



erwin Data Modeler

Git Support

Release 12.5



Legal Notices

This Documentation, which includes embedded help systems and electronically distributed materials (hereinafter referred to as the “Documentation”), is for your informational purposes only and is subject to change or withdrawal by Quest Software, Inc and/or its affiliates at any time. This Documentation is proprietary information of Quest Software, Inc and/or its affiliates and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of Quest Software, Inc and/or its affiliates

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all Quest Software, Inc and/or its affiliates copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to Quest Software, Inc and/or its affiliates that all copies and partial copies of the Documentation have been returned to Quest Software, Inc and/or its affiliates or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, QUEST SOFTWARE, INC. PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL QUEST SOFTWARE, INC. BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF QUEST SOFTWARE, INC. IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is Quest Software, Inc and/or its affiliates.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2025 Quest Software, Inc and/or its affiliates All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contact erwin

Understanding your Support

Review [support maintenance programs and offerings](#).

Registering for Support

Access the [erwin support](#) site and register for product support.

Accessing Technical Support

For your convenience, erwin provides easy access to "One Stop" support for all editions of [erwin Data Modeler](#), and includes the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- erwin Support policies and guidelines
- Other helpful resources appropriate for your product

For information about other erwin products, visit <http://erwin.com/products>.

Provide Feedback

If you have comments or questions, or feedback about erwin product documentation, you can send a message to techpubs@erwin.com.

News and Events

Visit [News and Events](#) to get up-to-date news, announcements, and events. View video demos and read up on customer success stories and articles by industry experts.

Contents

Git Support	6
Connecting to Git Repositories	7
Troubleshooting	12
Committing Forward Engineering Scripts	14
Scenario 1: Committing New or Full FE Scripts	14
Scenario 2: Committing Alter Scripts	23

Git Support

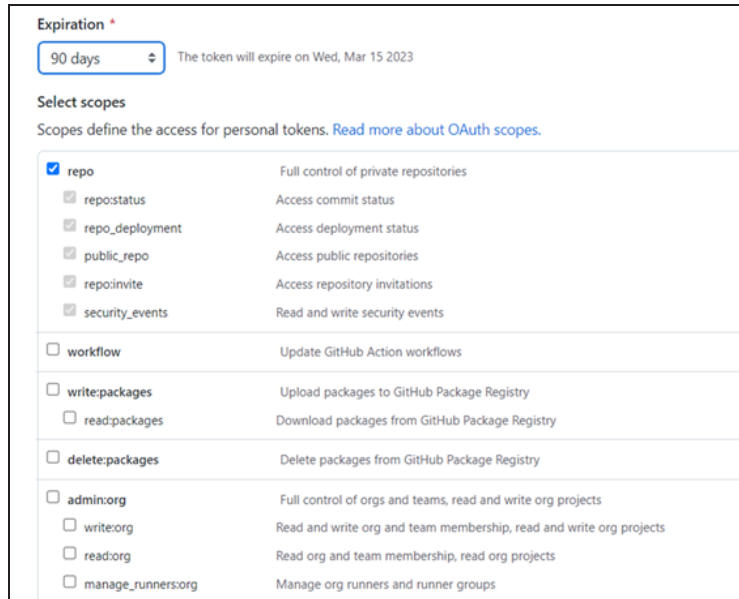
This guide walks you through the configuration and usage of Git via erwin Data Modeler.



Connecting to Git Repositories

A Git repository may be hosted on GitLab, GitHub, or Bitbucket. For a successful connection to these repositories, following are the prerequisites:

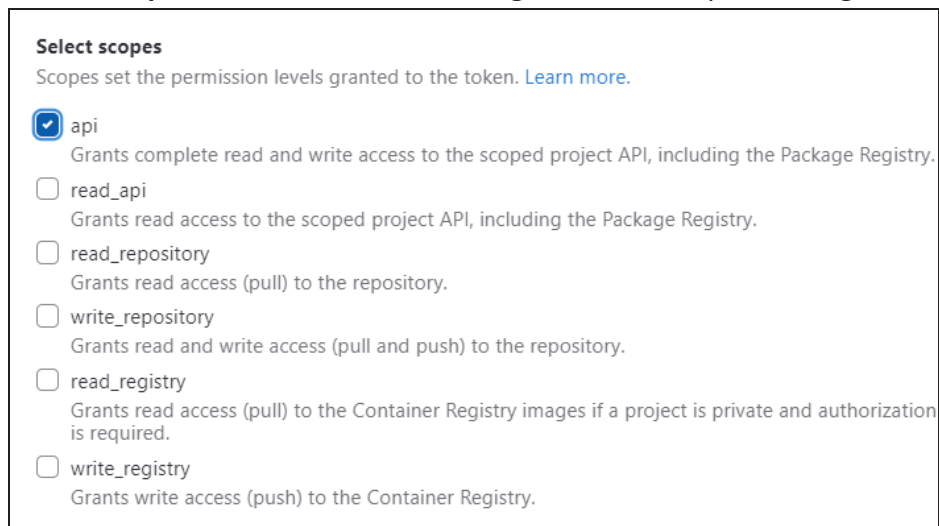
- **GitHub Scope:** Ensure that the following minimum scope is configured.



The screenshot shows the GitHub OAuth 'Select scopes' interface. At the top, there is an 'Expiration' dropdown set to '90 days' with a note 'The token will expire on Wed, Mar 15 2023'. Below this, a heading 'Select scopes' is followed by the text 'Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)'. The main area contains a list of scopes with checkboxes. The 'repo' scope is selected, and its sub-scopes are also checked. Other scopes like 'workflow', 'write:packages', 'delete:packages', and 'admin:org' are not selected.

Scope	Description
<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups

- **GitLab Scope:** Ensure that the following minimum scope is configured.



The screenshot shows the GitLab OAuth 'Select scopes' interface. It has a heading 'Select scopes' followed by the text 'Scopes set the permission levels granted to the token. [Learn more.](#)'. Below this is a list of scopes with checkboxes. The 'api' scope is selected, and its description is 'Grants complete read and write access to the scoped project API, including the Package Registry.' Other scopes like 'read_api', 'read_repository', 'write_repository', 'read_registry', and 'write_registry' are not selected.

Scope	Description
<input checked="" type="checkbox"/> api	Grants complete read and write access to the scoped project API, including the Package Registry.
<input type="checkbox"/> read_api	Grants read access to the scoped project API, including the Package Registry.
<input type="checkbox"/> read_repository	Grants read access (pull) to the repository.
<input type="checkbox"/> write_repository	Grants read and write access (pull and push) to the repository.
<input type="checkbox"/> read_registry	Grants read access (pull) to the Container Registry images if a project is private and authorization is required.
<input type="checkbox"/> write_registry	Grants write access (push) to the Container Registry.

Connecting to Git Repositories

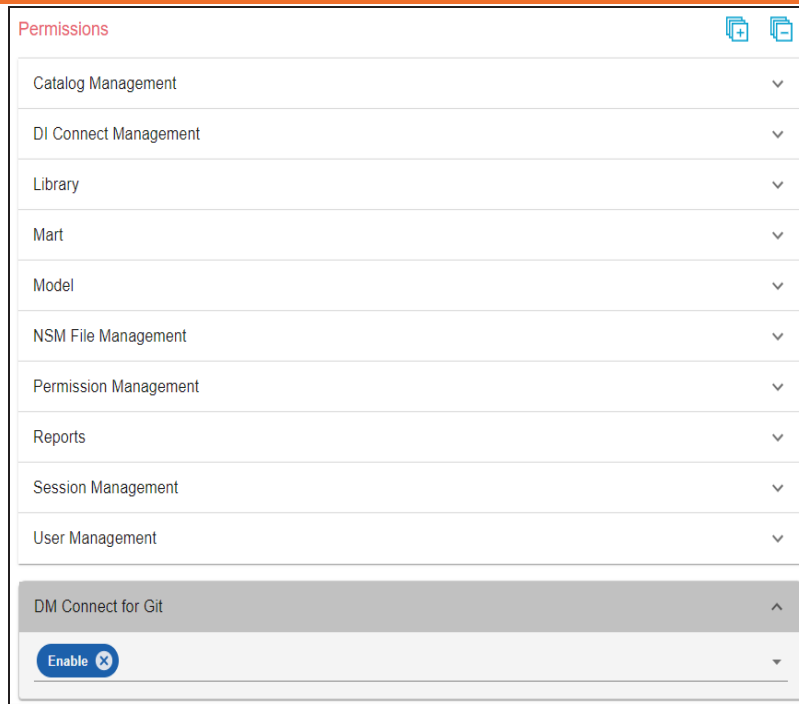
- **Bitbucket Scope:** Ensure that the following minimum scope is configured.

