erwin Data Modeler

Feature Tour

Release 15.0

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.

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 erwin by Quest Products page.

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

Introduction

The Feature Tour guide walks Data Architects, Data Administrators, Application Administrators, Database Administrators, and Partners through the features introduced in erwin Data Modeler (erwin DM) 15.0 release.

The features and enhancements introduced in this release are:

- Orchestration Integration with Jira
- OpenAPI Specification Models
- erwin DM-erwin DI Logical Names Mapping
- DBT Integration
- Snowflake Enhancements
- JSON Enhancements
- Google BigQuery Enhancements
- PostgreSQL Enhancements
- Productivity and UI Enhancements
- erwin Mart PortalEnhancements
- erwin ER360 Enhancements

Orchestration Integration with Jira

erwin DM introduces Jira integration to streamline collaboration between business users and data modelers. You can now link Jira tickets ID to models, harvest them to erwin ER360, and automatically sync ticket details, comments, and status updates without switching between applications.

> This feature is available only for Jira Cloud.

This feature provides the following benefits:

- Enhances cross-team collaboration between data architects and development teams
- Provides visibility of modeling changes within existing Jira workflows
- Reduces context switching between applications
- Enables trackable approval process
- Improves overall development process efficiency

Prerequisites

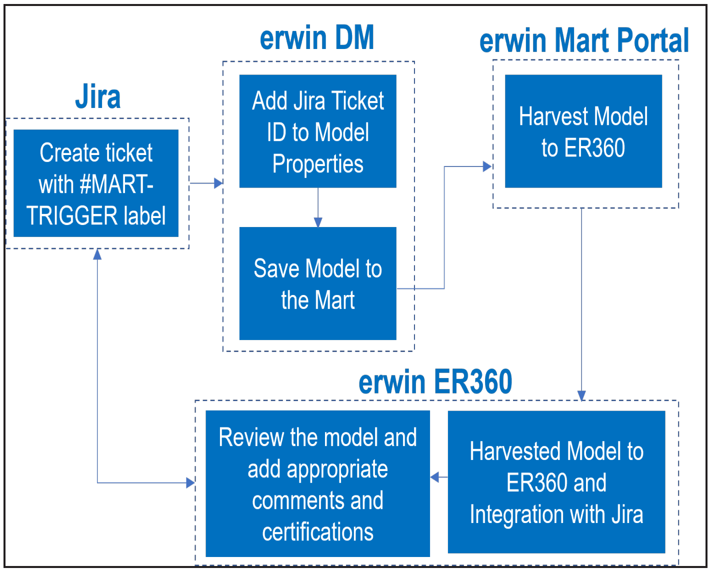
Ensure that the following prerequisites are in place:

- A Jira account with the Jira connector enabled. If it is not enabled, contact your sales team.
- Access to erwin Mart Portal
- Access to erwin ER360 with harvesting permissions

Workflow

erwin DM integrates with Jira to track model related tasks by associating Jira tickets ID to models. When you save models to Mart and harvest them to ER360, the integration automatically updates the associated Jira tickets with comments and model links. This two-way synchronization reflects changes made in ER360 within Jira and vice versa, improves collaboration, traceability.

To summarize, Jira, erwin DM, erwin Mart Portal, and erwin ER360 work together as follows.


Jira
Create ticket with #MART-TRIGGER label
erwin DM
Add Jira Ticket ID to Model Properties
Save Model to the Mart
erwin Mart Portal
Harvest Model to ER360
erwin ER360
Harvested Model to ER360 and Integration with Jira
Review the model and add appropriate comments and certifications

This process involves the following steps:

- Creating a Jira Ticket
- Associating Jira Tickets to Models
- Harvesting a Model to erwin ER360

Creating a Jira Ticket

To create a Jira ticket, follow these steps:

1. Login to your Jira account.
2. Click **Create**.

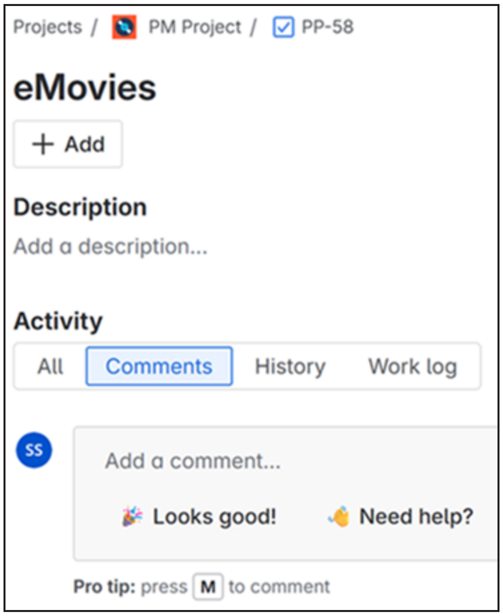
A **Create** page appears.

3. Enter appropriate values in the required fields.
4. Configure the Labels field value to #MART-TRIGGER.

   The #MART-TRIGGER label creates an association between models in erwin ER360 and Jira.

5. Click **Create**.

   A ticket is created with an ID. For example, the screenshot below displays an eMovies ticket with ID PP-58.



You can now associate this ticket ID to the model in erwin DM.

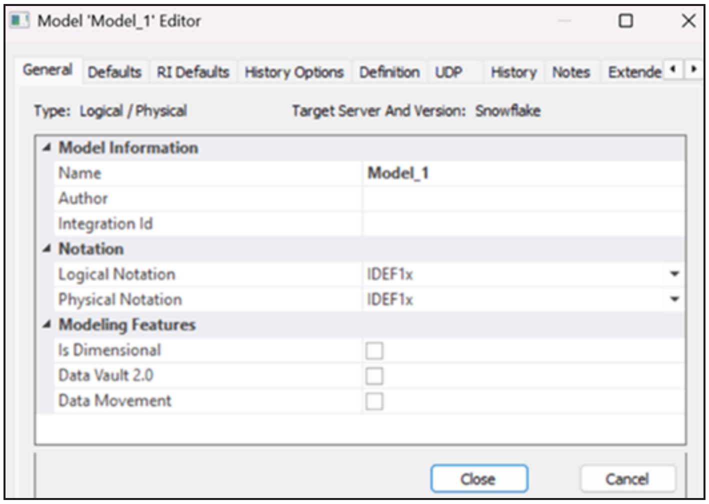
Associating Jira Tickets to Models

To enable Jira integration and modeling task tracking, you need to associate your models with the corresponding Jira ticket.

To associate Jira tickets to models, follow these steps:

1. In the Model Explorer, right-click the model and click **Properties**.

   The Model Editor opens and by default, the General tab appears.

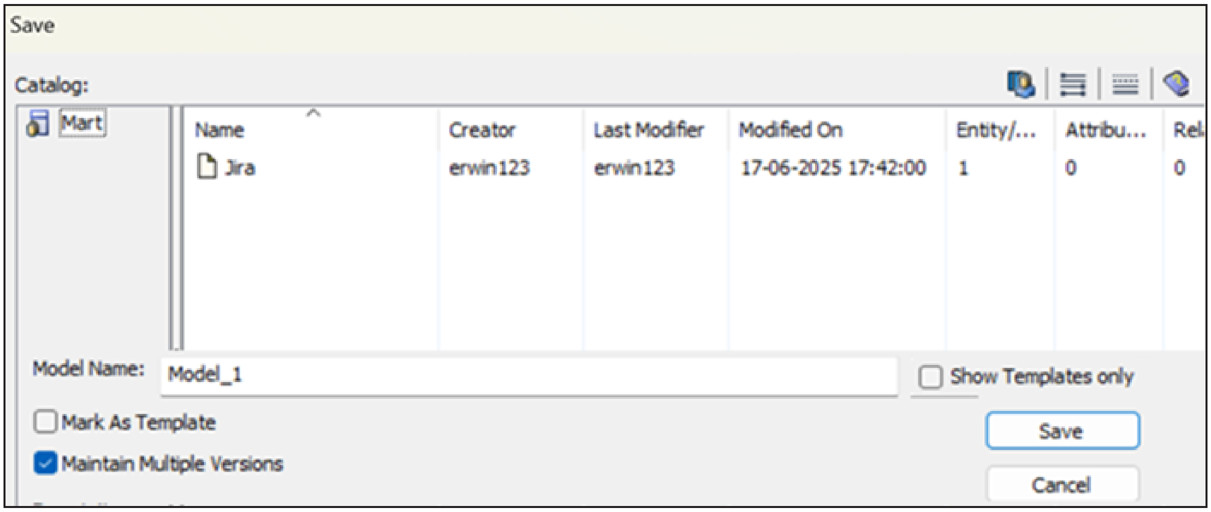
   

   Type: Logical / Physical   Target Server And Version: Snowflake

Model Information	
Name	**Model_1**
Author	
Integration Id	
Notation	
Logical Notation	IDEF1x
Physical Notation	IDEF1x
Modeling Features	
Is Dimensional	□
Data Vault 2.0	□
Data Movement	□

2. In the **Integration Id** box, enter the ticket ID. For example, PP-58.
3. Click **Close**.
4. Ensure that you are connected to erwin Mart Portal.
5. On the ribbon, click **Mart** > **Save**.

   The Save dialog box opens.



Save

Catalog:

Mart

Name	Creator	Last Modifier	Modified On	Entity/...	Attribu...	Rel
Jira	erwin123	erwin123	17-06-2025 17:42:00	1	0	0

Model Name: Model_1

□ Show Templates only

□ Mark As Template

☑ Maintain Multiple Versions

6. Select the library where you want to save your model.
7. In the Model Name box, enter a name of model and then, click **Save**.

   The model is saved to mart.

Harvesting a Model to erwin ER360

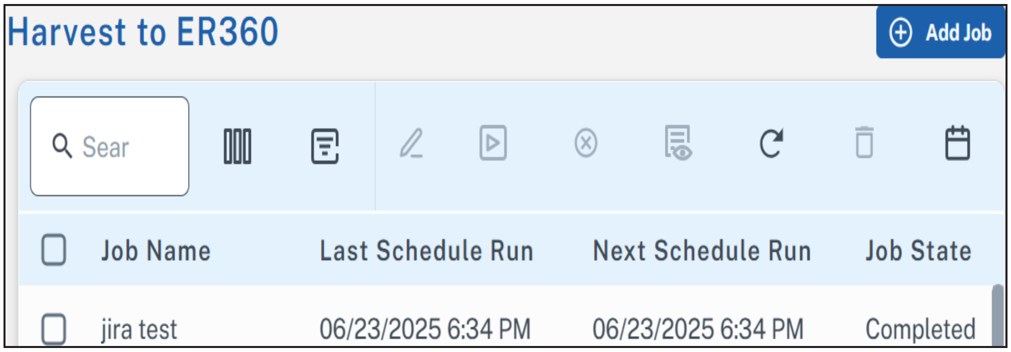
To harvest a model to erwin ER360, you must schedule a harvesting job. Ensure that you have data harvesting permissions to perform this job.

To schedule a job, follow these steps:

1. In the header pane, click ▦. Then, click **Harvest to ER360**.

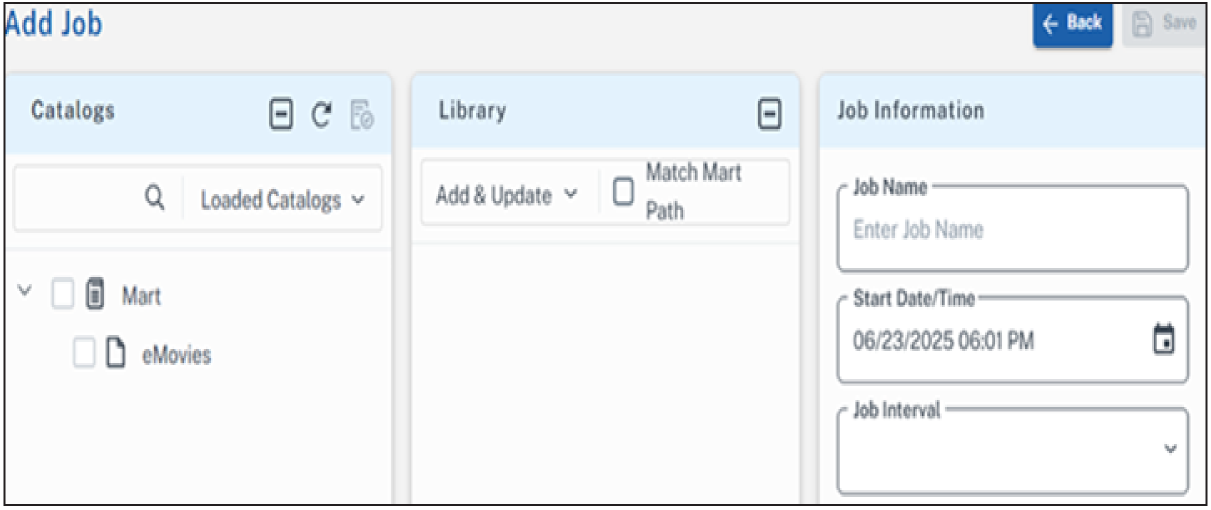
   > This option is available only if you have a license for erwin ER360 and have initialized it.

   The Harvest to ER360 page appears.



□	Job Name	Last Schedule Run	Next Schedule Run	Job State
□	jira test	06/23/2025 6:34 PM	06/23/2025 6:34 PM	Completed

2. Click **Add Job**.
   The Add Job page appears.
   Alternatively, you can click **Add Job** from the calendar view page to schedule jobs.



Job Name: Enter Job Name

Start Date/Time: 06/23/2025 06:01 PM

Job Interval:

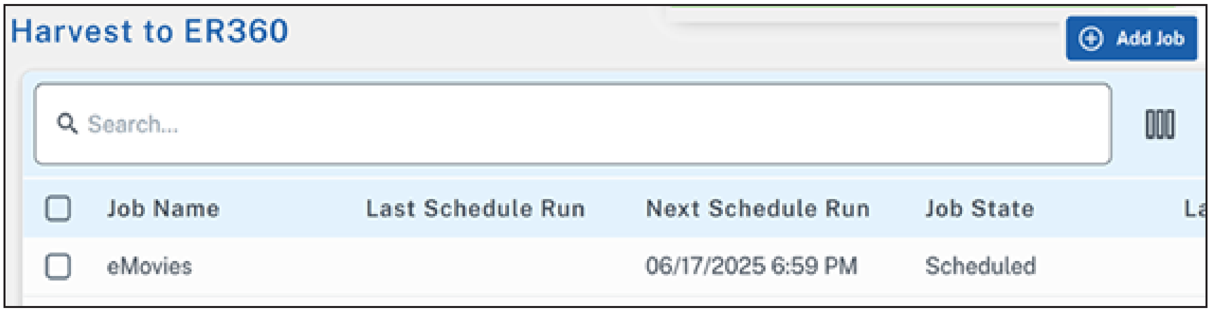
3. In the Catalogs pane, select your model to export to erwin ER360.

