EDB POSTGRES AI

MySQL to PostgreSQL Migration

enterprisedb.com

Background

Given the explosive growth of the PostgreSQL database and the continual decline of the MySQL database, the question “Should we migrate?” has become “How should we migrate?”

Some of the many reasons users are leaving MySQL in favor of Postgres® are:

- Slowed development and support
- Deployment restrictions
- Security exposure
- Strict relational limits
- Limited cloud-native tools and integrations
- Difficulty in handling advanced workloads for analytics and AI
- Lack of robust high-availability options
- Enterprise-grade management scalability
- Limited data types

Postgres, on the other hand, offers the speed, security, and flexibility you need to support critical workloads in today’s evolving landscape. This document provides a guide for migrating from MySQL to Postgres. Discussion topics will include best practices for planning your journey and a scalable, repeatable framework for success.

This document is part of a series designed to help you determine proper timing, plan resources, assess the level of effort, identify risks/mitigation, address high availability, and address security concerns. To learn more about using EDB Postgres AI to move beyond the limitations of MySQL, refer to our website for more details.

Strategy overview

This framework can be used for any database migration, but we will focus on migrating from MySQL to PostgreSQL in the later stages:

1. Design your future state architecture.
2. Assess your existing MySQL database.
3. Create an equivalently structured PostgreSQL database.
4. Migrate DML data stores.
5. Validate database integrity.

Design your future-state architecture

Modern architecture considerations

With EDB Postgres AI as your single Postgres platform, you can upgrade your legacy architecture components to a more modern, feature-rich environment. The benefits include, but are not limited, to the following:

- Cloud-native, container, and Kubernetes implementations
- Advanced high-availability options
- Hybrid multi-cloud implementation platforms
- Enterprise-grade security tools
- Analytics accelerators
- AI pipelines and implementations
- Long-term backup and recovery
- Management and observability

This is the opportunity to plan a future-state architecture. Extreme care should be taken to not change too many components at once, to minimize risk during implementation.

Platform-agnostic considerations

One of the key advantages of PostgreSQL is that there are many deployment options available, allowing you to avoid dependence on any specific vendor or platform and thus maximize flexibility and availability. Some of the deployment models available are:

- AWS compute services
- Azure compute services
- GCP compute services
- On-prem bare metal
- On-prem virtual machine (hypervisor managed)
- On-prem private cloud container/Kubernetes managed
- Appliance-based compute models
- Hyper-converged compute systems
- Kubernetes service-based deployments (EKS, GKS, AKS, etc.)

With the correct deployment strategy and replication/high-availability tools, you can even create hybrid multi-cloud deployments for maximum availability and flexibility.

Assess your existing MySQL schema

Given that both MySQL and PostgreSQL are SQL-based databases, changes to coding or the data are minimized during the migration process.

The basic items to analyze in the existing MySQL database include:

- Data types
- Views
- Triggers
- Stored procedures
- Clustering
- Partitioning

Many publicly available documents help identify common differences between MySQL and PostgreSQL. One such document can be found on the EDB website.

In addition, tools are available to perform the analysis automatically. Some are publicly available on the postgresql.org community website.

Others that offer commercial support include the EDB Migration Toolkit.

Figure 1. The EDB Migration Toolkit

Any conflicts identified during the assessment should be addressed and corrected before creating an equivalent schema for your new PostgreSQL database.

Create an equivalently structured PostgreSQL database

PostgreSQL database creation can be achieved in several ways. Choose the method that best suits your deployment model.

Compute services

Create the appropriate compute, network, and storage services (local on-prem bare metal, on-prem VM, cloud compute services). Choose the proper PostgreSQL distribution and follow the installation guides. Documentation regarding PostgreSQL distributions and instructions can be found on the EDB website.

Cloud-native/Kubernetes services

Create the appropriate container/Kubernetes services (AKS, EKS, GKS, on-prem OpenShift, on-prem Rancher, etc.). Install the appropriate PostgreSQL distribution for Kubernetes. Documentation for the Kubernetes distribution of PostgreSQL is available on the EDB website.

If you are using Kubernetes distributions for your enterprise, consider EDB Postgres AI Hybrid Manager, which provides a Kubernetes-native control plane to automate, observe, and manage your entire Postgres estate.

Migrate schemas

Invoke the EDB Migration Toolkit with your MySQL source DB and Postgres target with the mysql specification per the example below.