Scopes
[Learn more about scopes.](#)

Repositories	Pipelines
<input checked="" type="checkbox"/> Read	<input type="checkbox"/> Read
<input checked="" type="checkbox"/> Write	<input type="checkbox"/> Write
<input type="checkbox"/> Admin	<input type="checkbox"/> Edit variables
<input type="checkbox"/> Delete	
Pull requests	Runners
<input type="checkbox"/> Read	<input type="checkbox"/> Read
<input type="checkbox"/> Write	<input type="checkbox"/> Write
Webhooks	
<input type="checkbox"/> Read and write	

- **erwin Mart:** Ensure that,
 - erwin DM is connected to erwin Mart Portal. For more information on connecting erwin DM to Mart Portal, refer to the [Connect to Mart](#) topic.

Connecting to Git Repositories

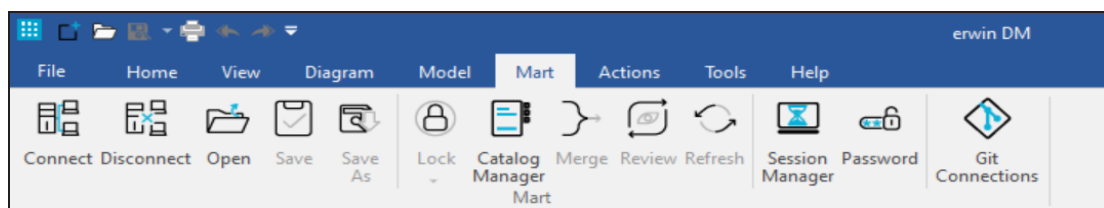


This permission is not available for other out-of-box profiles.

- **Personal Access Token:** Ensure that you have created the required personal access token. To know how to create personal access tokens for GitLab, refer to the GitLab documentation. To know how to create personal access tokens for GitHub, refer to the GitHub documentation.

Once, these prerequisites are in place, to connect Git repositories to erwin DM, follow these steps:

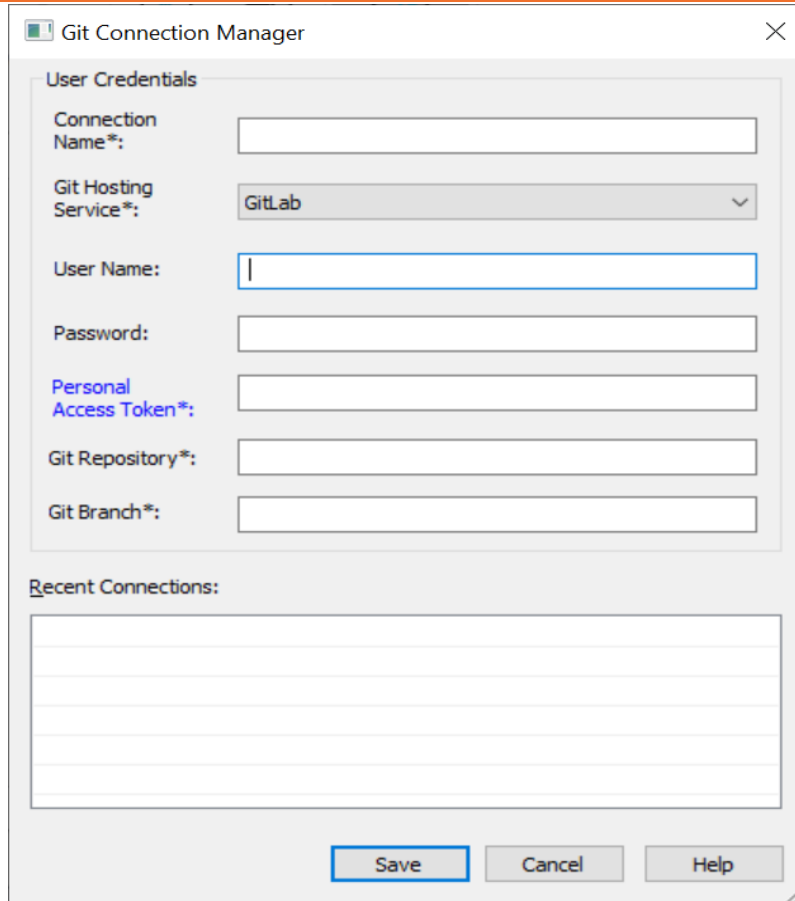
1. On the ribbon, click **Mart**.



2. Click **Git Connections**.

The Git Connection Manager page appears.

Connecting to Git Repositories



The screenshot shows the 'Git Connection Manager' dialog box. It has a title bar with a close button. The main area is divided into two sections. The top section, 'User Credentials', contains several input fields: 'Connection Name*' (a text box), 'Git Hosting Service*' (a dropdown menu with 'GitLab' selected), 'User Name:' (a text box with a cursor), 'Password:' (a text box), 'Personal Access Token*' (a text box), 'Git Repository*' (a text box), and 'Git Branch*' (a text box). The bottom section, 'Recent Connections:', contains a list box with several empty rows. At the bottom of the dialog are three buttons: 'Save' (highlighted with a blue border), 'Cancel', and 'Help'.

3. Enter appropriate values in the fields. Refer to the following table for field descriptions.

Field Name	Description	Additional Information
Connection Name	Specifies a user defined connection name	For example, ConnectGit. You can create multiple connections one for each Git repository.
Git Hosting Service	Specifies the Git hosting service to which erwin DM connects	GitLab : Indicates that erwin DM connects to GitLab GitHub : Indicates that erwin DM connects to GitHub

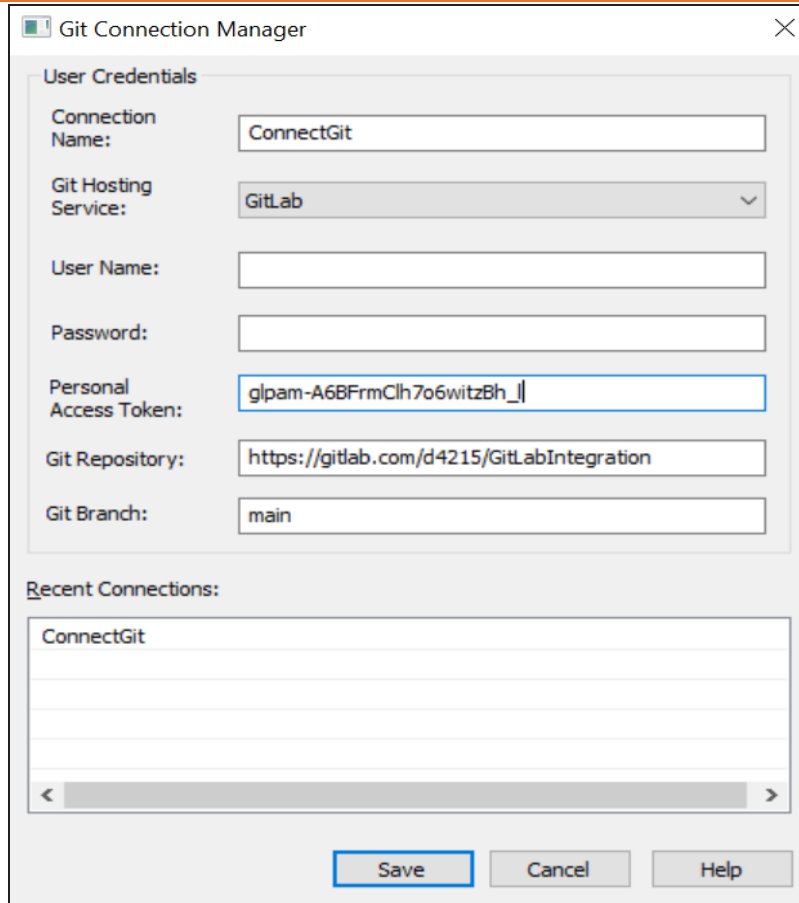
Connecting to Git Repositories

Field Name	Description	Additional Information
		Bitbucket: Indicates that erwin DM connects to Bitbucket
User Name	Specifies the username to log on to the Git hosting service	This field is not mandatory.
Password	Specifies the password to log on to the Git hosting service	This field is not mandatory.
Personal Access Token	Specifies the personal access token to connect to the Git hosting service	
Access Token	Specifies the personal access token to connect to the Bitbucket hosting service	This option is available only when the Git Hosting Service option is set to Bitbucket. Additionally, click the Access Token field label to learn about creating tokens and their types.
Git Repository	Specifies the URL of a Git repository where you want to push the forward engineering script	For example, https://gitlab.com/d4215/GitLabIntegration or https://github.com/poly-inc/poly-main-MCL
Git Branch	Specifies the branch that is used to push the forward engineering script	For example, main.