4. In the Library pane, select a library in erwin ER360 to save the exported models.

5. In the Job Information pane, enter appropriate values in the required fields.

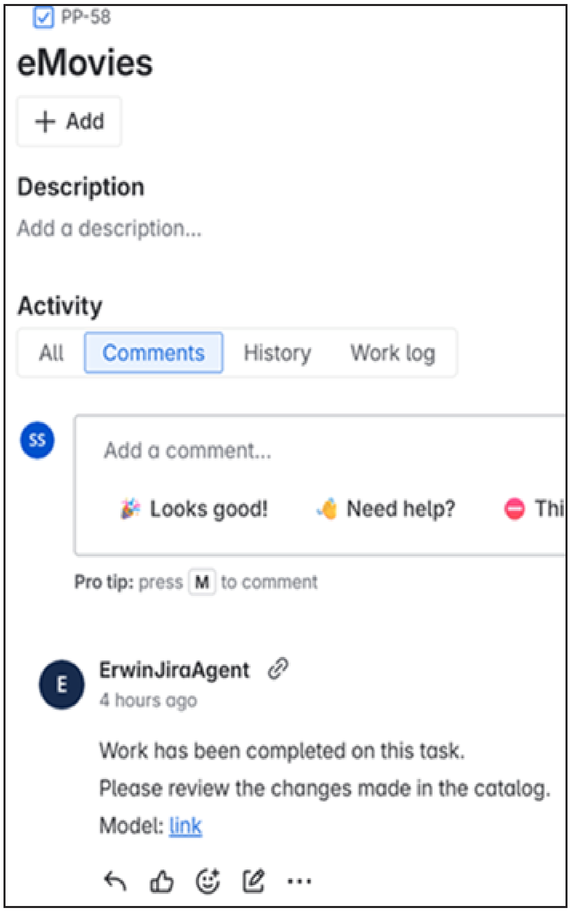
6. Select the Run Now checkbox to run the job immediately.

7. Click **Save**.
   The job is added to the list with its **Job State** set to Scheduled.



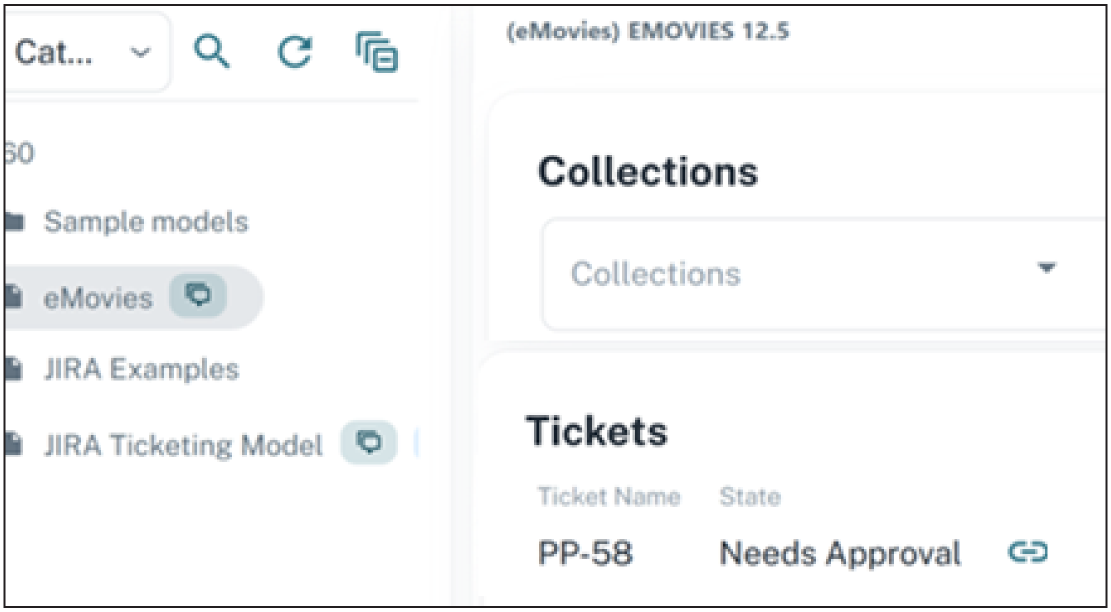
□	Job Name	Last Schedule Run	Next Schedule Run	Job State	La
□	eMovies		06/17/2025 6:59 PM	Scheduled	

Once the job is successful, the model is harvested to erwin ER360 and an automated comment is added to the linked Jira ticket with the model link and review request.



Work has been completed on this task.
Please review the changes made in the catalog.
Model: link

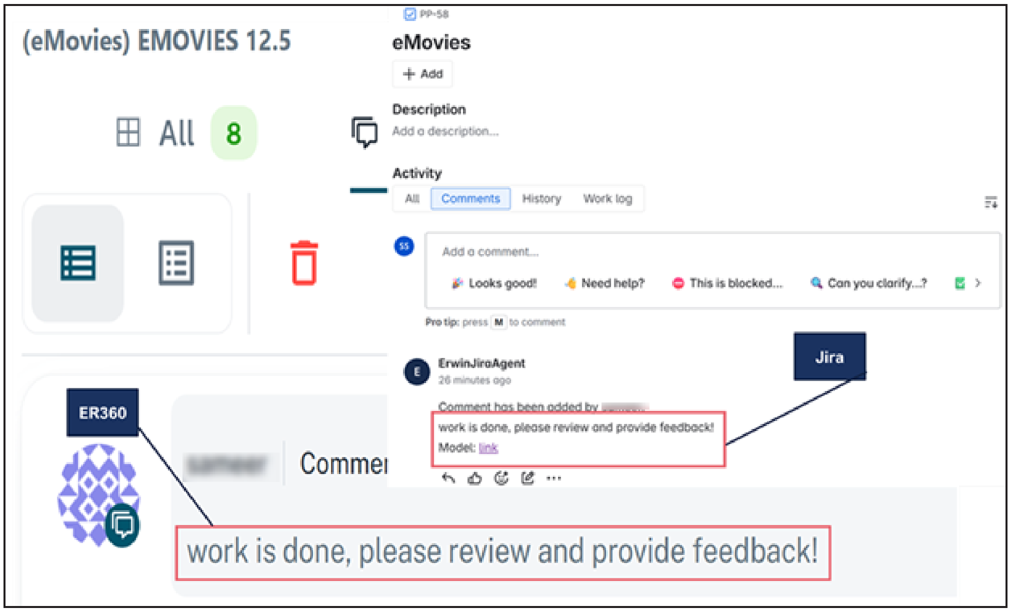
Clicking this link opens the model in erwin ER360, where you can view the ticket ID and status. For example, the screenshot below displays ticket ID and status. Also, you can click the link next to the State field to go back to the Jira ticket.



Tickets

Ticket Name	State
PP-58	Needs Approval

You can also review the model and add comments, which automatically sync between erwin ER360 and the linked Jira ticket. For example, the screenshot below shows how comments are synchronized between erwin ER360 and Jira.



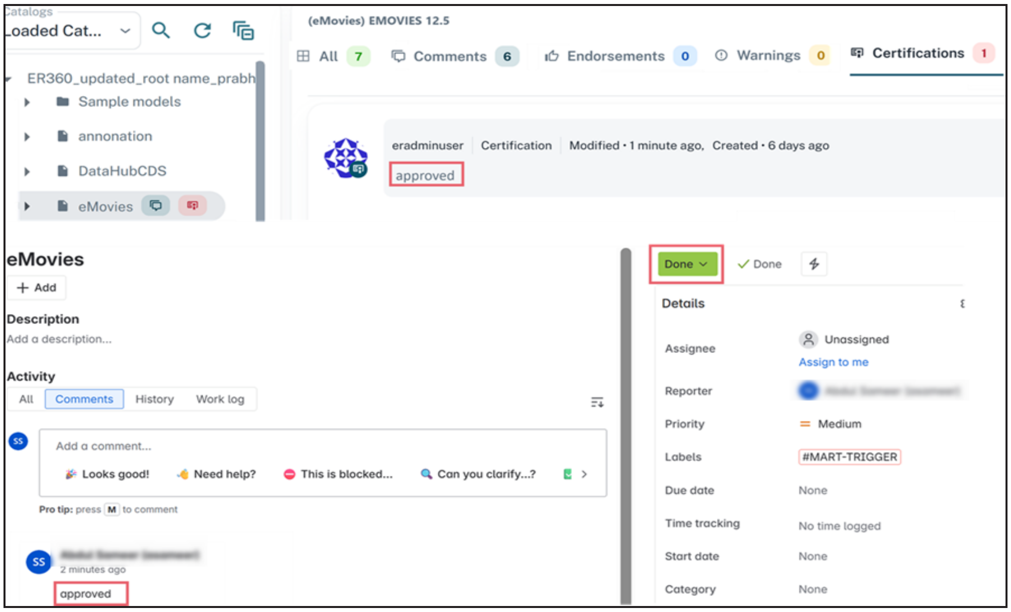
ErwinJiraAgent
26 minutes ago

Comment has been added by [redacted]
work is done, please review and provide feedback!
Model: link

[redacted] Commer

work is done, please review and provide feedback!

After reviewing the model, cetify it to mark it as approved. This action updates the linked Jira ticket status to Done. For example, the screenshot below shows a certification added in ER360 and the corresponding status update in Jira.

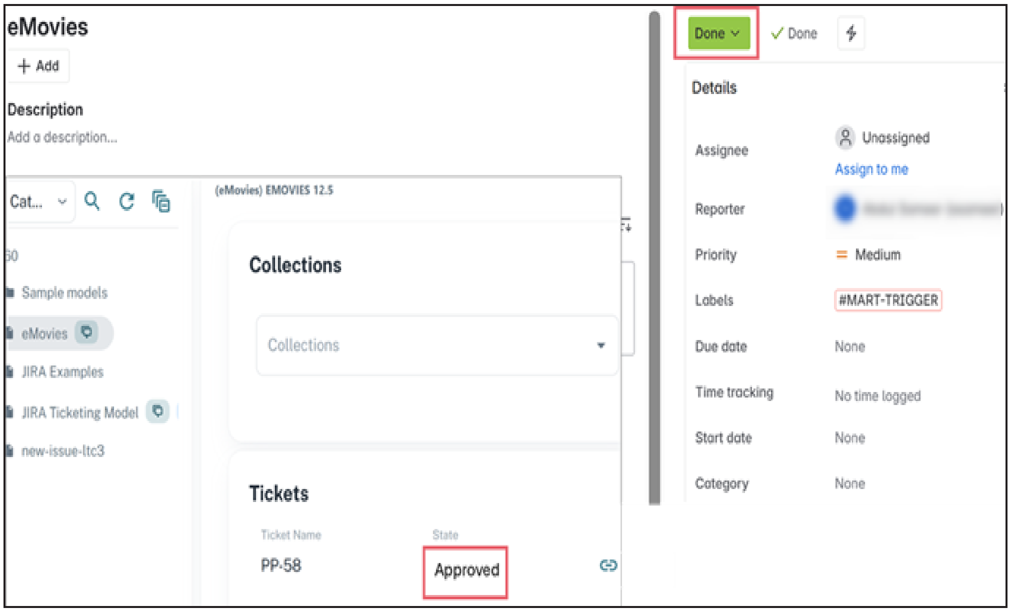


eradminuser | Certification | Modified • 1 minute ago, Created • 6 days ago
approved

SS [redacted]
2 minutes ago
approved

Assignee	Unassigned Assign to me
Reporter	[redacted]
Priority	Medium
Labels	#MART-TRIGGER
Due date	None
Time tracking	No time logged
Start date	None
Category	None

Likewise, when the Jira status changes to Done, ER360 updates the model status to Approved. For example, the screenshot below shows the status change between erwin ER360 and Jira.



eMovies

+ Add

Description

Add a description...

Cat...

60

- Sample models
- eMovies
- JIRA Examples
- JIRA Ticketing Model
- new-issue-ltc3

(eMovies) EMOVIES 12.5

Collections

Collections ▾

Tickets

Ticket Name	State
PP-58	Approved

Done ⌄ ✓ Done

Details

Assignee	Unassigned Assign to me
Reporter	[redacted]
Priority	= Medium
Labels	#MART-TRIGGER
Due date	None
Time tracking	No time logged
Start date	None
Category	None

OpenAPI Specification (OAS) Models

You can now create and manage OpenAPI specifications using a model-driven, diagram-based approach in erwin DM. This enables API developers to design OpenAPI specifications like data models with a familiar modeling environment. This ensures:

- Consistency between data structures and API definitions
- Reduced duplication by reusing existing data models
- Improved collaboration and communication via visual API representation

You can create physical OpenAPI models using predefined OpenAPI components and then build on top of it. These models support reverse engineering and forward engineering via specifications in JSON and YAML formats.

Creating OAS Models

erwin DM supports OpenAPI modeling using predefined specification components that follow structure and terminology according to the OpenAPI Specification (OAS). These components, when used for modeling are designed to include metadata for APIs, requests, and responses. The OAS implementation in erwin DM supports JSON and YAML file formats for reverse engineering and forward engineering.

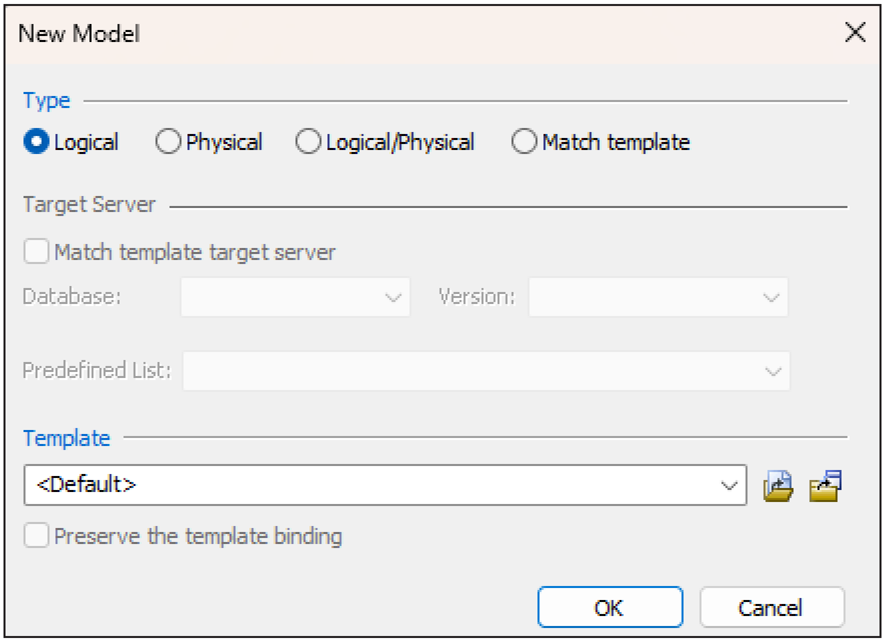
> OAS implementation supports only Physical modeling.