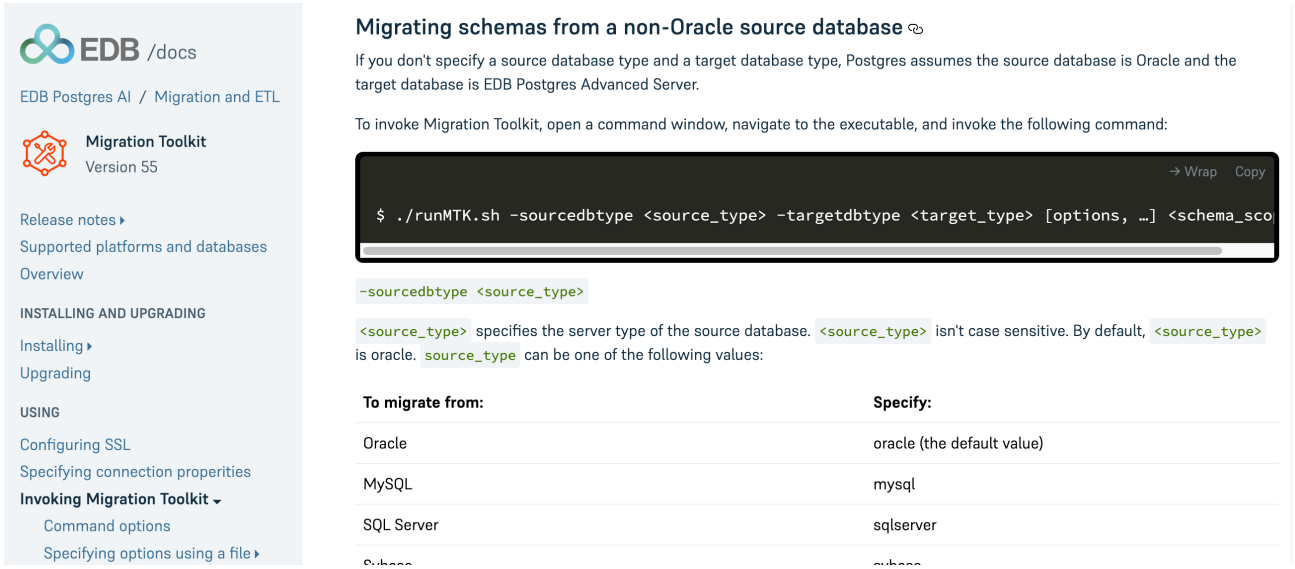

Figure 2. Example of migrating schemas with EDB Migration Toolkit

Migrate DML data stores

Now you are ready to migrate your data to the new PostgreSQL database. There are several different approaches you can take:

- **Online:** Migrate source data directly into the target PostgreSQL database for immediate use.
- **Offline:** Generate DDL/SQL scripts that can be reviewed, edited, and executed at a particular time.
- **Continuous:** Migrate chunks incrementally over a period of time.

A variety of tools can help you accomplish any of those approaches. Such tools include the commercially supported EDB Migration Toolkit or even the pgloader tool on the postgresql.org website.

Run an offline migration with all schemas.

Content of the options_textfile text file:

```
offlineMigration file_dest
allTables
schemaOnly
```

Syntax of the migration command:

```
./runMTK.sh -optionsFile options_textfile -allSchemas
```

Command line equivalent:

```
./runMTK.sh -offlineMigration file_dest -allTables -schemaOnly -allSchemas
```

Figure 3. Example of migrating DML with EDB Migration Toolkit

Validate database integrity

The final stage is to validate that the migration was a success. There are two main areas to validate:

- **Data integrity:** Test the data's integrity after conversion. There are many strategies and tools available for post migration data integrity.
- **SQL execution:** Evaluate SQL or application code against the new PostgreSQL database for validity.

Additional considerations

Post migration maintenance, management, automation, and observability should be integrated with existing infrastructure and business process.

Rollback/failover plan

It is always best practice to create a rollback plan so that you can revert back if necessary.

Supplemental resources and training

If you require more skilled or additional resources, professional services engagements are always available. Please contact EDB Professional Services for further information.

Complementary and purchased training is also available at the EDB Training Portal.

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.

© EnterpriseDB Corporation 2026. All rights reserved.