4. Click **Save**.

On successful connection, the connection name appears under Recent Connections.

Connecting to Git Repositories



The Git Connection Manager dialog box is used to configure a connection to a Git repository. It contains the following fields:

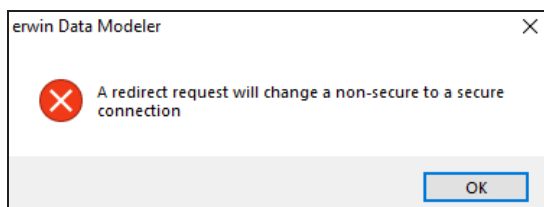
- Connection Name:** ConnectGit
- Git Hosting Service:** GitLab (selected from a dropdown menu)
- User Name:** (empty text field)
- Password:** (empty text field)
- Personal Access Token:** glpam-A6BFrmClh7o6witzBh_||
- Git Repository:** https://gitlab.com/d4215/GitLabIntegration
- Git Branch:** main

Below the fields is a section for **Recent Connections:** which contains a list with the connection name "ConnectGit".

At the bottom of the dialog are three buttons: **Save**, **Cancel**, and **Help**.

Troubleshooting

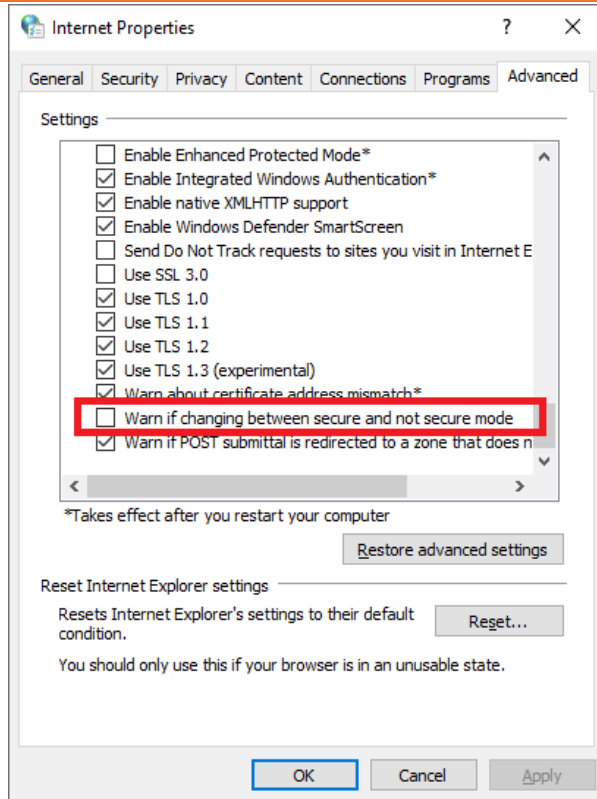
While setting up your connection, you may encounter the following error:



To resolve this error, follow these steps:

1. On your system, go to **Control Panel > Internet Options > Advance Tab**.
2. Clear the **Warn if changing between secure and not secure mode** check box.

Connecting to Git Repositories



3. Click **OK**.
 4. Close and reopen erwin DM.
 5. Connect erwin DM to Mart Portal.
 6. Launch the Git Connection Manager page and configure the Git connection.
- Once you are connected to a Git repository, you can [commit FE scripts](#).

Committing Forward Engineering Scripts

There are two scenarios in which you commit Forward Engineering (FE) scripts to a Git repository:

- **Scenario 1: Committing new or full FE scripts:**

Use the Forward Engineer Schema Generation Wizard to commit a physical database schema or FE script from a Mart Model.

To avoid script files from being overwritten, ensure that you use unique file names.

- **Scenario 2: Committing alter scripts:**

Use the Forward Engineer Alter Script Schema Generation Wizard to commit an alter script after you make changes to a Mart Model. You can commit an alter script in two ways:

- **Commit and append an alter script to an existing script file**
- **Commit and create a new alter script file in the Git repository**

For more information, refer to the Scenario 2: Committing Alter Scripts topic.

Scenario 1: Committing New or Full FE Scripts

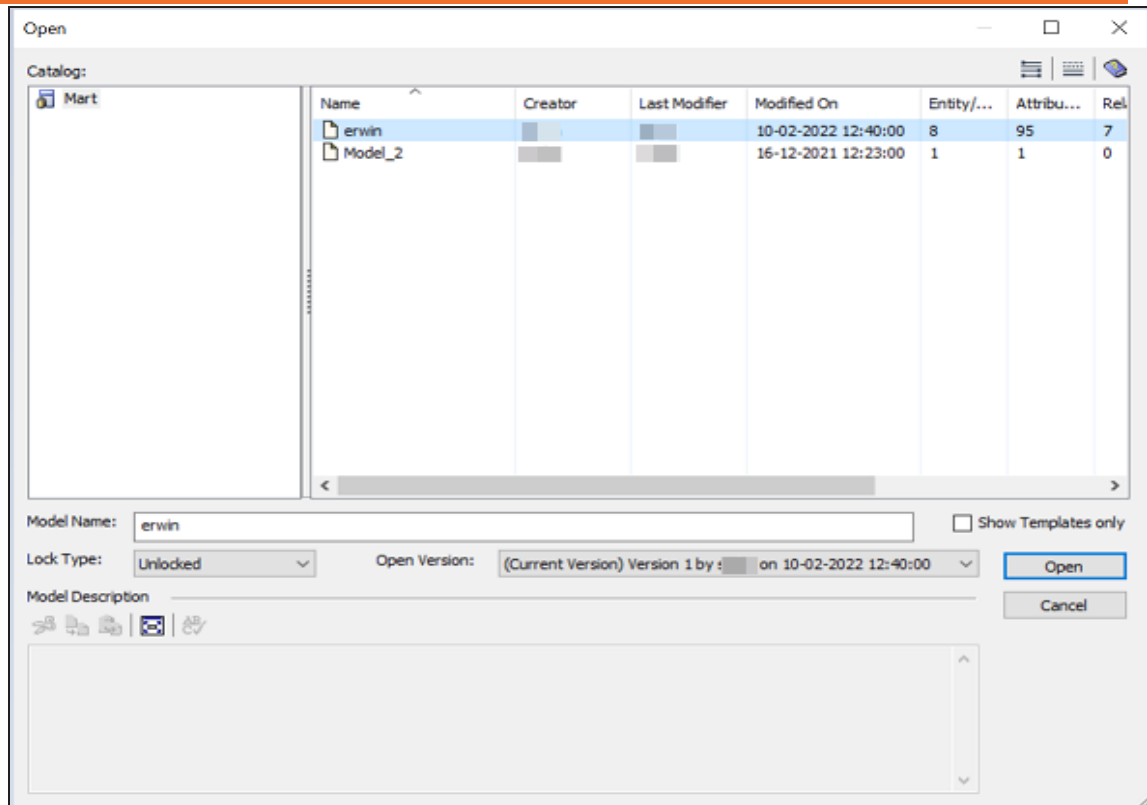
The Forward Engineer Schema Generation Wizard generates a physical database schema or Forward Engineering (FE) script. For a Mart Model, you can push the FE script to a Git repository.

To commit new or full FE scripts to Git repositories, follow these steps:

1. On the ribbon, go to **Mart > Open**.

The Open page appears.