Creating OAS Models

To create OAS models and objects, and define their properties, follow these steps:

1. In erwin DM, click **File** > **New**.

   The New Model screen appears.



New Model

Type

● Logical ○ Physical ○ Logical/Physical ○ Match template

Target Server

□ Match template target server

Database:

Version:

Predefined List:

Template

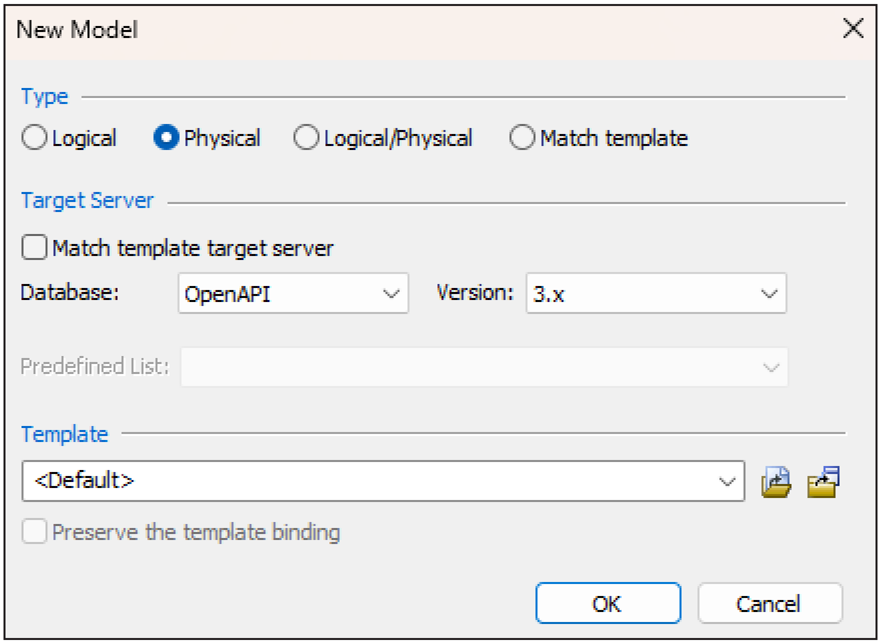
<Default>

□ Preserve the template binding

2. Configure the following options:

   1. Click **Physical**.

   2. In the Database list, select **OpenAPI**.



New Model

Type

○ Logical ● Physical ○ Logical/Physical ○ Match template

Target Server

□ Match template target server

Database: OpenAPI Version: 3.x

Predefined List:

Template

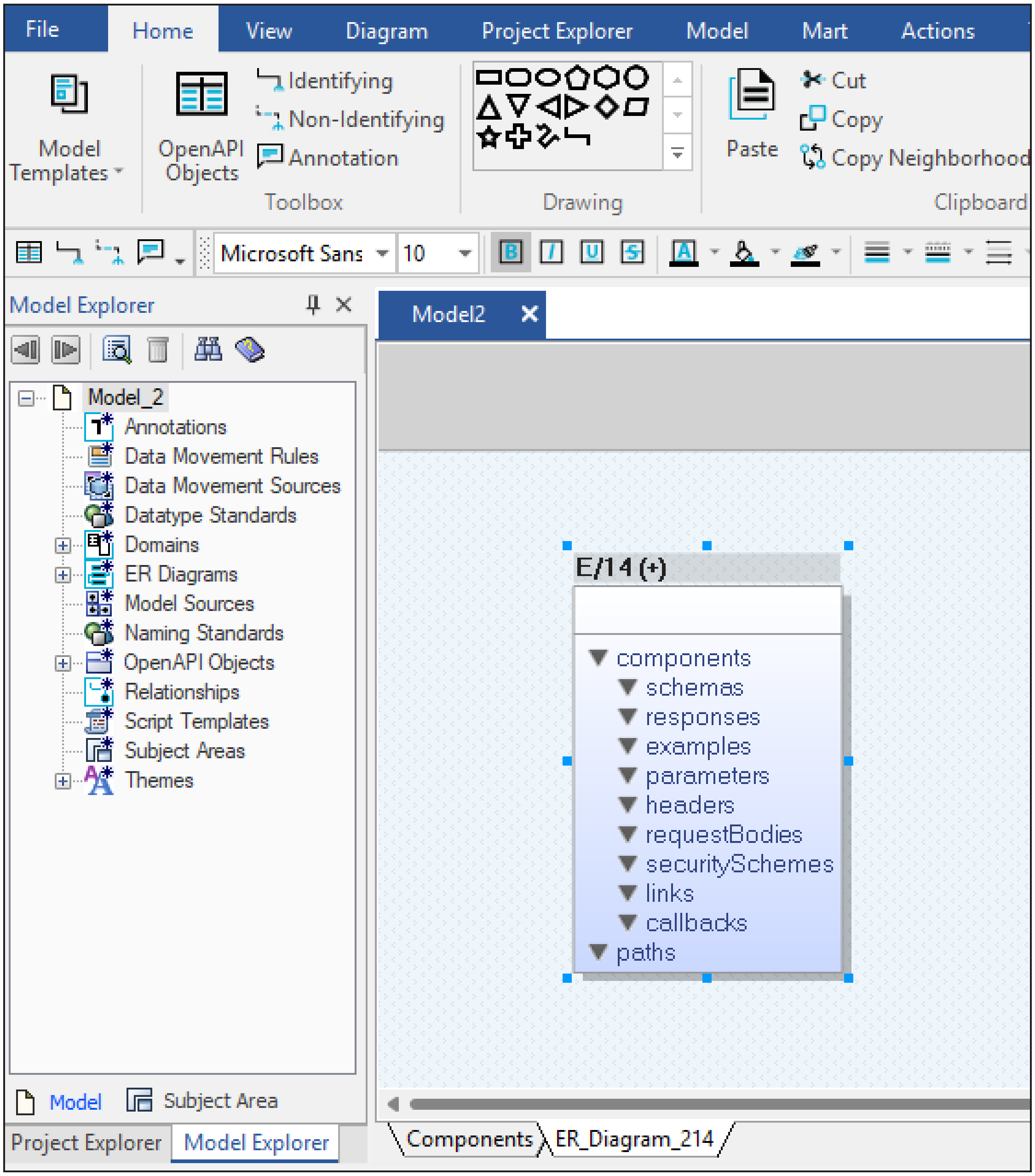
<Default>

□ Preserve the template binding

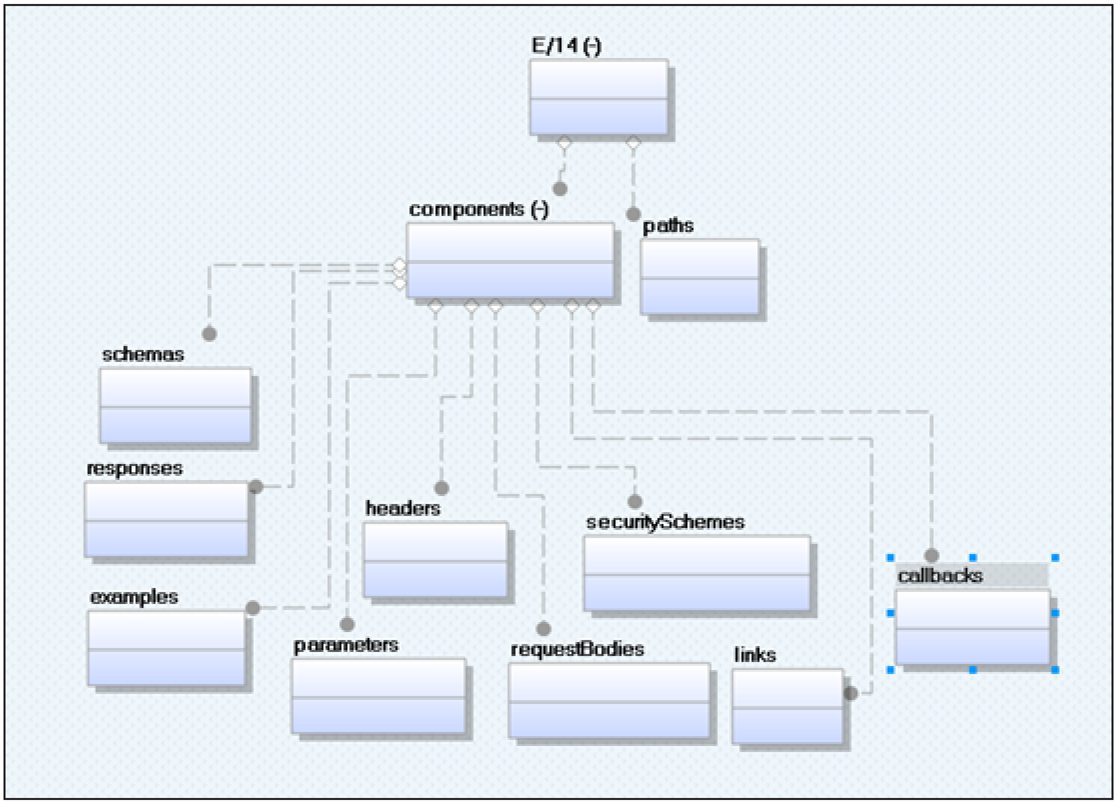
3. Click **OK**.

   A blank physical model is created.

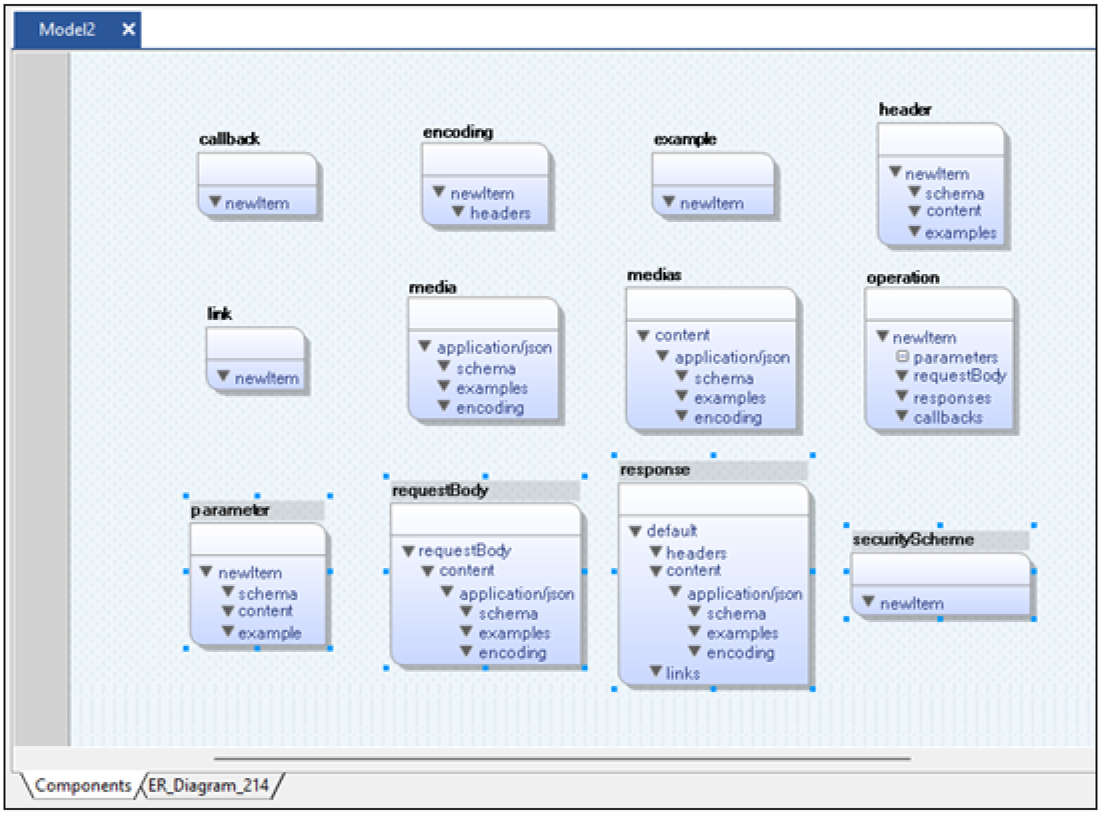
4. On the ribbon, click **Home** > **OpenAPI Objects** and add it to the diagram.



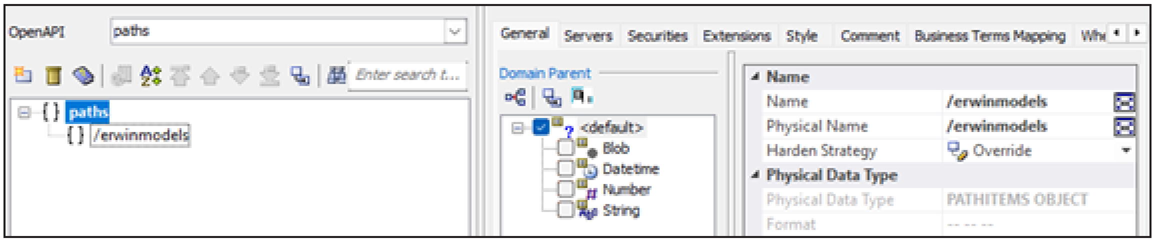
An object with empty predefined OAS object types is added to the model diagram. The + sign on the object indicates that you can expand it. To expand the object, you can double-click the object name or on the ribbon, click **Actions** > **Hierarchical View**.



To view all available predefined OAS objects, click the **Components** tab. These components are reusable objects.



