



MySQL[®] to Postgres[®]: A technical migration playbook

What the converter handles, what it leaves you,
and the behavioral differences that reach
production undetected.

What this playbook covers

For the DBAs, migration architects and application owners who will do the work. It answers three questions: what the tooling actually migrates, what it leaves as hand work, and which MySQL behaviors change on Postgres in ways that do not announce themselves.

There is no license cliff behind a MySQL migration, and this document makes no end-of-life or urgency argument. The business case is elsewhere. This is about scoping the work honestly.

What EDB supports, and what it does not

COMPONENT	MYSQL SUPPORT	NOTE
EDB Migration Toolkit	Yes: MySQL 5.5, 5.7, 8.0	Into community PostgreSQL 14–18 or EDB Postgres® AI (EDB PG AI) 14–18. Both targets are available, which makes the target choice a real decision.
EDB Migration Portal	No	Oracle sources only, per its supported-platforms list. Its AI Copilot is therefore also out of scope for MySQL.
EDB Replication Server	No	Single-master replication supports PostgreSQL, EDB PG AI, Oracle and SQL Server. MySQL is not listed.
EDB Data Migration Service	No	MySQL is absent from the source lists of both the LTS and innovation releases.

Source: EDB documentation for Migration Toolkit (supported platforms and databases), Migration Portal (supported platforms), Replication Server (introduction) and Data Migration Service (supported databases), checked October 2026.

The consequence, stated plainly

There is no first-party EDB change-data-capture path for MySQL. The supported mechanism is Migration Toolkit, online (directly into the target) or offline (generated scripts). Plan the cutover around a maintenance window. If the system genuinely cannot take one, that means a third-party CDC tool running alongside Migration Toolkit, and it changes the shape and cost of the project, so raise it in stage one rather than at cutover planning.

Five classes migrate. Everything else is engineering.

This is the most important table in the document. EDB Migration Toolkit's documented object matrix covers five object classes for MySQL sources.

OBJECT CLASS	MIGRATION TOOLKIT	NOTE
Schemas, tables, constraints, indexes, table data	Migrated	The relational skeleton and the rows. Largely mechanical.
Views	Hand work	Not in the MySQL object matrix. pgloader does not convert them either.
Triggers	Hand work	Not in the MySQL object matrix, and pgloader does not migrate them either.
Stored procedures and functions	Hand work	MySQL's procedural language and PL/pgSQL are not compatible. Each routine is a rewrite, or moves to the application layer. Usually the largest single line item in the estimate.
Partitioned tables	Hand work	Partition schemes are redesigned rather than ported.
Users, roles and grants	Hand work	Plan as a cutover task with its own security sign-off.
Sequences, object types, database links, queues	Not applicable / hand work	Not in the MySQL matrix.

Per EDB Migration Toolkit's documented object-migration matrix and pgloader's published MySQL limitations. For comparison, Oracle sources receive coverage across all of these classes; the MySQL matrix is genuinely narrower, and an estimate built on the Oracle shape will be wrong.

How to turn this into a number

Before anything else, count them. Views, triggers, stored procedures, functions, partition schemes. That count, multiplied by an agreed effort band per object class, is your migration estimate; everything else on this list is comparatively predictable. A project that has not produced that count has not been estimated, whatever the plan says.



The ones that do not announce themselves

Case-insensitive comparison: the silent failure

MySQL's common default collations compare strings case-insensitively, so `WHERE name = 'bob'` matches 'Bob'. Postgres text comparison is case-sensitive. After migration the same query returns zero rows and raises no error at all.

This is the single most dangerous difference on the list, because every other item on it fails loudly. Remediations: the `citext` type, an expression index on `lower(column)`, `ILIKE`, or a nondeterministic collation. Which one depends on how the column is queried, but finding every affected column is a stage-one inventory task, not a stage-three surprise.

Non-strict mode: why this surfaces late in the load

MySQL's non-strict SQL mode silently truncates over-length strings and coerces invalid values. Postgres rejects them. Every row MySQL quietly accepted over the years gets re-validated on load, which is why data migration commonly fails late rather than early. Budget a data-quality remediation pass; on an older estate it is not optional.