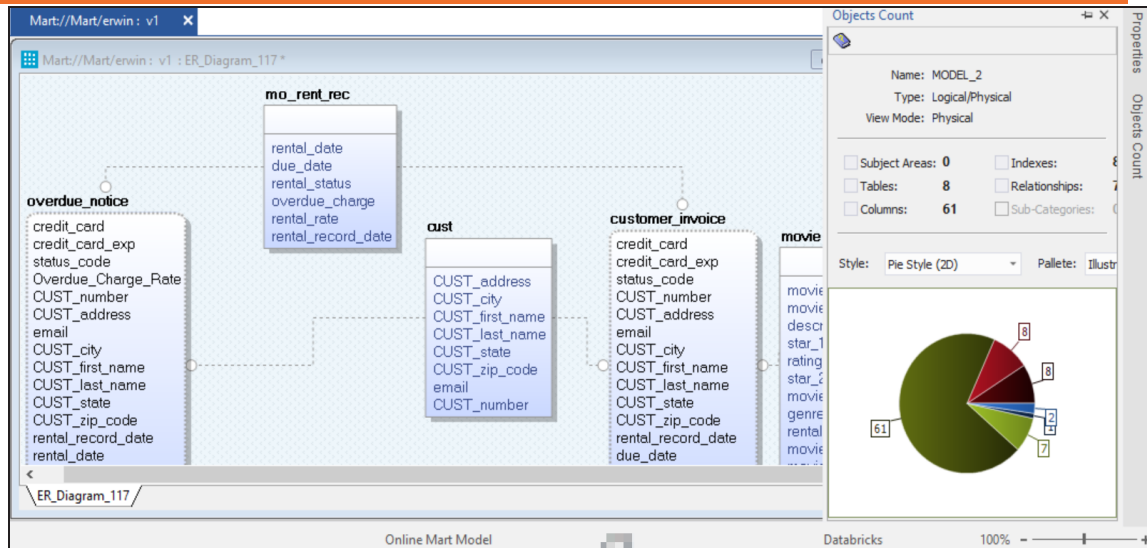
Committing Forward Engineering Scripts



2. Select a model, and then click **Open**.

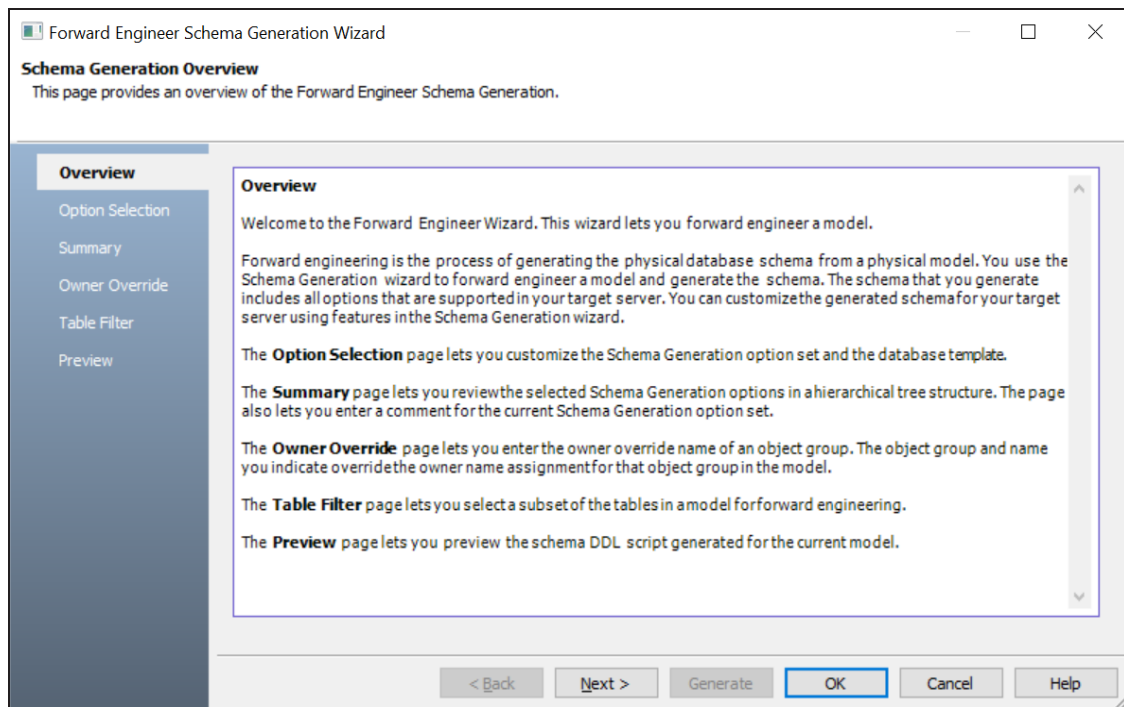
The Mart Model opens.

Committing Forward Engineering Scripts



3. Go to **Actions > Schema**.

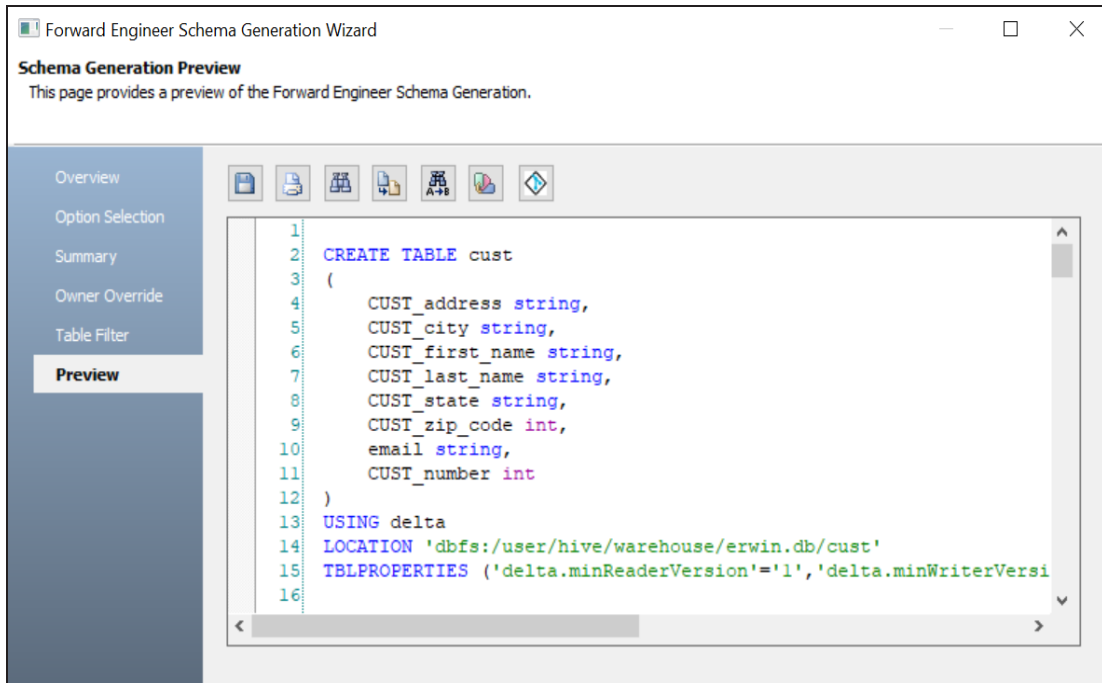
The Forward Engineer Schema Generation Wizard appears.



4. On the **Forward Engineer Schema Generation Wizard**, click the **Preview** section.

Committing Forward Engineering Scripts

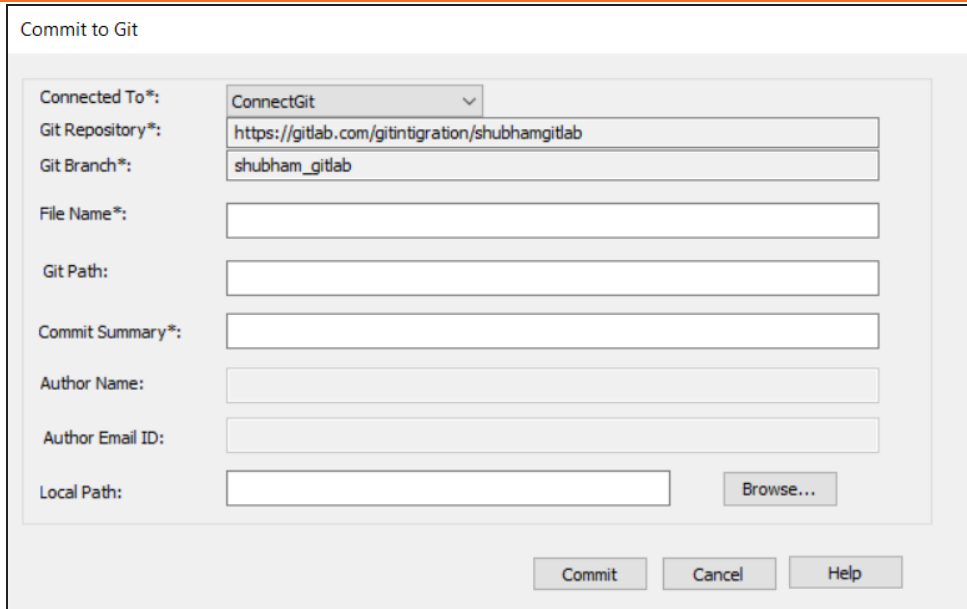
The FE script appears. For example, in the following image the Preview section displays FE script of a Databricks database. For more information on generating FE scripts, refer to the [Forward Engineering/Schema Generation for Databases](#) topic.



5. Click .

The Commit to Git screen appears.

Committing Forward Engineering Scripts



6. Enter appropriate values in the fields. Fields marked with an asterisk (*) are mandatory. Refer to the following table for field descriptions.