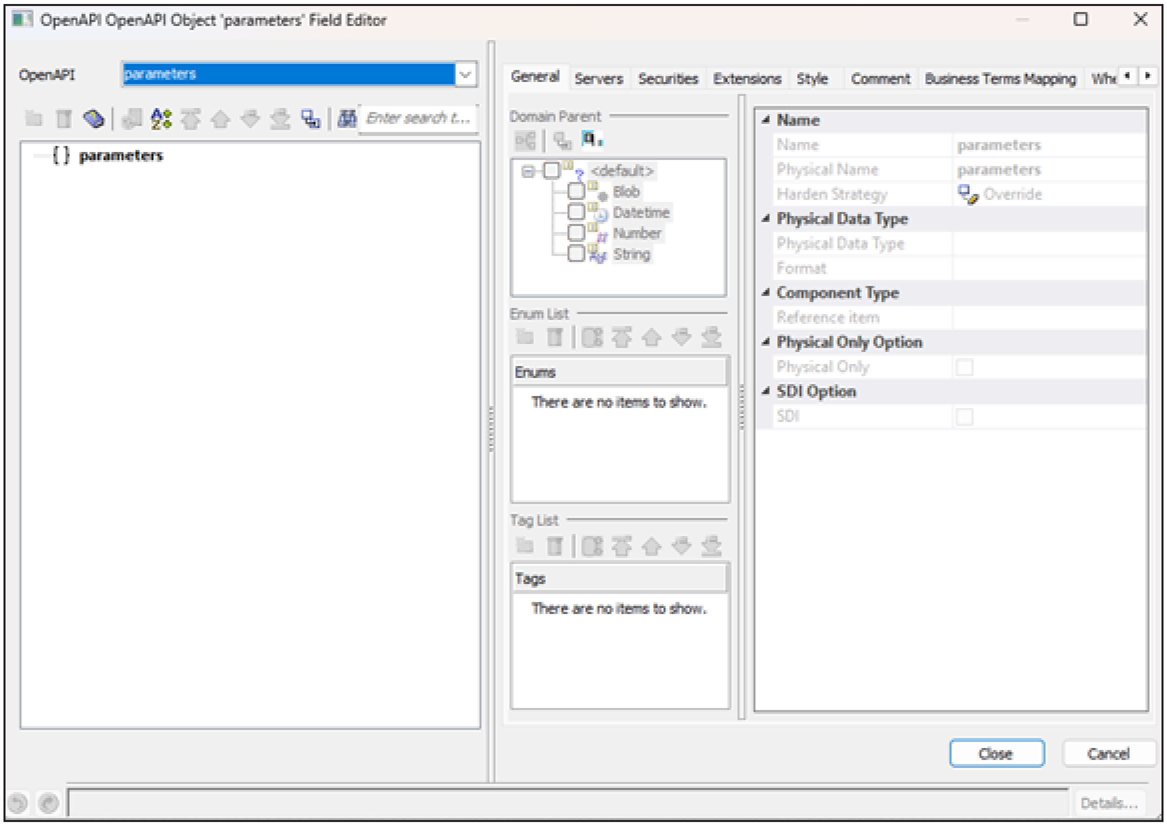
Use the path object available in the model diagram to specify the relative path to an individual API endpoint. Path names must start with a / (front-slash). For example, /erwinmodels as shown in the following image:



Name	
Name	/erwinmodels
Physical Name	/erwinmodels
Harden Strategy	Override
Physical Data Type	
Physical Data Type	PATHITEMS OBJECT
Format	-- -- --

5. On the object type where you want to add objects, right-click and click **Field Properties**.

   For example, right-click the parameters object type and click **Field Properties**. These properties form the API metadata.

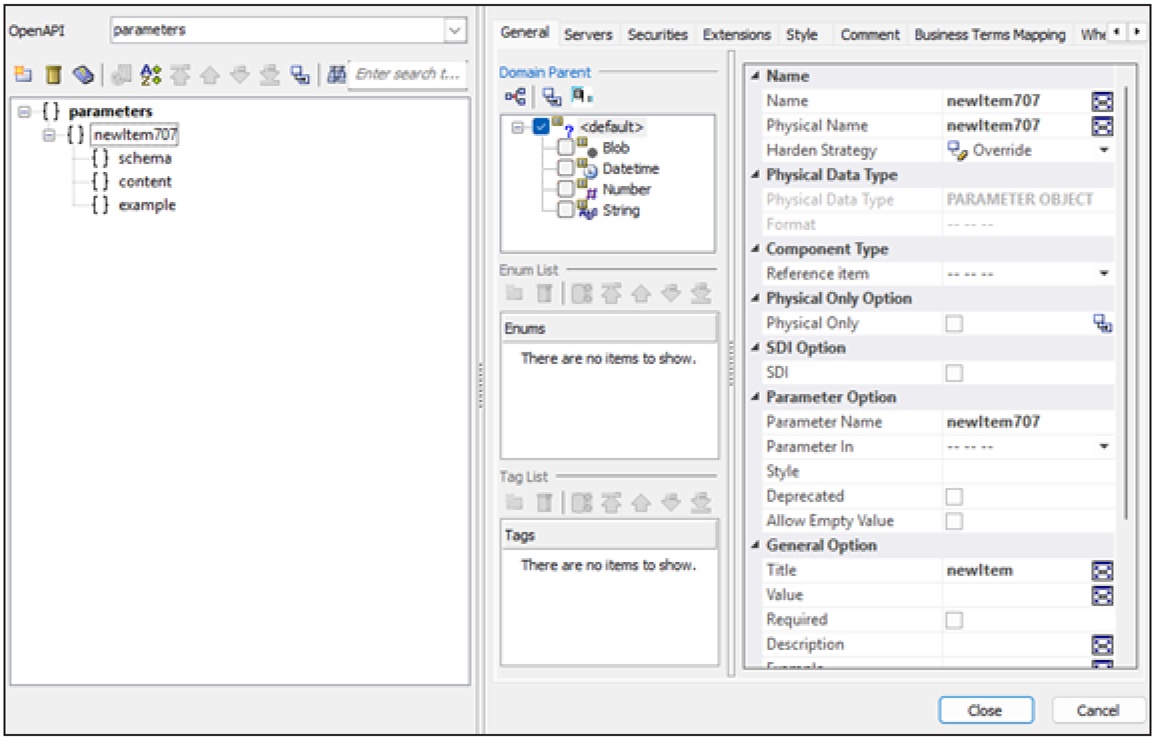


Name	
Name	parameters
Physical Name	parameters
Harden Strategy	Override
Physical Data Type	
Physical Data Type	
Format	
Component Type	
Reference item	
Physical Only Option	
Physical Only	☐
SDI Option	
SDI	☐

6. On the Field Editor, right-click the component name and click **Add Component**.

   This adds the corresponding predefined component with all the necessary properties.

   For example, right-click **{ } parameters** and click **Add Component**.



OpenAPI: parameters

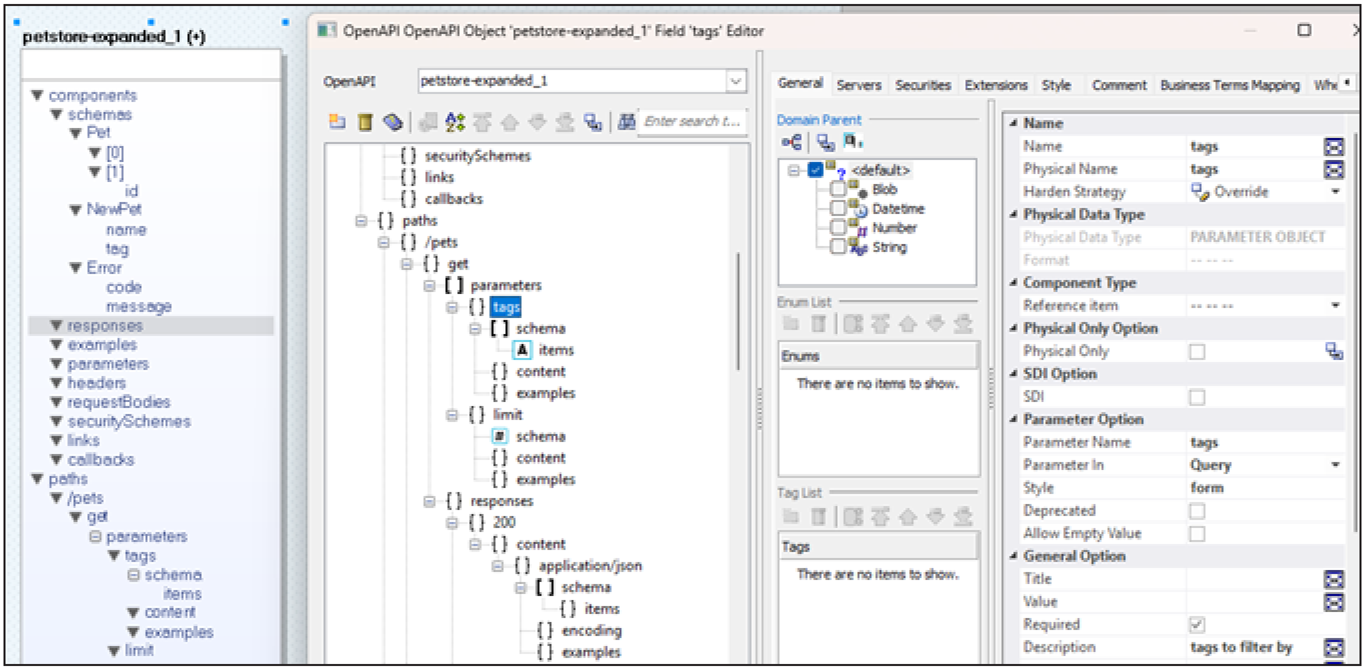
- { } parameters
  - { } newItem707
    - { } schema
    - { } content
    - { } example

Name	
Name	newItem707
Physical Name	newItem707
Harden Strategy	Override
Physical Data Type	
Physical Data Type	PARAMETER OBJECT
Format	-- -- --
Component Type	
Reference item	-- -- --
Physical Only Option	
Physical Only	☐
SDI Option	
SDI	☐
Parameter Option	
Parameter Name	newItem707
Parameter In	-- -- --
Style	
Deprecated	☐
Allow Empty Value	☐
General Option	
Title	newItem
Value	
Required	☐
Description	

7. Set up the required property values and click **Close**.

   Similarly, set up all the required object properties to complete the OAS model.

For a better idea about OAS models, the following image shows a sample petstore model and properties.



Name	
Name	tags
Physical Name	tags
Harden Strategy	Override
Physical Data Type	
Physical Data Type	PARAMETER OBJECT
Format	-- -- --
Component Type	
Reference item	-- -- --
Physical Only Option	
Physical Only	□
SDI Option	
SDI	□
Parameter Option	
Parameter Name	tags
Parameter In	Query
Style	form
Deprecated	□
Allow Empty Value	□
General Option	
Title	
Value	
Required	☑
Description	tags to filter by

> Refer to the OAS documentation for detailed information on OAS description structure.

Reverse Engineering OAS Models

Following sections explain the reverse engineering options for OpenAPI.

Overview

Parameter	Description	Additional Information
Script File	Specifies the reverse engineering source	**Script File**: Indicates that the model is reverse engineered from a script
File	Specifies the path of the script file that should be used for reverse engineering	Supported file formats are JSON and YAML.

Detailed Options

Parameter	Description	Additional Information
Glossary CSV File	Specifies the naming standard glossary file in the .CSV format	
Case Conversion of Physical Names	Specifies how the case conversion of physical names is handled	Not applicable
Case Conversion of Logical Names	Specifies how the case conversion of logical names is handled	Not applicable
Save Field Value	Specifies whether values of attributes or fields are saved to the model	

Scheduler

The options on this tab are available only while reverse engineering via erwin DM Scheduler.

Parameter	Description	Additional Information
Model	Specifies the location and name of the reverse engineered model	For example: C:\Scheduler\<Model Name>.erwin When you schedule a job on a remote server, ensure the model path is same for remote and local server.
Mart Folder	Specifies the location or library in your mart where the reverse engineered model is saved	To use this option, ensure that you are connected to mart. For more information, refer to the Connecting to Mart topic.
Complete Compare	Specifies whether the Complete Compare (CC) process should run while reverse engineering	
Output File	Specifies the location of the CC output file generated	
File	Specifies that the target model location is on the local system	

Parameter	Description	Additional Information
Mart	Specifies that the target model location is in the mart	
Using Latest Version	Specifies whether the target model is the latest version of the model in the mart	This option is available only when Mart is selected.
Save To Mart	Specifies whether the reverse engineered model is saved to the mart	This option is available only when Using Latest Version is selected.
Target Model	Specifies the location of the target model for CC	
Option Set	Specifies the option set that is used for CC	**Advanced Default Option Set**: Indicates that all erwin DM metadata is included. CC works the slowest with this option. **Speed Option Set**: Indicates that only the essential metadata is included. CC works the fastest with this option set. **Standard Default Option Set**: Indicates that standard metadata is included. CC works fast with this option set compared to the Advanced option set. In addition to the above options, click **Browse** to select a custom option set for complete compare.
Compare Level	Specifies the selection type for the compare	**Logical / Physical**: Compares all objects on the logical or physical level of a model **Logical**: Compares all objects on the logical level of a model **Physical**: Compares all objects on the physical level of a model **Database**: Compares all objects on the database level of a model

erwin Project

Parameter	Description	Additional Information
erwin Project	Specifies whether to configure the model for an existing erwin project	
Model Name	Specifies the name of the erwin project	
Location	Specifies the location of the project	

Forward Engineering OAS Models

Following sections explain the forward engineering options for OpenAPI.