Implicit type coercion

MySQL compares a string value to an integer column without complaint; Postgres raises an error for most cross-type comparisons, such as an integer column against a text column or a string-typed parameter. These surface at runtime, not at schema validation, which is precisely why schema-only testing gives false confidence and why the application SQL inventory in section 06 matters more than it looks.

Storage engines

InnoDB is already a multi-version concurrency engine, so "Postgres has MVCC" is not a differentiator against an InnoDB estate and should not be used as one.

Where the engine does matter is MyISAM: no transactions, no foreign keys, so referential integrity was often enforced only in application code. Landing that schema in Postgres with real foreign keys will surface orphan rows.

Separately, MySQL's clustered-by-primary-key storage against Postgres's heap plus separate indexes changes which queries are fast. Index strategy has to be re-derived, not ported.



Construct by construct

MYSQL	POSTGRES	NOTE
AUTO_INCREMENT	GENERATED ALWAYS AS IDENTITY	Mechanical. SERIAL is the legacy form; identity is SQL-standard with cleaner ownership semantics. Reset sequences after load or the first insert is a duplicate-key error.
TINYINT(1)	boolean, or smallint	MySQL's de facto boolean. May genuinely be a boolean in one table and a small integer in another, so the mapping is a per-column decision.
Unsigned integers	Widen one step	No Postgres equivalent. tinyint to smallint, smallint to integer, mediumint to integer, integer to bigint.
ENUM	CREATE TYPE, or a CHECK constraint	Inline in MySQL, a named type in Postgres. See the open item in section 08 on enum support in the Oracle-compatible edition.
TEXT / MEDIUMTEXT / LONGTEXT	text	All collapse to one type.
BLOB family	bytea	
DATETIME / TIMESTAMP	timestamptz	Decide the timezone semantics deliberately rather than by default.
YEAR	integer	
TIME	time / interval	MySQL TIME can hold interval-style values beyond 24 hours, outside the Postgres TIME range. Documented to error on load.
Zero dates (0000-00-00)	Not representable	pgloader converts to NULL by default. A hand-rolled load needs a null-or-sentinel decision per column.
utf8 (utf8mb3)	UTF-8	A three-byte encoding that cannot store emoji or four-byte CJK; estates still on it may hold data that was already lossy. utf8mb4 maps cleanly. Collation names do not carry over; sort order changes unless ICU collations are chosen deliberately.
Backtick-quoted identifiers	Double-quoted, folded lowercase	MyTable becomes mytable unless quoted, and if you quote, every application query must quote too.

Mappings per PostgreSQL project documentation, pgloader's published casting rules and EDB Migration Toolkit documentation (FAQ, MySQL TIME). Some behaviors also depend on MySQL server configuration.



Concede it, then move the argument

pgloader is mature, free and good. A single command converts the schema, creates tables and indexes, migrates foreign keys, resets sequences and loads data in parallel. Its default casting rules already handle AUTO_INCREMENT, TINYINT(1) to boolean, zero dates to NULL, unsigned widening, ENUM conversion and index-name unification. EDB's own MySQL campaign material names it. There is no case for pretending otherwise.

	PGLOADER	EDB MIGRATION TOOLKIT
Schema, tables, indexes, FKs, data	Yes	Yes
Views	No	No
Triggers	No	No
Stored procedures and functions	No	No
Application SQL	No	No
Vendor support for the migrated database	No	Yes

pgloader offers paid support tiers for the tool itself, not for the target database. Sources: pgloader MySQL reference; pgloader.io/support; EDB Migration Toolkit documentation.

So the comparison is not tool versus tool

On schema objects the two stop in the same place, and the rows where both say No are the whole point. The converter was never the hard part.

What decides the timeline is the remediation backlog it leaves: every view, trigger and routine rewritten; every application query reclassified; and HA, backup and point-in-time recovery standing up before cutover rather than after. Neither tool converts stored procedures; treat any converter's claim to do so as a draft for engineering review, not a finished rewrite.

Classify the application SQL, early

After schema conversion, run an application SQL inventory and sort every statement into four buckets: portable as written; needs syntax changes; needs a data-model change; better replaced with a Postgres-native pattern. That classification is the application-side estimate, and it is the piece most often discovered late.



The five stages

The same five stages, in the same order, wherever code and schema have to be converted.