Field Name	Description	Additional Information
Connected To	Specifies the connection that connects erwin DM to a Git repository	For example, ConnectGit.
Git Repository	Specifies the Git repository configured for the connection	For example, https://-gitlab.com/d4215/GitLabIntegration is set for the ConnectGit connection. This field autopopulates based on the repository configured in the Git Connection Manager.
Git Branch	Specifies the Git branch configured for the connection	For example, main is set for the ConnectGit connection. This field autopopulates based on the repository configured in the Git Connection Manager.

Committing Forward Engineering Scripts

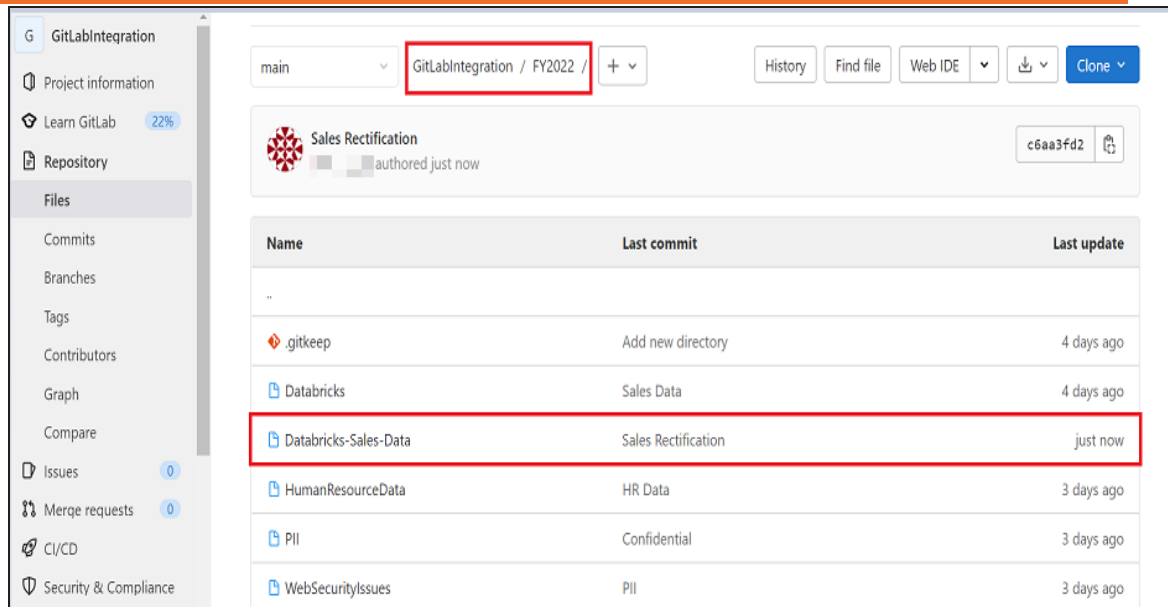
Field Name	Description	Additional Information
File Name	Specifies the user-defined name of the FE script file being committed to a Git repository	For example, Databricks-Sales-Data.sql To avoid script files from being overwritten, ensure that you use unique file names.
Git Path	Specifies the location in the Git repository where the FE script is committed	For example, FY2022/ The FE script is committed to the FY2022 folder inside the root folder of your Git repository.
Commit Summary	Specifies the summary of the push commit	For example, Sales Rectification.
Author Name	Specifies the name of the author pushes the commit	
Author Email ID	Specifies the email address of the author pushes the commit	
Local Path	Specifies the location on your local machine where the FE script is saved	C:\Users\SO\Documents\Databricks

7. Click **Commit**.

The FE script file is saved on the local path and committed to the Git repository.

For example, in the following image, FE script is committed to a GitLab repository in a file, Databricks-Sales-Data, with a commit summary, Sales Rectification using the main branch.

Committing Forward Engineering Scripts



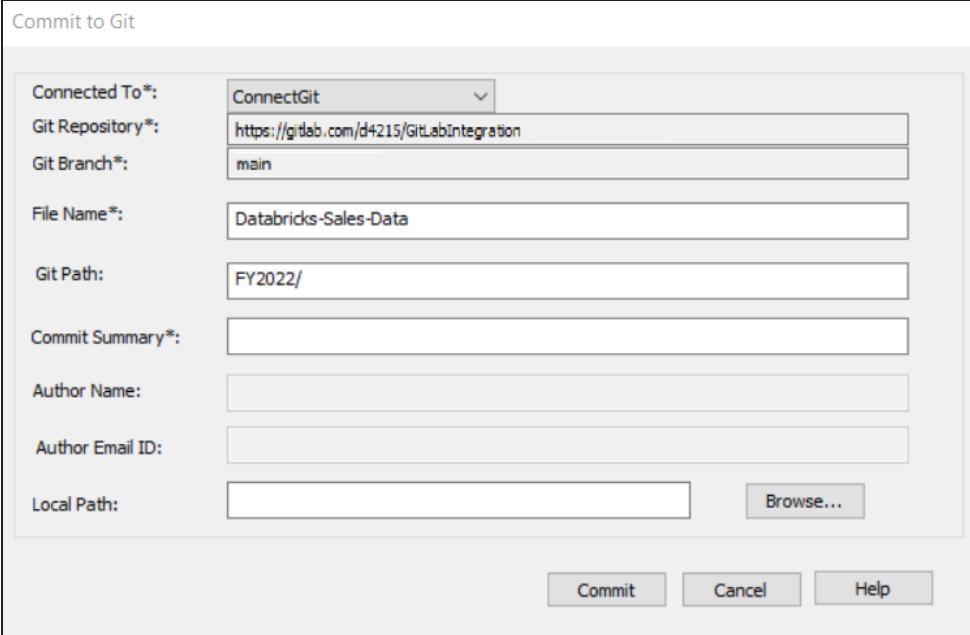
You can click the file to review its content. For example, in the following image, Databricks-Sales-Data's content is visible.



You can use FE Schema Generation Wizard to commit FE script using the same connection again. The Commit to Git screen autopopulates the previously set values in File Name and Git Path.

Committing Forward Engineering Scripts

For example, in the following image File Name is set to Databricks-Sales-Data and Git Path is set to FY2022/.



Commit to Git

Connected To*: ConnectGit

Git Repository*: https://gitlab.com/d4215/GitLabIntegration

Git Branch*: main

File Name*: Databricks-Sales-Data

Git Path: FY2022/

Commit Summary*:

Author Name:

Author Email ID:

Local Path: Browse...

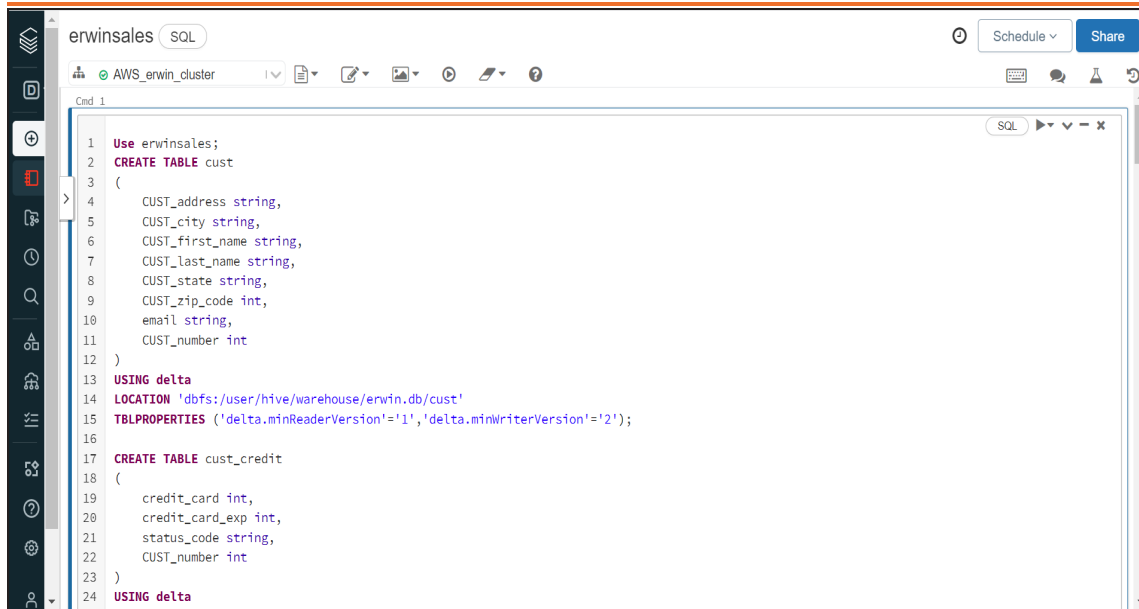
Commit Cancel Help

Committing the FE script again with the same File Name and Git Path overwrites the previous file in the Git repository.