Option Selection

Parameter	Description	Additional Information
Option Set	Specifies the option set template for forward engineering	**Open**: Use this option to open a saved XML option set file. **Save**: Use this option to save a configured option set. **Save As**: Use this option to save an option set in the XML format. **Delete**: Use this option to delete an option set.
Database Template	Specifies the database template for controlling schema generation	
Script Option	Specifies the script option for the schema generation	**Pre-Script**: Indicates whether pre-scripts attached to the schema are executed **Post-Script**: Indicates whether the post-scripts attached to the schema are executed
General Syntax Option	Specifies the general options for schema generation	**Data**: Indicates whether to include model data in the schema **Schema**: Indicates whether to include

Parameter	Description	Additional Information
		model design details in the schema **Comments**: Indicates whether comments are included in the schema
Collection Syntax Option	Specifies the collection options for schema generation	**Blank Value**: Indicates whether to include a blank value instead of other characters in the schema

Object Filter

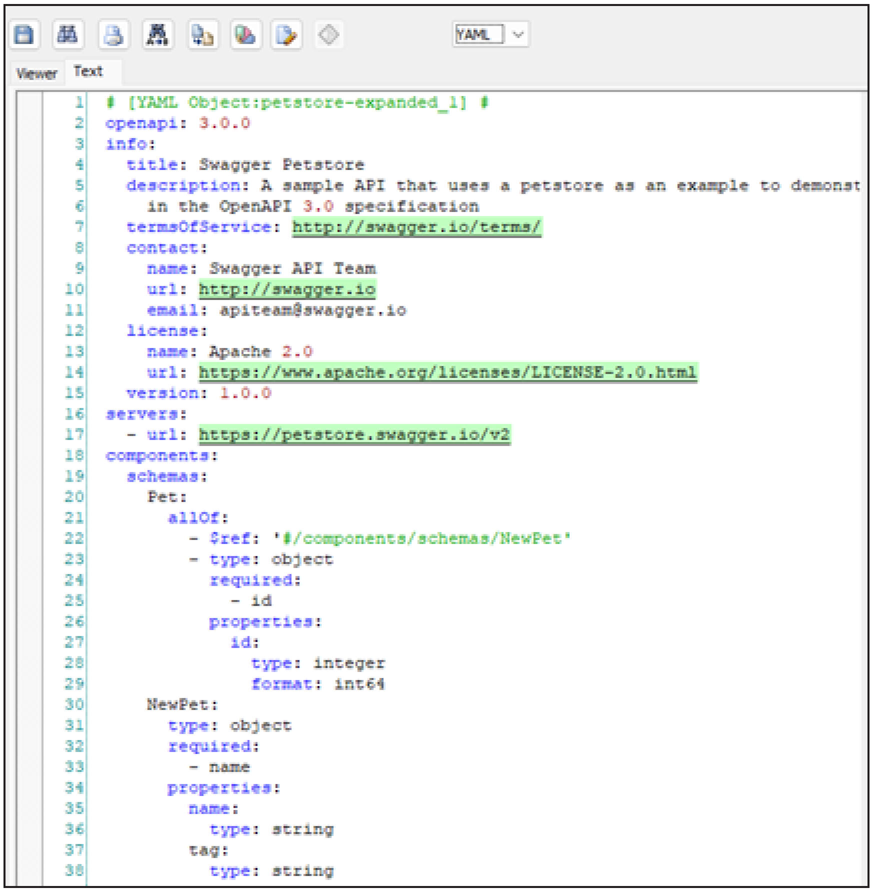
Parameter	Description	Additional Information
Object	Specifies the selected OpenAPI object	

Preview

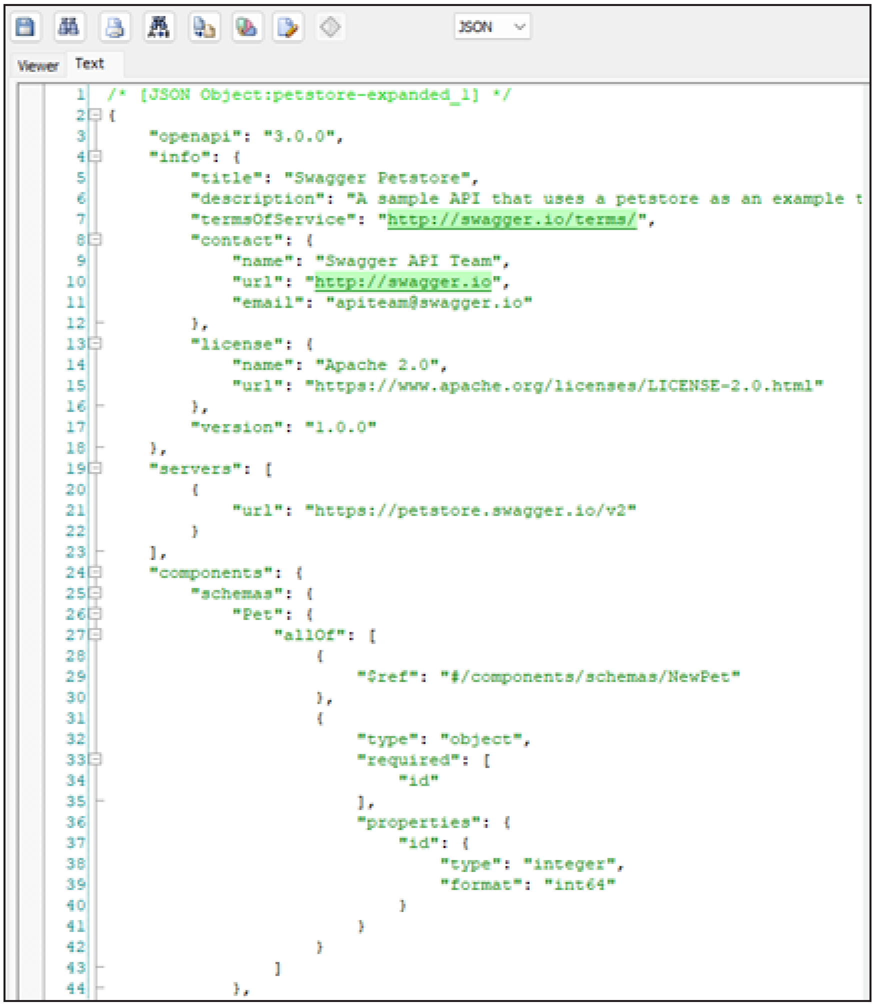
Parameter	Description	Additional Information
Viewer	Displays the OpenAPI schema in the viewer editor	**Collapse All**: Use this option to collapse all the nodes. **Search**: Use this option to search a text entered in the search box. **Find Previous**: Use this option to navigate to previous search string in the search results **Find Next**: Use this option to navigate to next search string in the search result.
Text	Displays the OpenAPI schema in the text editor	Select the file format in which you want to generate the OpenAPI Specification. Supported formats are JSON and YAML. **Save**: Use this option to save the generated schema. **Search**: Use this option to search through the generated schema. **Print**: Use this option to print the generated schema.

Parameter	Description	Additional Information
		Replace: Use this option to find and replace text in the generated schema. **Copy**: Use this option to copy the selected text in the schema. **Text Options**: Use this option to edit window settings, fonts, and syntax color. **Error Check**:Use this option to run an error check. Based on the results, you can correct the generated script.

The following images show the forward engineering script for an OAS model in YAML and JSON formats respectively.



```
# [YAML Object:petstore-expanded_1] #
openapi: 3.0.0
info:
  title: Swagger Petstore
  description: A sample API that uses a petstore as an example to demonst
    in the OpenAPI 3.0 specification
  termsOfService: http://swagger.io/terms/
  contact:
    name: Swagger API Team
    url: http://swagger.io
    email: apiteam@swagger.io
  license:
    name: Apache 2.0
    url: https://www.apache.org/licenses/LICENSE-2.0.html
  version: 1.0.0
servers:
  - url: https://petstore.swagger.io/v2
components:
  schemas:
    Pet:
      allOf:
        - $ref: '#/components/schemas/NewPet'
        - type: object
          required:
            - id
          properties:
            id:
              type: integer
              format: int64
    NewPet:
      type: object
      required:
        - name
      properties:
        name:
          type: string
        tag:
          type: string
```



```
/* [JSON Object:petstore-expanded_1] */
{
    "openapi": "3.0.0",
    "info": {
        "title": "Swagger Petstore",
        "description": "A sample API that uses a petstore as an example t
        "termsOfService": "http://swagger.io/terms/",
        "contact": {
            "name": "Swagger API Team",
            "url": "http://swagger.io",
            "email": "apiteam@swagger.io"
        },
        "license": {
            "name": "Apache 2.0",
            "url": "https://www.apache.org/licenses/LICENSE-2.0.html"
        },
        "version": "1.0.0"
    },
    "servers": [
        {
            "url": "https://petstore.swagger.io/v2"
        }
    ],
    "components": {
        "schemas": {
            "Pet": {
                "allOf": [
                    {
                        "$ref": "#/components/schemas/NewPet"
                    },
                    {
                        "type": "object",
                        "required": [
                            "id"
                        ],
                        "properties": {
                            "id": {
                                "type": "integer",
                                "format": "int64"
                            }
                        }
                    }
                ]
            },
```

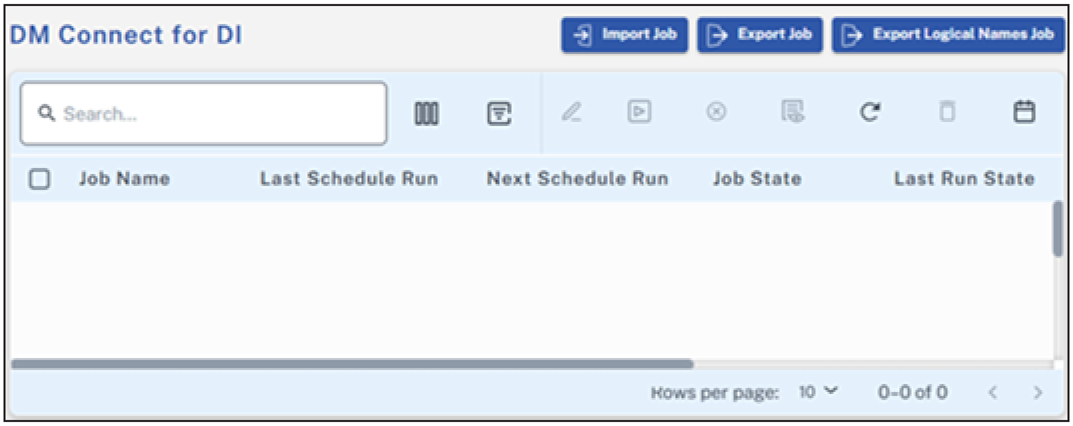
erwin DM-erwin DI Logical Names Mapping

An Export BGM job converts logical names to an erwin DI-compatible format and exports them to the Business Glossary Manager as Business Terms.

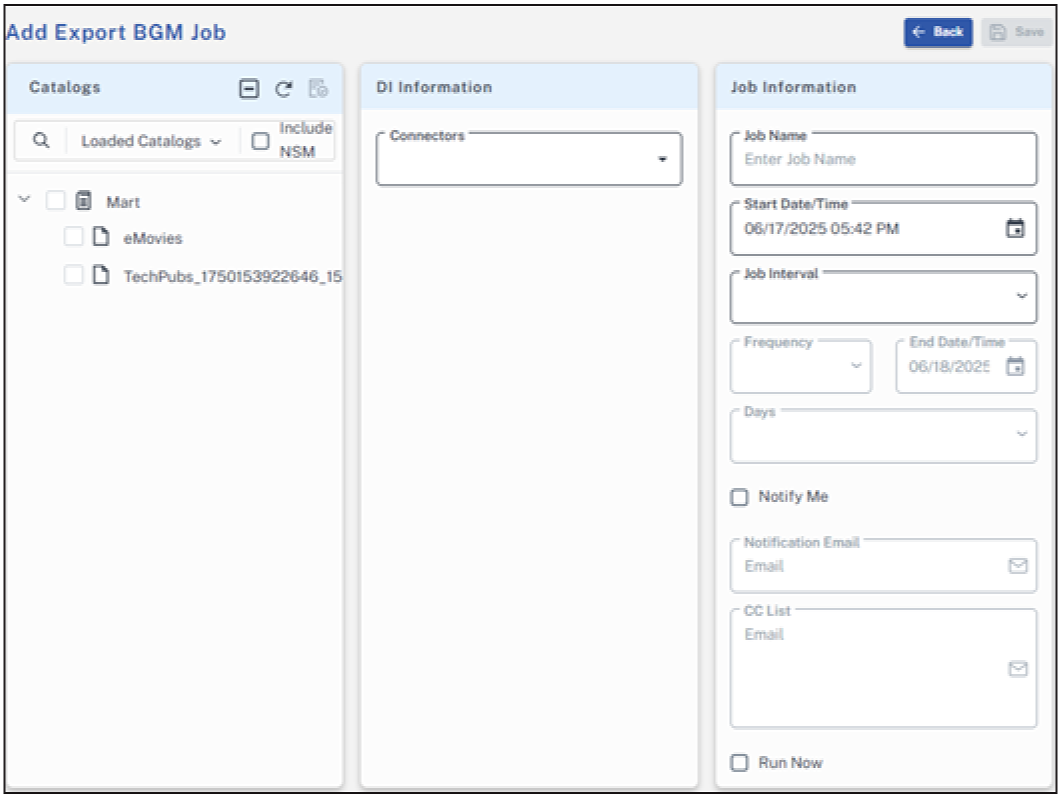
For more information about data sharing between erwin Data Modeler (erwin DM) and erwin Data Intelligence(erwin DI), refer to the Data Sharing topic.

To schedule logical name export jobs, follow these steps:

1. In the header pane, click ▦ and then click **DM Connect for DI**.
   The DM Connect for DI page opens.



2. Click **Export Logical Names Job**.
   The Add Export BGM Job page appears.



3. Set up job parameters as follows:

Tab	Field	Description
Catalogs	Catalog Tree	Select models from catalog to export. Before you select models, you can use the All Catalogs or Loaded Catalogs to display all available catalogs or only the expanded catalogs respectively. Apart from that, after you select catalogs, you can click to view only the selected catalogs in the Catalogs section.
	Include NSM	Select whether naming standards must be exported. A catalog named by NSM file is created under **Business Glossary Manager** > **DM NSM Files** custom

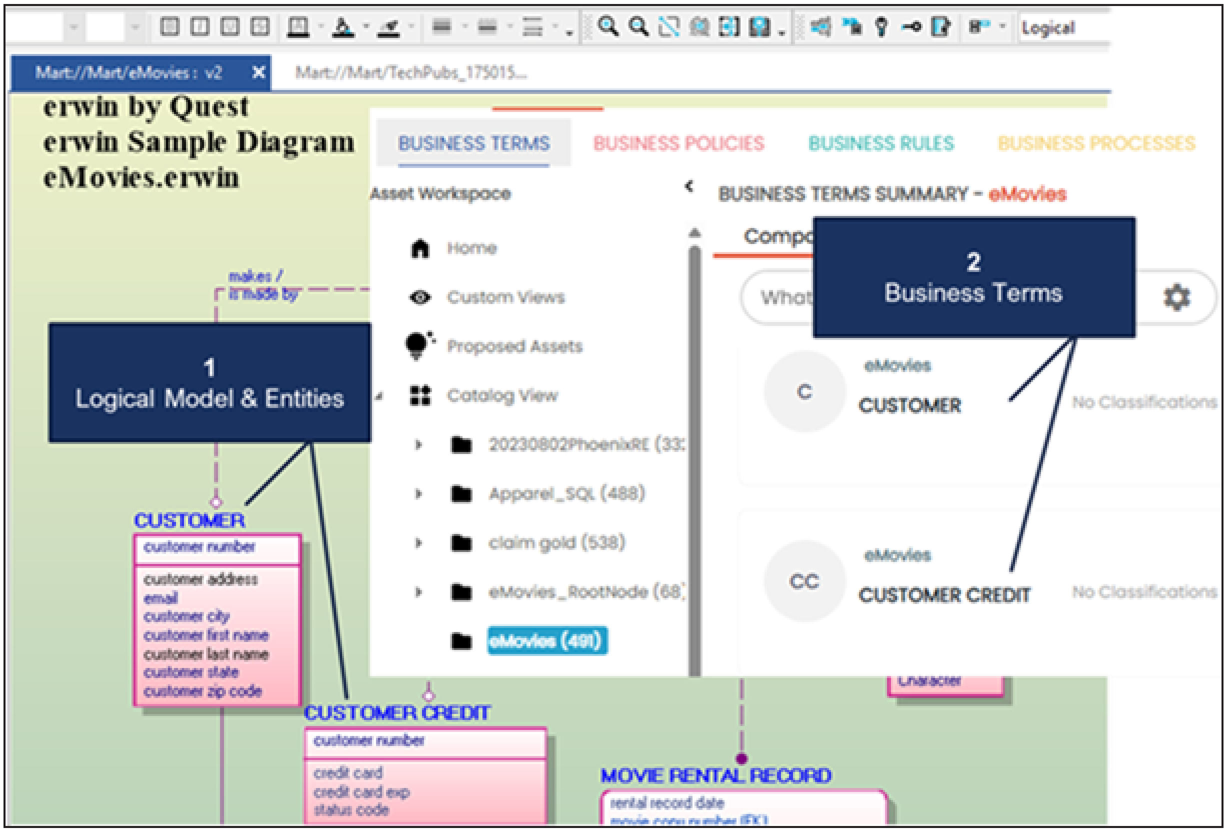
DM-DI-LogicalNamesMapping

Tab	Field	Description
		asset. Ensure that the DM NSM Files asset is available in the Business Glossary Manager.
DI Information	Connectors	Select a configuration to use for the export job.

