRUCT	PG composite types map to BigQuery STRUCT.
Geography (PostGIS)	GEOGRAPHY	BigQuery has native GEOGRAPHY with similar semantics to PostGIS.

Code object conversion

DBMigrateAIPro transpiles what maps cleanly into BigQuery scripting. Converted objects are saved side-by-side with originals:

- Procedures — PL/pgSQL → BigQuery Standard SQL scripting.
- Scalar functions — SQL functions translate cleanly; complex PL/pgSQL maps to JavaScript UDFs where needed.
- Triggers — BigQuery has no triggers; flagged for redesign as scheduled queries or external orchestration.
- Views and materialized views — DDL translated; BigQuery MVs have different refresh semantics.
- Built-in function calls — COALESCE, NOW, DATE_TRUNC, EXTRACT, SUBSTRING, ARRAY_AGG and 80+ others rewritten.
- Window functions — mostly pass through unchanged.

PostgreSQL → BigQuery

- CTEs pass through unchanged.
- Full-text search (tsvector) — flagged for redesign using BigQuery SEARCH or an external search system.
- Foreign data wrappers — not applicable; BigQuery's federated queries (external tables) are configured separately.

Validation, safety, rollback

DBMigrateAIPro validates every row it copies — using md5() on both sides so the data never leaves either platform.

For each table, Autopilot computes an MD5 over a stable canonical row representation on PostgreSQL and on BigQuery, then joins the two sets by primary key. Matches are confirmed; mismatches recorded with PK and byte-level diff.

The validation report — row counts, match counts, quarantined rows, per-mismatch detail — is a single artefact for sign-off.

Safety guarantees

- Pre-flight assessment is read-only — production PostgreSQL is never modified.
- The target is written only inside the project's BigQuery dataset.
- Every DDL and LOAD JOB is logged to an auditable project workspace.
- Row-hash validation uses native DB functions — data never leaves PostgreSQL or BigQuery.
- Automatic rollback on any validation failure; no partial state left behind.
- All generated SQL is reviewable before execution — nothing is a black box.
- GCS staging bucket is configured by you and credentials stay in your GCP project.

Typical outcomes

Aggregated from the beta cohort running production PostgreSQL → BigQuery replications on operational databases between 100 GB and 8 TB.

Metric	What teams typically see
Time to first replication	Hours from install to a green dev replication on a representative schema.
End-to-end project duration	Days to a few weeks for typical estates; CDC setup and analyst onboarding dominate.
PL/pgSQL auto-conversion	75%+ translate; the 25% balance usually moves to an external orchestrator (dbt, Dataflow).
Validation accuracy	99.99% row-hash match on production workloads.
Analytics query time	Heavy analytical queries typically run 5–50× faster on BigQuery vs. the same query on PG.
Operational database load	Analytics workload moved off PG; typical CPU reduction of 30–60% on the operational database.

Next steps

- Install DBMigrateAIPro (desktop or CLI) — free for Hobby-scale databases during beta.
- Set up a GCS staging bucket in your GCP project.
- Run the pre-flight assessment against a read-only PostgreSQL replica.
- Review the generated workspace: converted DDL, partition/cluster proposals, flagged items (triggers, tsvector, complex PL/pgSQL).
- Run a dry migration to a test BigQuery dataset; review the row-hash validation report.
- Book a 30-minute migration review with our team (free) to triage flagged items and plan incremental CDC.
- Schedule the initial bulk load; set up ongoing CDC replication to keep BigQuery fresh.

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