Once the FE script is committed, you can run it on your database to generate and verify the physical schema.

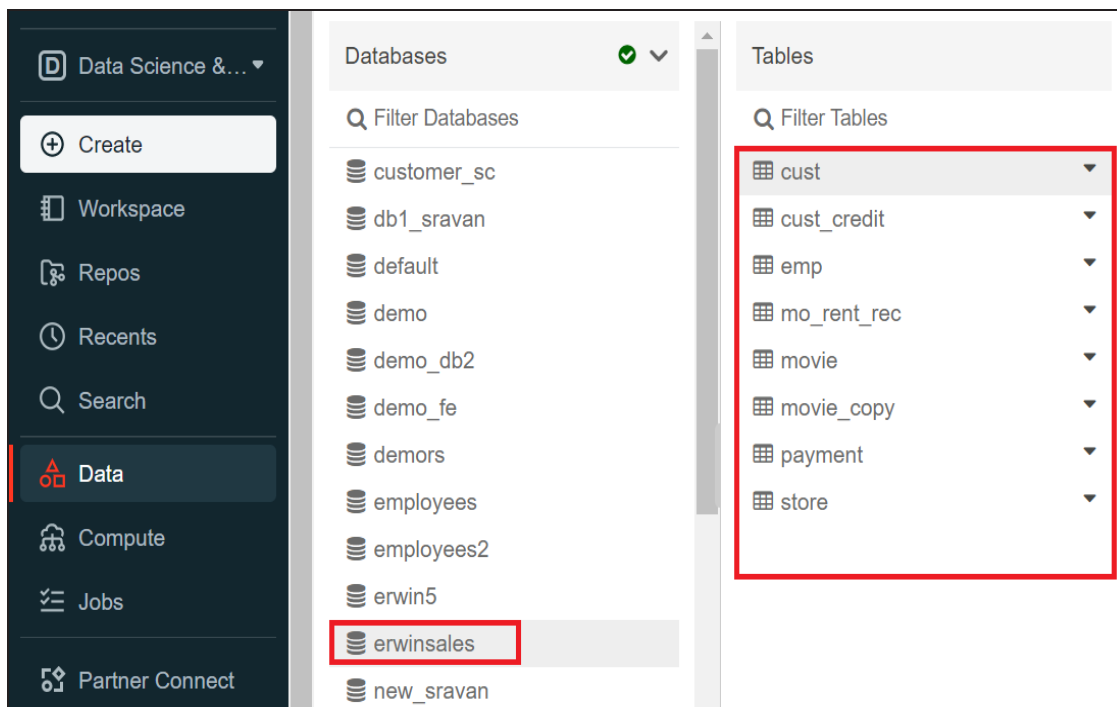
For example, in the following Databricks database, the FE script copied from the Git repository is run.

Committing Forward Engineering Scripts



```
1 Use erwinsales;
2 CREATE TABLE cust
3 (
4     CUST_address string,
5     CUST_city string,
6     CUST_first_name string,
7     CUST_last_name string,
8     CUST_state string,
9     CUST_zip_code int,
10    email string,
11    CUST_number int
12 )
13 USING delta
14 LOCATION 'dbfs:/user/hive/warehouse/erwin.db/cust'
15 TBLPROPERTIES ('delta.minReaderVersion'='1','delta.minWriterVersion'='2');
16
17 CREATE TABLE cust_credit
18 (
19     credit_card int,
20     credit_card_exp int,
21     status_code string,
22     CUST_number int
23 )
24 USING delta
```

After running the FE script, the required database objects are created. You can access these objects from the database. For example, the following tables can be accessed in a Databricks database.



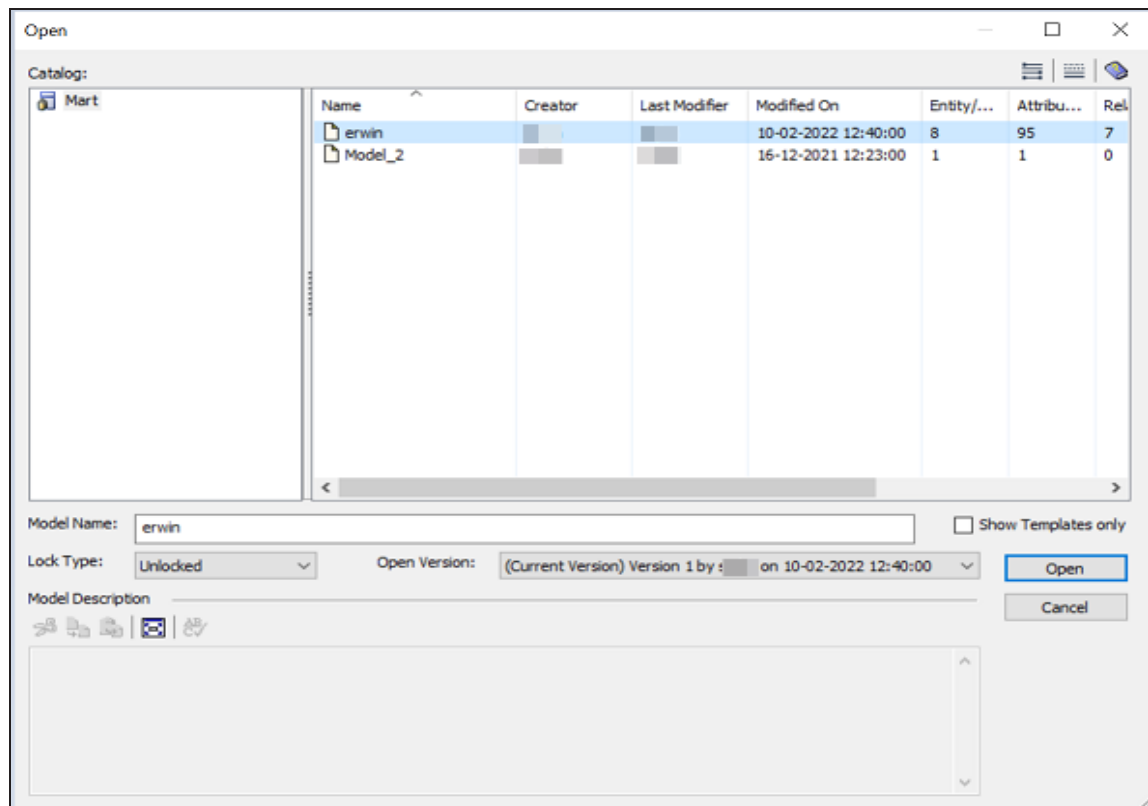
Scenario 2: Committing Alter Scripts

The Forward Engineer Alter Schema Generation Wizard generates an alter script for a database after you make changes to a model. For a Mart Model, you can push the alter script to a Git repository.

To commit alter scripts to Git repositories, follow these steps:

1. On the ribbon, go to **Mart > Open**

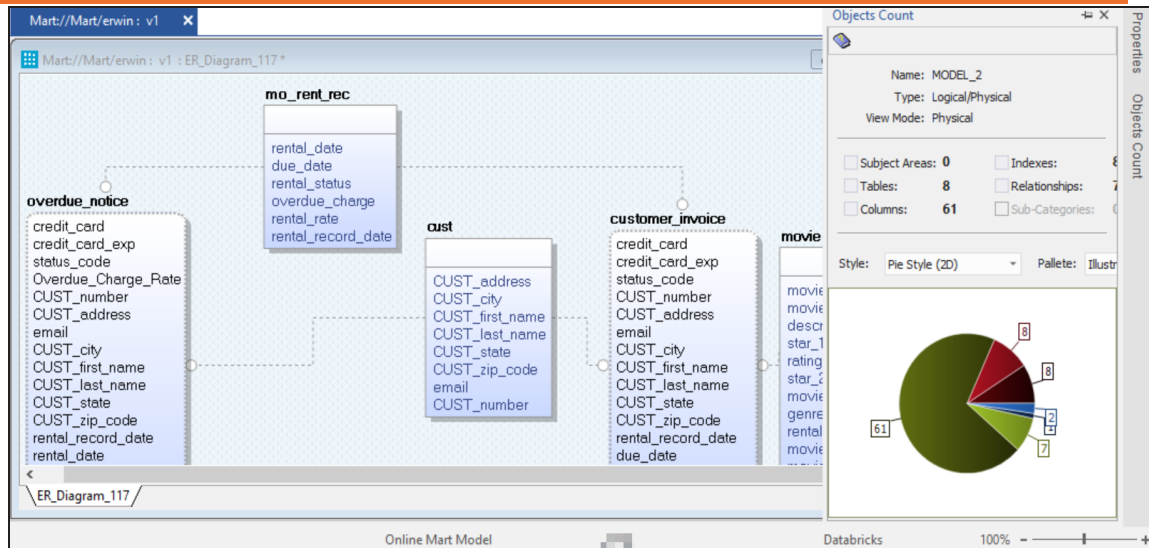
The Open page appears.