Tab	Field	Description
Job Information	Job Name	Specify a job name.
	Start Date/Time	Select the date and time at which the job must start.
	Job Interval	Select a suitable frequency at which the job must run. You can set the job to run once or recur daily, weekly, monthly, or yearly. You can also set up custom recurrence for jobs.
	Frequency	Select the hourly frequency at which the job should run. This property is available only when you set the Job Interval to Recurring.
	End Date/Time	If you set up recurring jobs, select the date and time at which the recurrence must end.
	Days	Select the days of the week on which the job should run. The days available here depend on the End Date/Time. This property is available only when you set the Job Interval to Recurring.
	Notify Me	Select the check box to receive a notification when the job status changes. This enables the Notification Email and CC List fields.
	Notification Email	Specify the email address at which you want to receive the notification.
	CC List	Specify a semi-colon-separated list of email addresses that must receive the job notification.
	Run Now	Select the check box to run the job immediately.

4. Click **Save**.
   The job is added to the calendar with its **Job State** set to Scheduled.

The job runs according to the schedule and exports logical names to Business Glossary Manager. For example, the logical names, Customer and Customer Credit, from the eMovies model are saved as business terms in the Business Glossary Manager.



DBT Integration

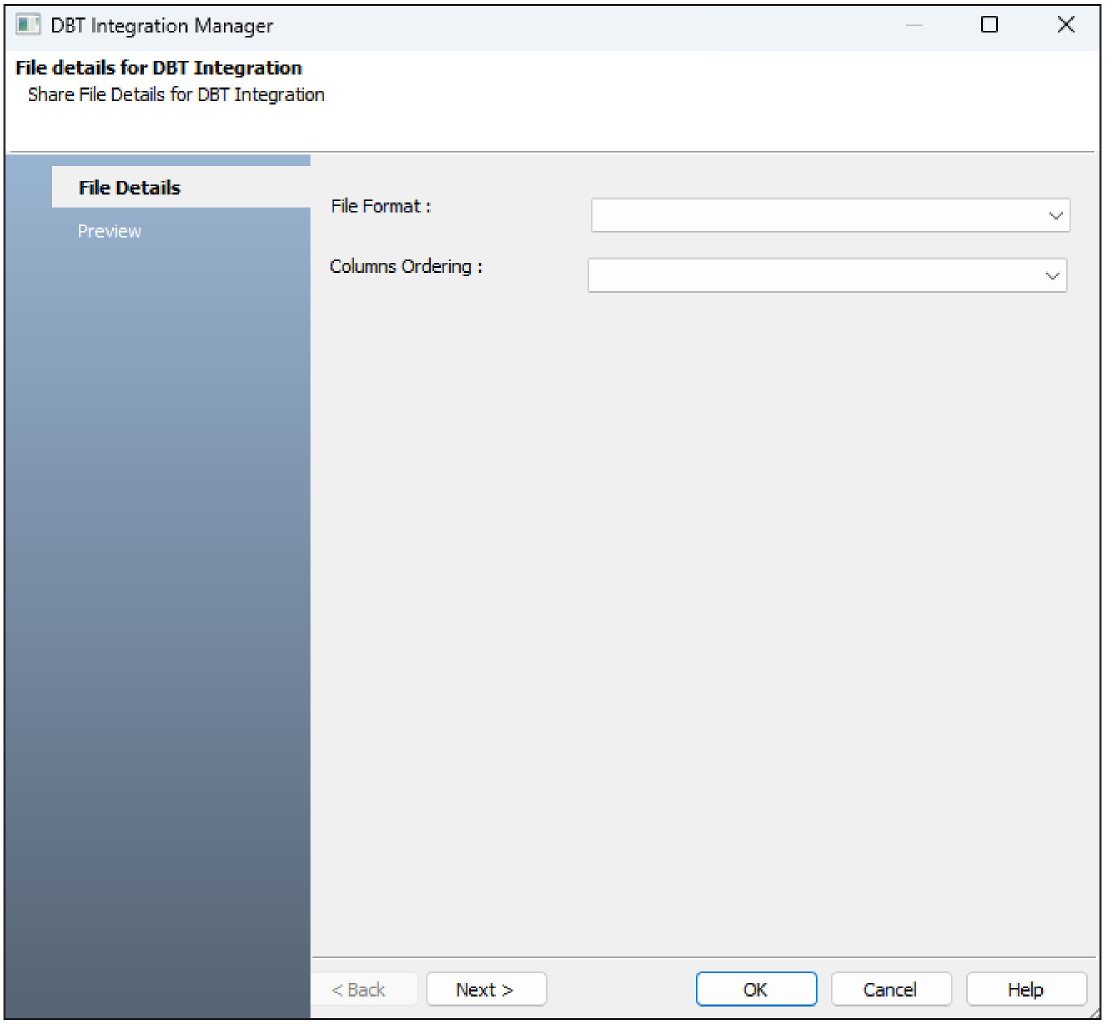
erwin DM now integrates with DBT (Data Build Tool). This feature automatically generates DBT-compatible YAML files from models saved in Mart and opened in erwin DM. You can use these files to create data tables and transformations, improving efficiency and consistency.

Depending on your requirements, use the options in the DBT Integration Manager to define the file format and columns ordering for generating YAML files.

Generating YAML Files

To generate YAML files, follow these steps :

1. On the ribbon, click **Mart** > **DBT Integration**.
2. The DBT Integration Manager opens.



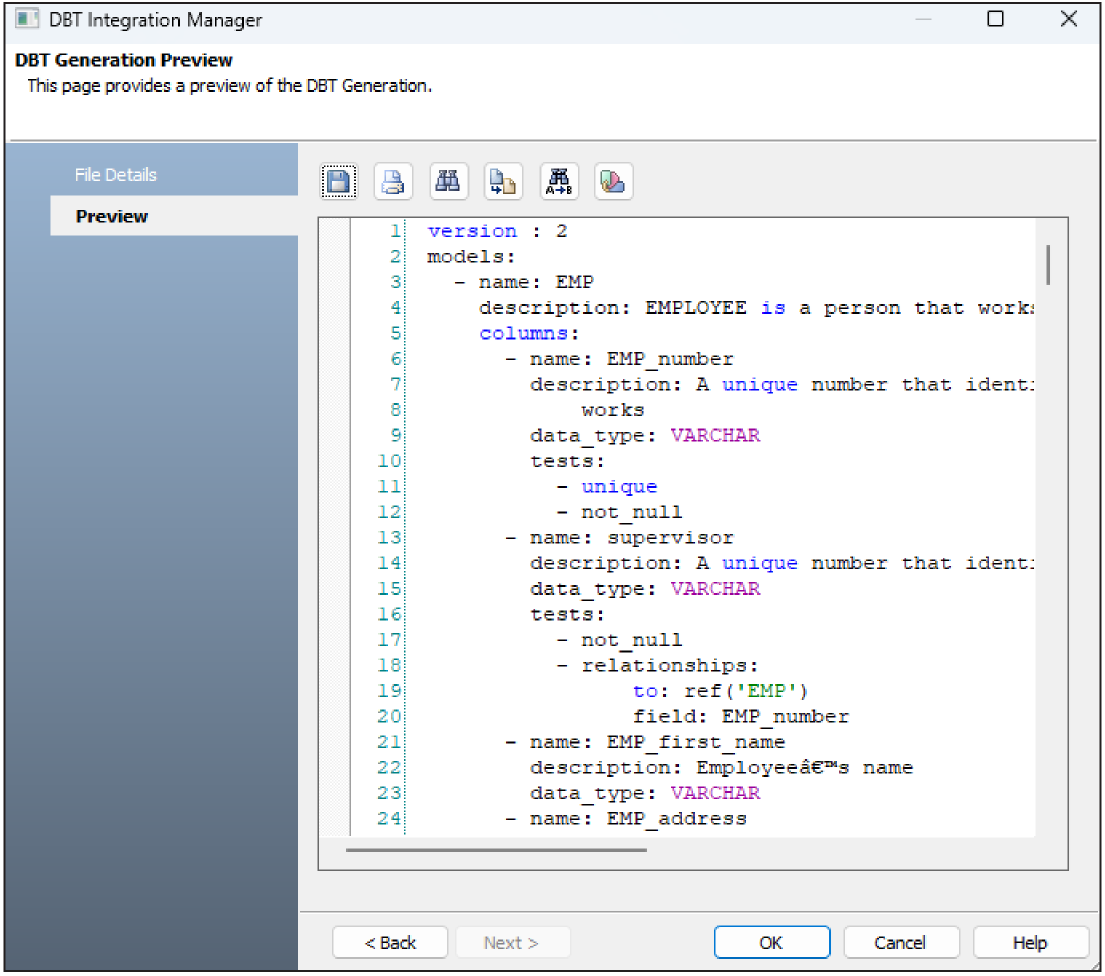
File details for DBT Integration

Share File Details for DBT Integration

By default, the File Details tab opens.

3. Click the File Format drop-down list, and select one of the following options:
   - **Model**: Specifies that the generated YAML file defines the model as a transformational file for DBT, including column metadata, tests, and inter-model relationships.
   - **Source**: Specifies that the generated YAML file defines the model as raw tables, including location, columns, tests, and relationships.

4. Select the preferred order from the Columns Ordering drop-down list:
    - **Attributes_Order_Ref**: Specifies that the column ordering aligns with the logical order of attributes defined in the data model.
    - **Columns_Order_Ref**: Specifies that the column ordering aligns with the physical order in which columns are defined in the table.
    - **Physical_Columns_Order_Ref**: Specifies that the column ordering aligns with the physical layout of columns as stored in the database.
5. Click **Next**.
6. The Preview tab opens and displays the YAML script.



```
version : 2
models:
  - name: EMP
    description: EMPLOYEE is a person that work
    columns:
      - name: EMP_number
        description: A unique number that ident
            works
        data_type: VARCHAR
        tests:
          - unique
          - not_null
      - name: supervisor
        description: A unique number that ident
        data_type: VARCHAR
        tests:
          - not_null
          - relationships:
              to: ref('EMP')
              field: EMP_number
      - name: EMP_first_name
        description: Employeeâ€™s name
        data_type: VARCHAR
      - name: EMP_address
```

Use the following options:

- **Save** (): Use this option to save the generated script in the YAML format. Save this file as DDL.
- **Print** (): Use this option to print the generated script.
- **Search** (): Use this option to search for a word or characters in the schema.
- **Copy** (): Use this option to copy the script.
- **Replace** (): Use this option to find and replace characters in the script.
- **Text Options** (): Use this option to configure the preview text editor's look and feel, such as window, font, and syntax color settings.

7. Click **Ok**.
   The YAML file is generated and saved locally.

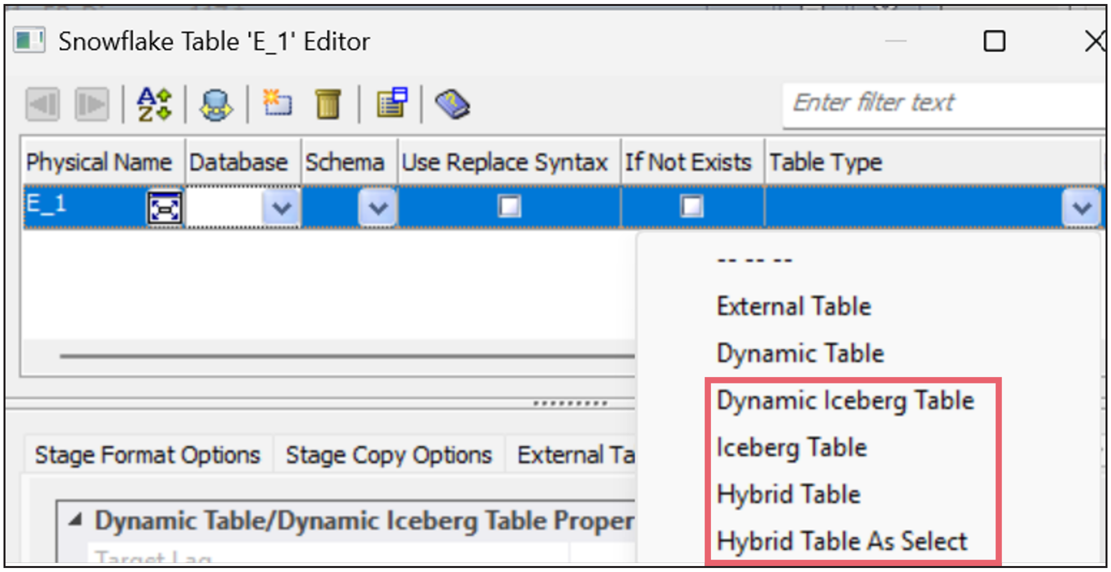
Snowflake Enhancements

The following snowflake objects are now supported:

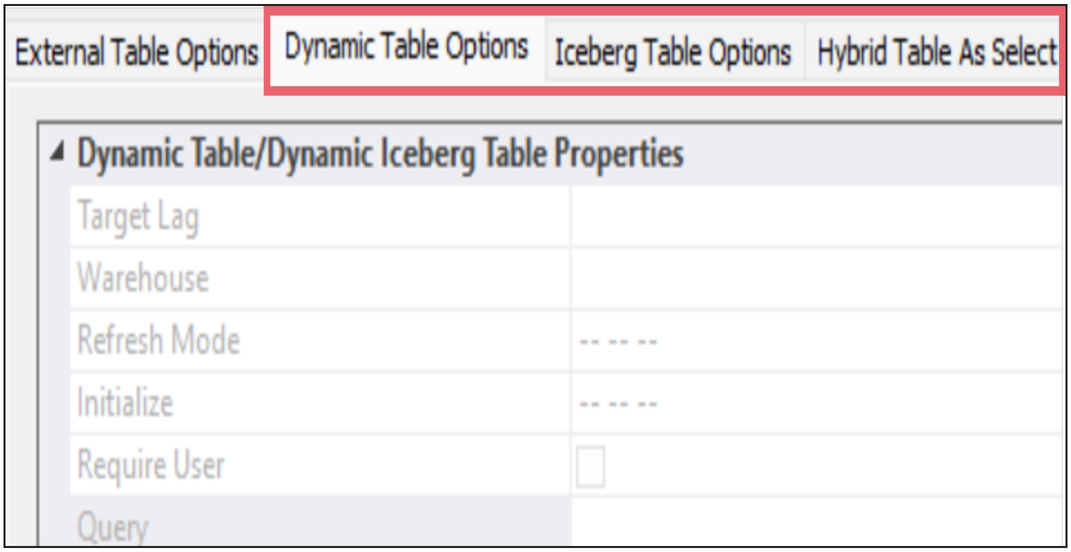
- Masking Policy
- Row Access Policy
- Iceberg Table
- Dynamic Iceberg Table
- Hybrid Table
- Hybrid Table As Select
- Table Index

Snowflake Table Editor

erwin DM 15.0 introduces additional Snowflake table types in the Table Type drop-down list within the Table Editor. The screenshot below displays the newly added table types.