STAGE	WHAT HAPPENS	EXIT CRITERION
1 • Assess	Object counts by class; application SQL inventory; collation and case-sensitivity audit; MyISAM and data-quality survey.	A remediation backlog with effort per item.
2 • Transform	Schema conversion; views, triggers and routines rewritten.	Schema builds cleanly on target.
3 • Remediate	Behavioral differences, collation handling, constraint violations surfaced by strict validation.	No unresolved conversion or load exceptions.
4 • Move	Bulk load, reconcile row counts and checksums, reset sequences.	Reconciled data and working sequences.
5 • Validate	Functional and performance verification under representative load.	Signed-off test evidence.

Prove three things before go-live

- Backups complete.
- WAL archiving works.
- A full restore plus point-in-time recovery has been tested on the target: tested, not assumed.

Build the target as an HA cluster first and migrate into it, rather than retrofitting resilience afterward. Streaming replication with EDB Failover Manager or repmgr; backup with Barman or pgBackRest. The companion target-selection guide covers which topology fits.

Two things to confirm before you plan around them

Enum support in the Oracle-compatible edition

EDB's Migration Toolkit FAQ states that the Enterprise Postgres (Oracle Compatible) edition, the EDB server Migration Toolkit targets, doesn't currently support the enum data type, so MySQL ENUM columns move to CHECK constraints there. That page was last updated in May 2024.

MySQL ENUM columns are common, so if this still holds it materially affects that edition as a target. Confirm with EDB before it enters a design.

Existing binlog-driven pipelines

If anything downstream consumes the MySQL binary log (analytics feeds, search indexes, caches, event streams), there is no EDB-side consumer for it. Those pipelines need a replacement design on the target, typically logical replication, or a third-party CDC tool.

On some estates this is a larger scope item than the database migration itself, and it is routinely missed in the first estimate.

Where to start

1

Count the objects

Views, triggers, stored procedures, functions, partition schemes. This count is the estimate. Nothing else moves until it exists.

2

Audit collation and case sensitivity

Find every column compared as a case-insensitive string. This is the failure that reaches production silently.

3

Inventory the application SQL

Four buckets, per section 06.

4

Establish downtime tolerance

It decides whether the supported Migration Toolkit path fits or whether third-party CDC enters the design. Answer it in stage one.

Request an assessment

Go to enterprisedb.com and request a migration assessment, or contact your EDB account team. If you received this document from an EDB representative, reply to them directly.



Count the objects first. That count is the estimate.

Coverage and platform support reflect EDB documentation at time of writing. Verify them before you commit to a project plan.

Sources

- EDB Migration Toolkit documentation: supported platforms and databases; overview and object migration support; FAQ (enterprisedb.com/docs/migration_toolkit/latest).
- EDB Migration Portal, supported platforms (enterprisedb.com/docs/migration_portal/latest); EDB Replication Server, introduction (enterprisedb.com/docs/eprs/latest); EDB Data Migration Service, supported databases, innovation and LTS releases (enterprisedb.com/docs/edb-postgres-ai/latest/hybrid-manager).
- pgloader documentation: MySQL reference, default casting rules and published MySQL migration limitations (pgloader.readthedocs.io); support tiers (pgloader.io/support).
- PostgreSQL project documentation and community migration reference material on MySQL behavioral differences.

This playbook is engineering planning guidance and does not constitute a contractual commitment, a performance guarantee, or legal, audit or compliance advice. Object coverage and supported versions are per EDB product documentation at time of writing (checked October 2026); verify against current documentation before making a purchasing decision. MySQL and Oracle are registered trademarks of Oracle Corporation. Postgres and PostgreSQL are registered trademarks of the PostgreSQL Community Association of Canada and used with their permission. EDB Postgres® AI is a trademark of EnterpriseDB Corporation. © EnterpriseDB Corporation 2026. All rights reserved.

EDB Postgres® AI is the first open, enterprise-grade sovereign data and AI platform, with a secure, compliant, and fully scalable environment, on premises and across clouds. Supported by a global partner network, EDB Postgres AI unifies transactional, analytical, and AI workloads, enabling organizations to operationalize their data and LLMs where, when, and how they need them. For more information, visit enterprisedb.com.