2. Select a model, and then click **Open**.

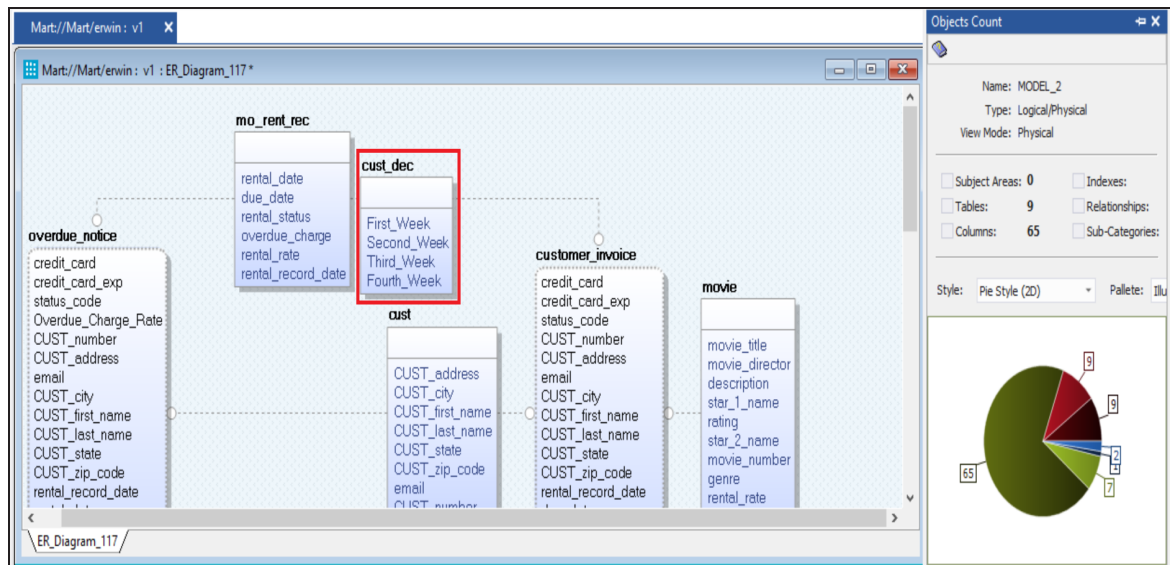
The Mart Model opens.

Committing Forward Engineering Scripts



3. Make the required changes in the model.

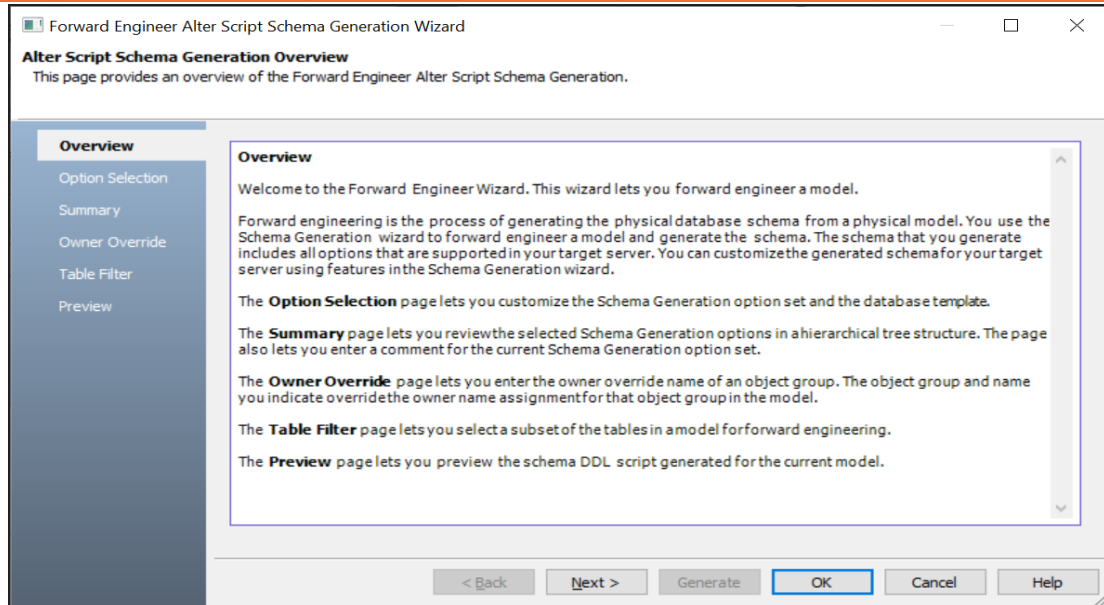
For example, in the following model, a new table, **cust_dec** with four columns is added.



4. Go to **Actions > Alter Script**.

The Forward Engineer Alter Script Schema Generation Wizard appears.

Committing Forward Engineering Scripts

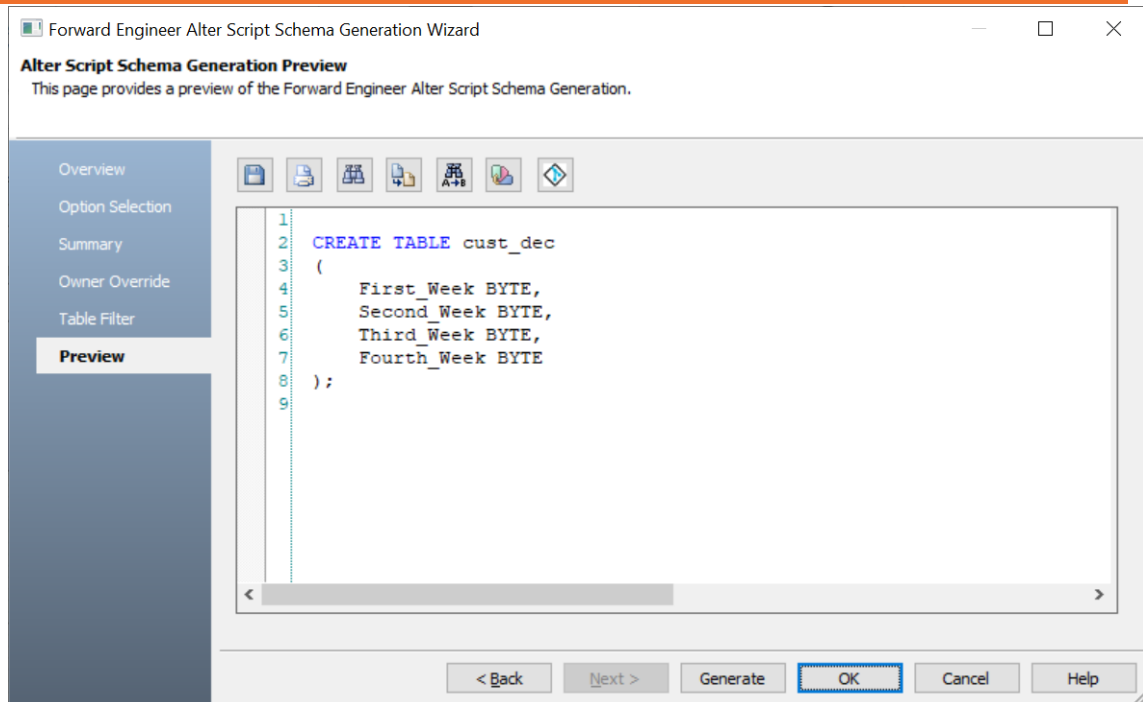


5. On the **Forward Engineer Alter Schema Generation Wizard**, click the **Preview** section.

The alter script appears. For more information on generating alter scripts, refer to the [Generating Alter Script for Databases](#) topic.

For example, in the following image the Preview section displays an alter script of a Databricks database.

Committing Forward Engineering Scripts



6. Click .

The Commit to Git screen appears. The File Name and Git Path values autopopulates with the values configured in the previous commit. You can update the File Name and Git Path as per the requirement.

Committing Forward Engineering Scripts

Commit to Git

Connected To*: ConnectGit

Git Repository*: https://gitlab.com/gitintigration/shubhamgitlab

Git Branch*: shubham_gitlab

File Name*:

Git Path:

Commit Summary*:

Author Name:

Author Email ID:

Local Path: Browse...

☒ Auto Append

Commit Cancel Help

7. Enter appropriate values in the fields. Fields marked with an asterisk (*) are mandatory. Refer to the following table for field descriptions.