Additionally, the Snowflake Table Editor displays property tabs for each table type. You can configure and manage attributes specific to each table type directly within the editor. These properties appear only when you select a corresponding table type. The screenshot below displays the property tabs for the newly added table types.



Dynamic Table/Dynamic Iceberg Table Properties

Target Lag	
Warehouse	
Refresh Mode	-- -- --
Initialize	-- -- --
Require User	□
Query	

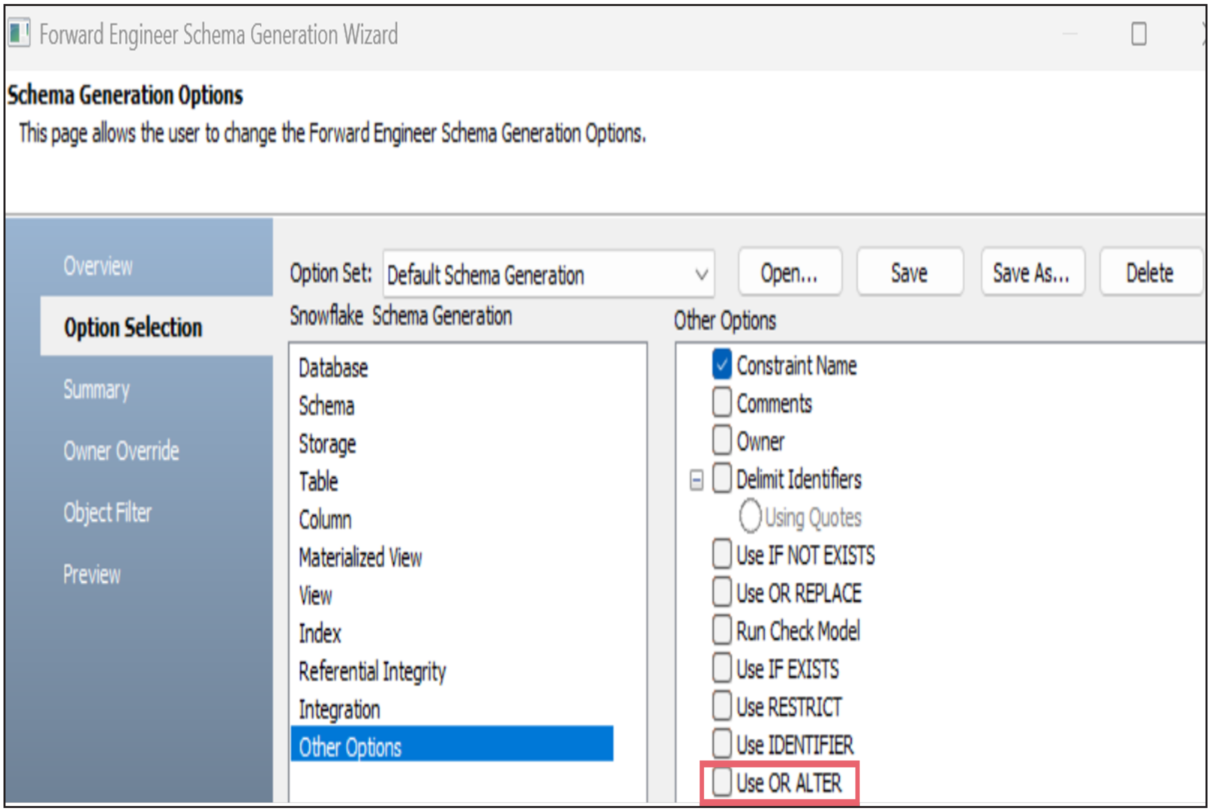
Index Support for Snowflake Hybrid Tables

Hybrid Tables support Unique and Non-Unique Indexes.

Use OR ALTER Option in Forward Engineering

The Forward Engineer Schema Generation Wizard now includes a "Use OR ALTER" checkbox for Snowflake.

Under the Other Options pane, select the **Use OR ALTER** checkbox to generate conditional DDL for Snowflake objects. This option enables you to create or modify objects using a single statement.



Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Default Schema Generation

Snowflake Schema Generation

Database
Schema
Storage
Table
Column
Materialized View
View
Index
Referential Integrity
Integration
Other Options

Other Options

☑ Constraint Name
□ Comments
□ Owner
□ Delimit Identifiers
○ Using Quotes
□ Use IF NOT EXISTS
□ Use OR REPLACE
□ Run Check Model
□ Use IF EXISTS
□ Use RESTRICT
□ Use IDENTIFIER
□ Use OR ALTER

Column and Constraint Support for Iceberg Table

- Retrieves available column metadata during reverse engineering and excludes column definitions during forward engineering for catalog types. For example, AWS Glue.
- Constraints are now generated within the CREATE TABLE statement, as Iceberg Tables do not support ALTER TABLE for constraint additions.

JSON Enhancements

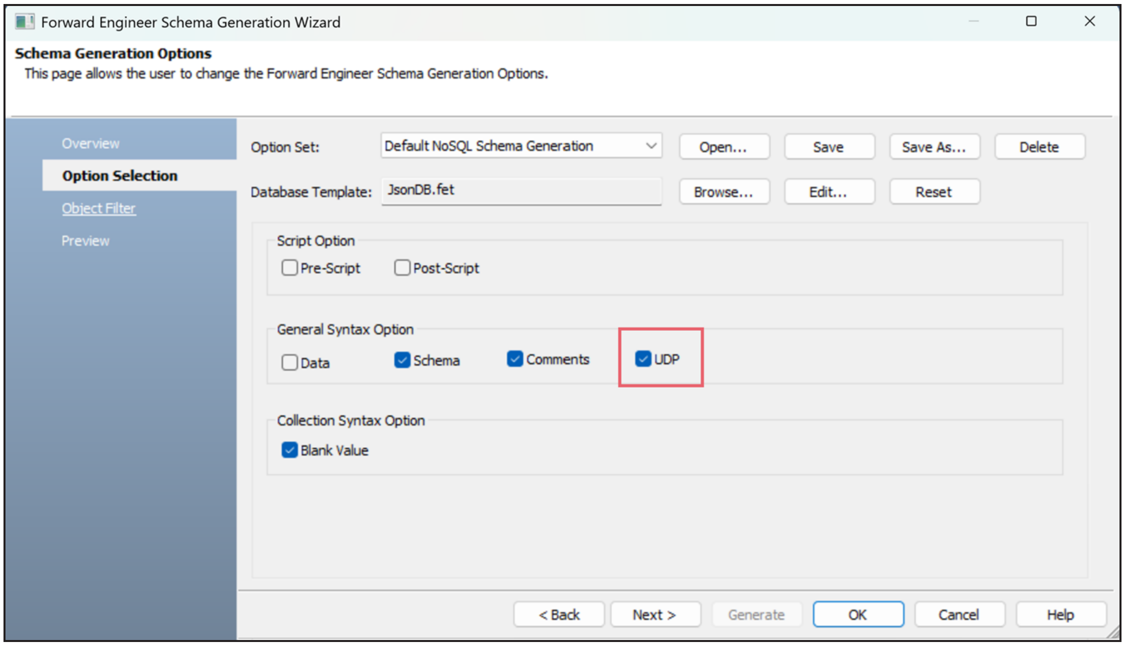
Several enhancements have been implemented for JSON:

- User-Defined Properties in JSON Forward Engineering
- Definition for JSON Fields
- Array Object Type

User-Defined Properties in JSON Forward Engineering

To include user-defined properties in forward engineering, you need to select the following options:

- **UDP**: In the Option Selection section of the Forward Engineer Schema Generation Wizard, select the **UDP** checkbox.



Forward Engineer Schema Generation Wizard

Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Default NoSQL Schema Generation

Database Template: JsonDB.fet

Script Option

□ Pre-Script □ Post-Script

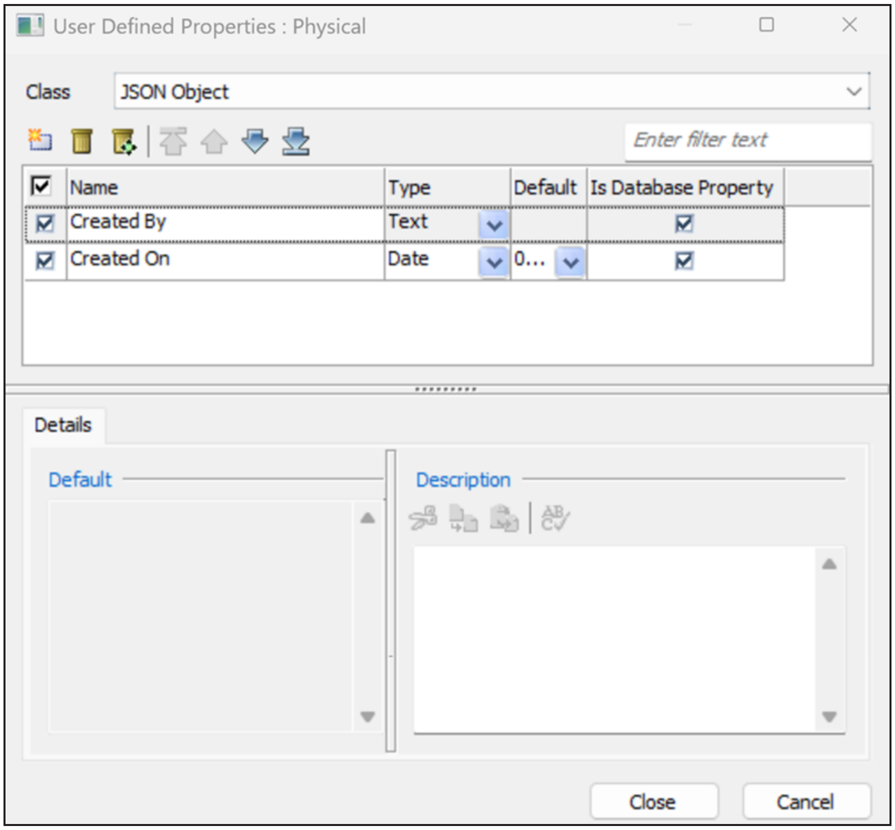
General Syntax Option

□ Data ☑ Schema ☑ Comments ☑ UDP

Collection Syntax Option

☑ Blank Value

- **Is Database Property**: In the User Defined Properties editor, select the **Is Database Property** checkbox .

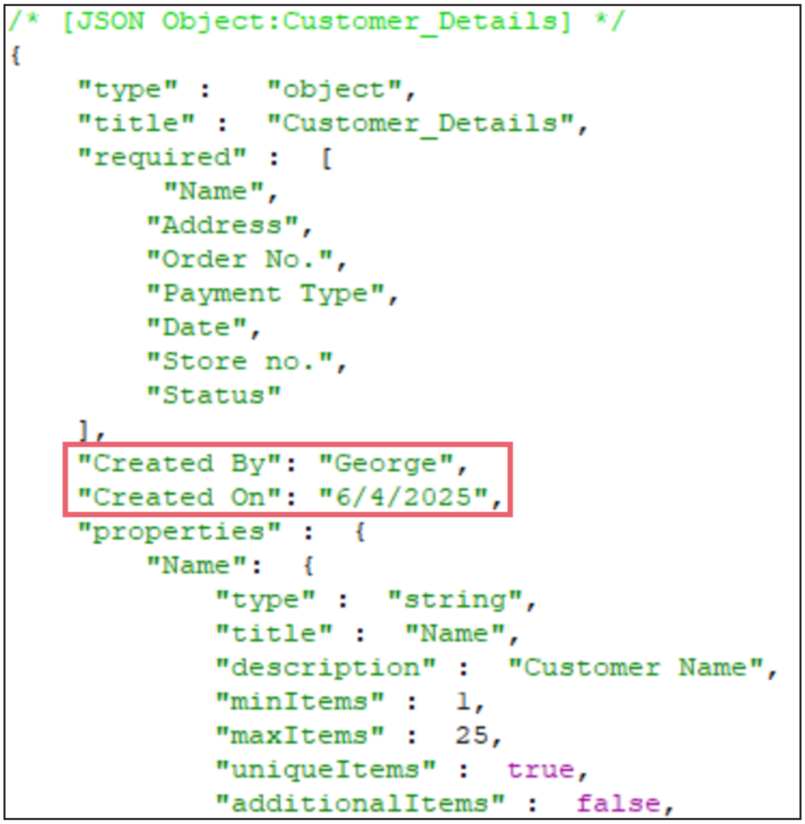


User Defined Properties : Physical

Class: JSON Object

☑	Name	Type	Default	Is Database Property
☑	Created By	Text		☑
☑	Created On	Date	0...	☑

When you select these options and generate a script for a JSON model, user-defined properties are also generated as displayed in the following image.



```
/* [JSON Object:Customer_Details] */
{
    "type" :   "object",
    "title" :  "Customer_Details",
    "required" :  [
         "Name",
        "Address",
        "Order No.",
        "Payment Type",
        "Date",
        "Store no.",
        "Status"
    ],
    "Created By": "George",
    "Created On": "6/4/2025",
    "properties" :  {
        "Name":  {
            "type" :  "string",
            "title" :  "Name",
            "description" :  "Customer Name",
            "minItems" :  1,
            "maxItems" :  25,
            "uniqueItems" :  true,
            "additionalItems" :  false,
```

Definitions for JSON Fields

You can now create a list of reusable definitions using the predefined keyword, $Defs. These definitions can then be assigned to fields within the object in which they are created, or to fields in other objects, depending on the definition type.