Field Name	Description	Additional Information
Connected To	Specifies the connection that connects erwin DM to a Git repository	For example, ConnectGit.
Git Repository	Specifies the Git repository configured for Connection	For example, https://-gitlab.com/d4215/GitLabIntegration is set for the ConnectGit connection. This field autopopulates based on the repository configured in the Git Connection Manager.
Git Branch	Specifies the Git branch that was set for connection in the Git Connection Manager	For example, main is set for the ConnectGit connection. This field autopopulates based on the repository configured in the Git Connection Manager.

Committing Forward Engineering Scripts

Field Name	Description	Additional Information
File Name	Specifies the user-defined name of the FE script file being committed to a Git repository	For example, Databricks-Sales-Data.sql
Git Path	Specifies the location in the Git repository where the FE script is committed	For example, FY2022/ The FE script is committed to the FY2022 folder inside the root folder of your Git repository.
Commit Summary	Specifies the summary of the push commit	For example, Append December Sales.
Author Name	Specifies the name of the author pushes the commit	
Author Email ID	Specifies the email address of the author pushes the commit	
Local Path	Specifies the location on your local machine where the Alter script is saved	C:\Users\SO\Documents\Databricks
Auto Append	Specifies whether the alter script is appended to the file set in File Name and Git	By default, the Auto Append check box is selected. To create a new script file, clear the Auto Append check box and set the File Name and File Path belonging to an existing file. A new file with the following naming convention: <File Name>_YYYY-MM-DD_HH-MM-SS is created.

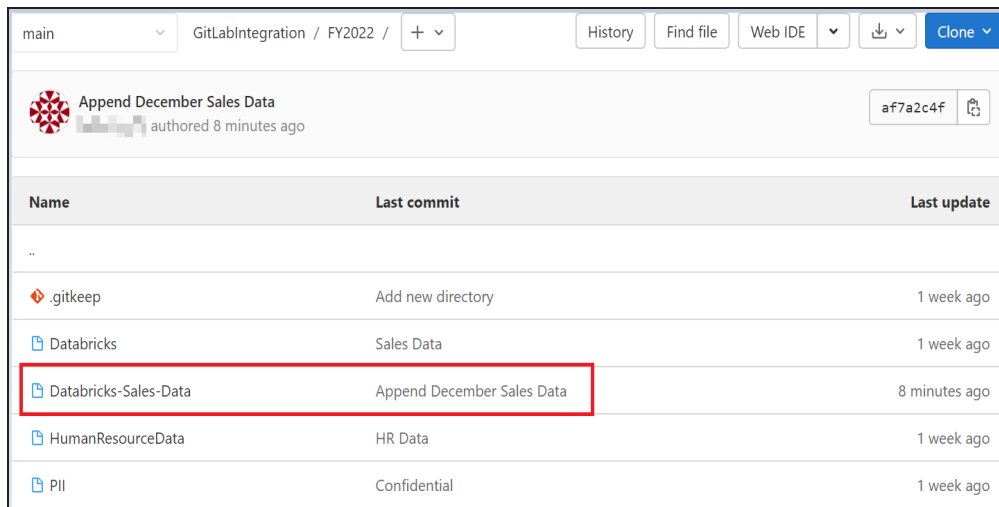
Committing Forward Engineering Scripts

Field Name	Description	Additional Information
	Path	Ensure that you use this check box consistently every time you commit an alter script.

8. Click **Commit**.

The alter script file is saved on the local path and committed to the Git repository.

For example, in the following image, an alter script file is committed to a GitLab repository and appended to an existing file, Databricks-Sales-Data, with a commit summary, Append December Sales using the main branch.



The screenshot shows a GitLab repository interface. At the top, there's a breadcrumb trail: 'main' > 'GitLabIntegration / FY2022 /'. To the right are buttons for 'History', 'Find file', 'Web IDE', and a 'Clone' button. Below this is a commit summary for 'Append December Sales Data' by a user, with a commit hash 'af7a2c4f' and a timestamp 'authored 8 minutes ago'. The main part of the image is a table of commit history.

Name	Last commit	Last update
..		
.gitkeep	Add new directory	1 week ago
Databricks	Sales Data	1 week ago
Databricks-Sales-Data	Append December Sales Data	8 minutes ago
HumanResourceData	HR Data	1 week ago
PII	Confidential	1 week ago

You can click the file to review its content. For example, in the following image, Databricks-Sales-Data contains the alter script.

Committing Forward Engineering Scripts

```
21         status_code string,
22         CUST_number int
23     )
24     USING delta
25     LOCATION 'dbfs:/user/hive/warehouse/erwin.db/cust_credit'
26     TBLPROPERTIES ('delta.minReaderVersion'='1','delta.minWriterVersion'='2');
27
28     CREATE TABLE cust_dec
29     (
30         First_Week BYTE,
31         Second_Week BYTE,
32         Third_Week BYTE,
33         Fourth_Week BYTE
34     );
35
36     CREATE TABLE emp
37     (
38         EMP_first_name string,
39         EMP_address string,
40         EMP_phone int,
41         EMP_address_2 string,
42         email string,
43         salary int,
44         hire_date timestamp,
45         soc_sec_number int,
46         EMP_number string
```

Clearing the Auto Append check box and setting the File Name and File Path belonging to an existing file creates a new file with the following naming convention: <File Name>_YYYY-MM-DD_HH-MM-SS.

For example, in the following image, a file is created with a time stamp in a Git repository.

Committing Forward Engineering Scripts

DocumentationTeam > GitLabIntegration > Repository

main GitLabIntegration / FY2022 / + ▾ History Find file Web IDE ▾ ▾ ▾ Clone ▾

Append Sales Data a4005f10 authored just now

Name	Last commit	Last update
..		
.gitkeep	Add new directory	1 week ago
Databricks	Sales Data	1 week ago
Databricks-Sales-Data	Append December Sales Data	3 days ago
Databricks-Sales-Data_2022-02-21_16-30-04	Append Sales Data	just now
HumanResourceData	HR Data	1 week ago

This file contains only the alter script.

main GitLabIntegration / FY2022 / Databricks-Sales-Data_2022-02-21_16-30-04 Find file Blame History Permalink

Append Sales Data a4005f10 authored just now

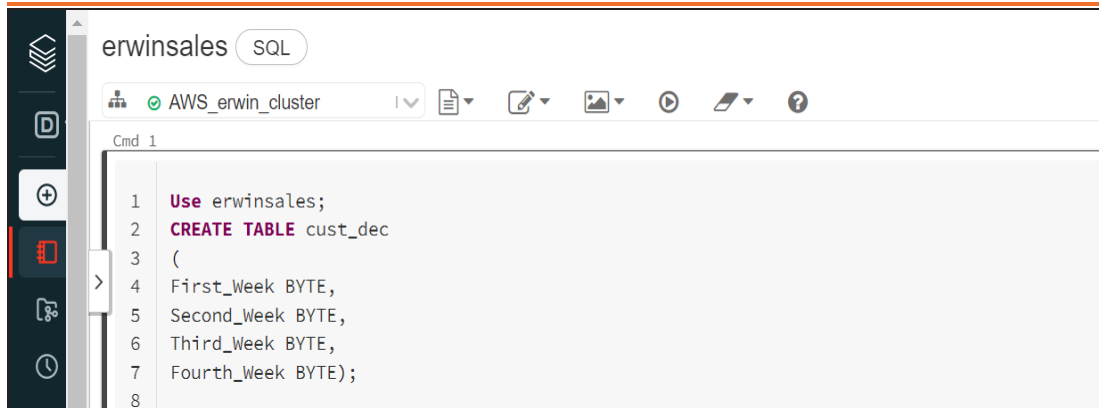
Databricks-Sales-Data_2022-02-21_16-30-04 6.72 KB Edit in Web IDE ▾ Replace Delete

```
1 CREATE TABLE cust_dec
2 (
3     First_Week BYTE,
4     Second_Week BYTE,
5     Third_Week BYTE,
6     Fourth_Week BYTE
7 );
8
```

Use the committed FE script to generate a physical schema in your database. To generate schema, copy the FE script from your Git repository and run the script in the database.

For example, in the following Databricks database, the FE script copied from the Git repository is run.

Committing Forward Engineering Scripts



```
erwinsales SQL
AWS_erwin_cluster
Cmd 1
1 Use erwinsales;
2 CREATE TABLE cust_dec
3 (
4   First_Week BYTE,
5   Second_Week BYTE,
6   Third_Week BYTE,
7   Fourth_Week BYTE);
8
```

The cust_dec table is created in a Databricks database.