To create JSON definition libraries with a predefined field, follow these steps:

1. In the Model Explorer, double-click the object where you want to create a definition library.
2. Right-click the Fields node and click **New**.
   An instance of the field is created.
3. Name the field as $Defs.
4. Right-click the $Defs field and click **Properties**.
   The field's property editor opens.
5. On the General tab, set the values of all the required properties.

   Physical Data Type

   Specifies the data type for fields. This must be Object for the $Defs field.

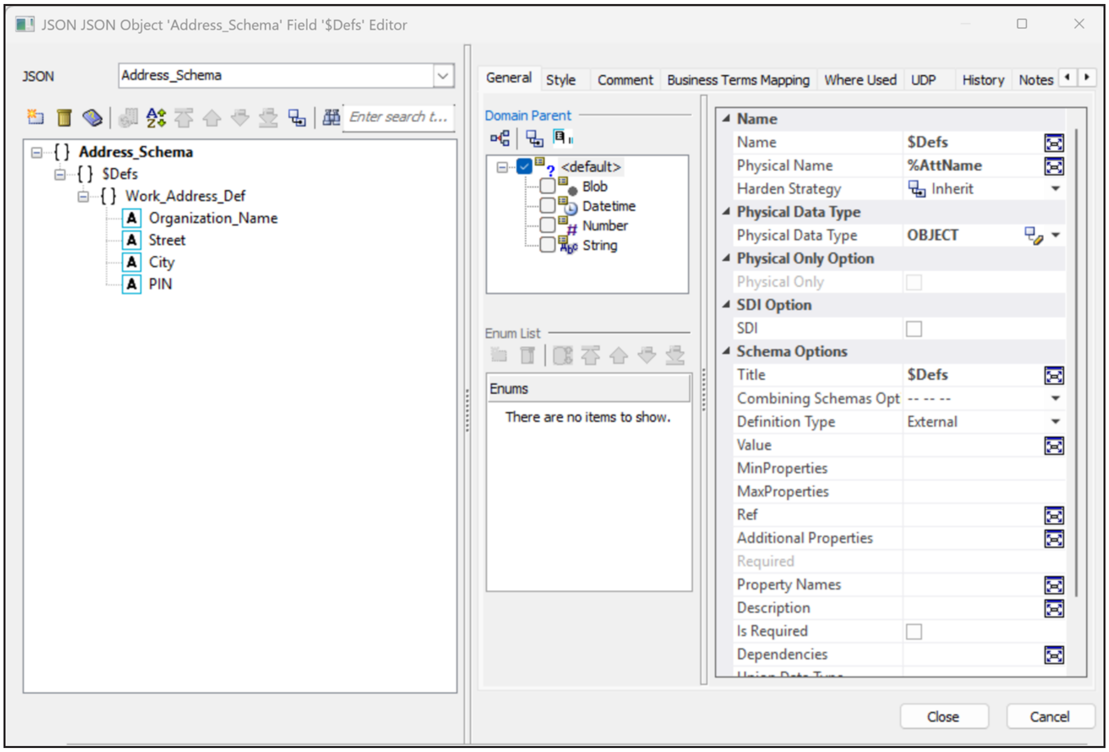
   Definition Type

   Specifies the definition of the field. This option is available only for the $Defs fields and includes following options:

   - **External**: Select this option to apply the definition to fields across all objects in the model.
   - **Internal**: Select this option to apply the definition only to fields within the object they were created in.
6. Right-click the $Defs field and click **New**.

   An instance of the definition is created.
7. Name the definition as required.

Similarly, you can create a list of definitions.



JSON JSON Object 'Address_Schema' Field '$Defs' Editor

JSON: Address_Schema

- { } Address_Schema
  - { } $Defs
    - { } Work_Address_Def
      - A Organization_Name
      - A Street
      - A City
      - A PIN

Domain Parent

- ☑ <default>
  - □ Blob
  - □ Datetime
  - □ Number
  - □ String

Enum List

Enums: There are no items to show.

Name	
Name	$Defs
Physical Name	%AttName
Harden Strategy	Inherit
Physical Data Type	
Physical Data Type	OBJECT
Physical Only Option	
Physical Only	□
SDI Option	
SDI	□
Schema Options	
Title	$Defs
Combining Schemas Opt	-- -- --
Definition Type	External
Value	
MinProperties	
MaxProperties	
Ref	
Additional Properties	
Required	
Property Names	
Description	
Is Required	□
Dependencies	

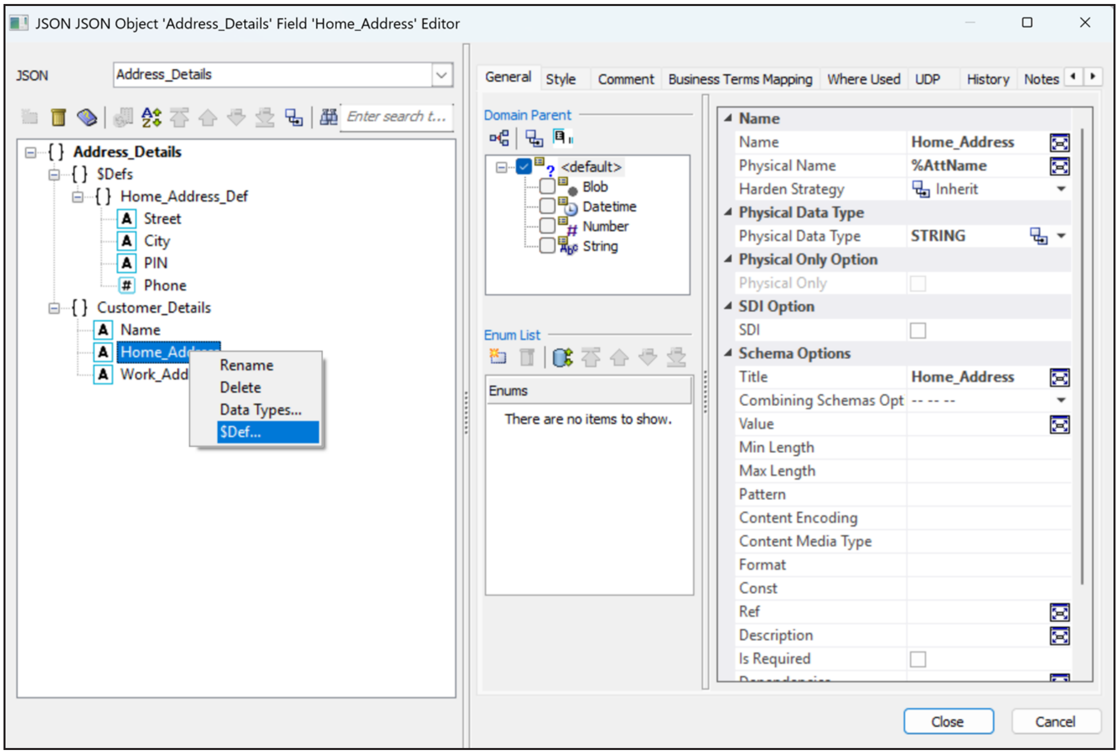
For information on properties, refer to the Defining JSON Fields topic.

> The Physical Data Type for the $Defs fields must be Object. And a JSON object cannot contain both internal and external definitions.

Assigning JSON Definitions to Fields

Follow these steps to assign a JSON definition to fields:

1. In the JSON Field Editor, right-click the required field node and click **$Defs**.



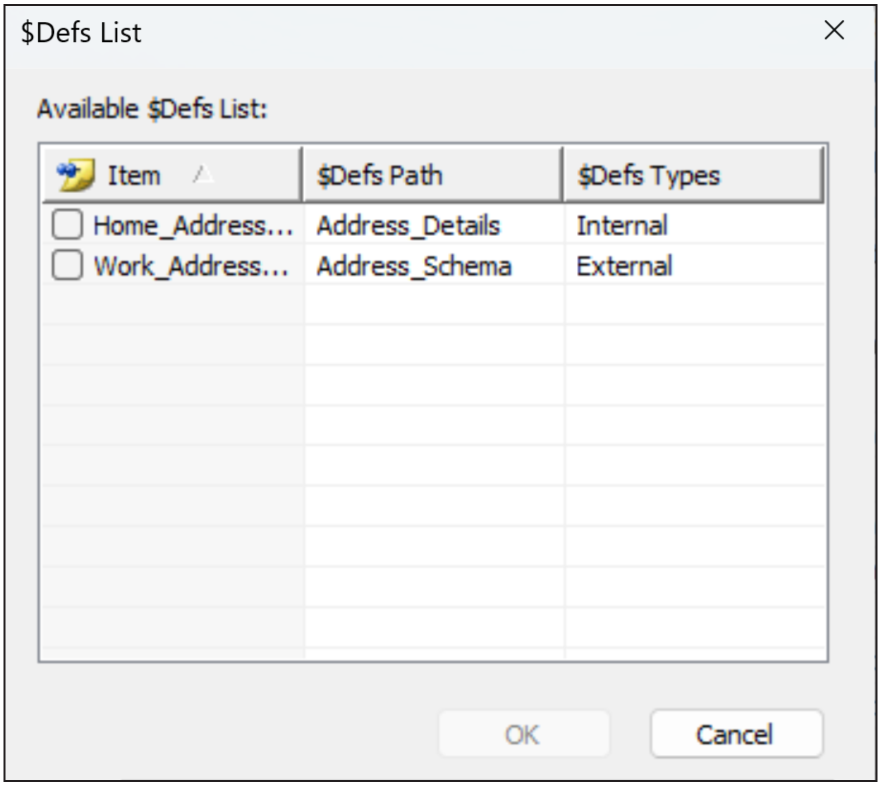
JSON JSON Object 'Address_Details' Field 'Home_Address' Editor

JSON: Address_Details

- { } Address_Details
  - { } $Defs
    - { } Home_Address_Def
      - A Street
      - A City
      - A PIN
      - # Phone
  - { } Customer_Details
    - A Name
    - A Home_Address
    - A Work_Add

Name	
Name	Home_Address
Physical Name	%AttName
Harden Strategy	Inherit
Physical Data Type	
Physical Data Type	STRING
Physical Only Option	
Physical Only	□
SDI Option	
SDI	□
Schema Options	
Title	Home_Address
Combining Schemas Opt	-- -- --
Value	
Min Length	
Max Length	
Pattern	
Content Encoding	
Content Media Type	
Format	
Const	
Ref	
Description	
Is Required	□

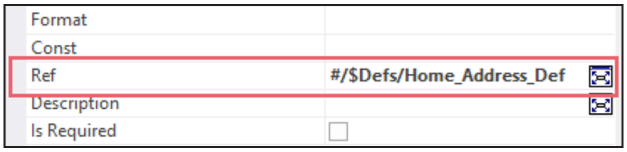
2. On the $Defs List window, select the definition you want to use and then click **OK**.



Available $Defs List:

Item	$Defs Path	$Defs Types
☐ Home_Address...	Address_Details	Internal
☐ Work_Address...	Address_Schema	External

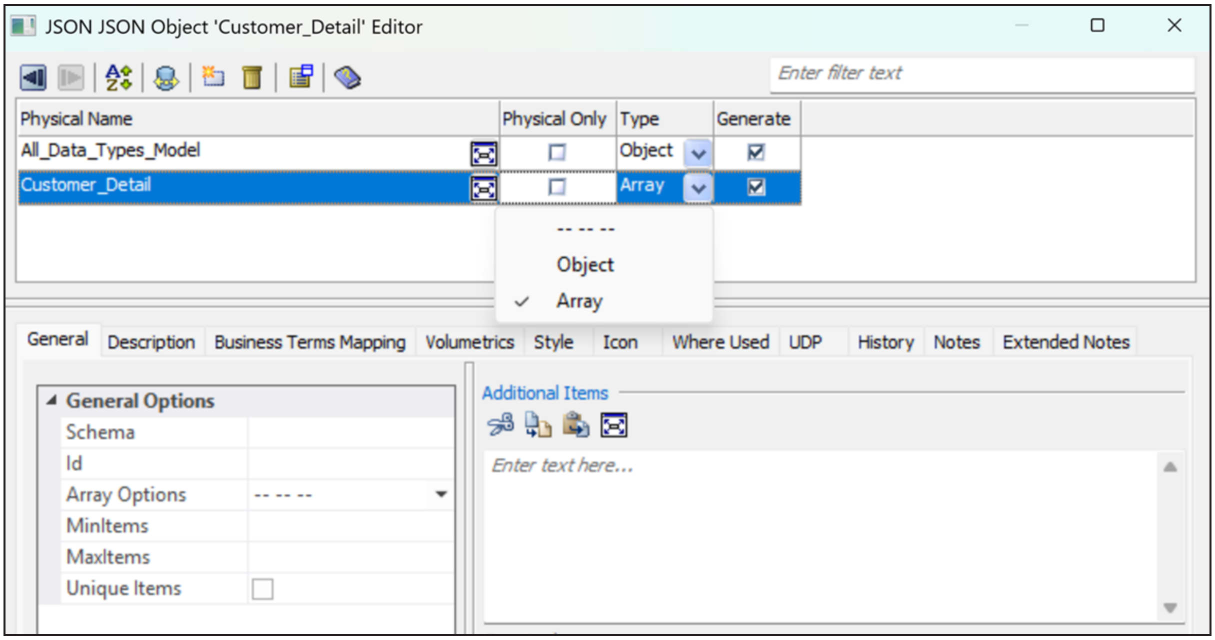
The selected definition is assigned to the field. When you select the field, the Ref property displays the assigned definition.



Format	
Const	
Ref	#/$Defs/Home_Address_Def
Description	
Is Required	☐

Array Object Type

JSON models now support Array type for objects



Physical Name	Physical Only	Type	Generate
All_Data_Types_Model	☐	Object	☑
Customer_Detail	☐	Array	☑

General Options

Schema	
Id	
Array Options	-- -- --
MinItems	
MaxItems	
Unique Items	☐

Google BigQuery Enhancements

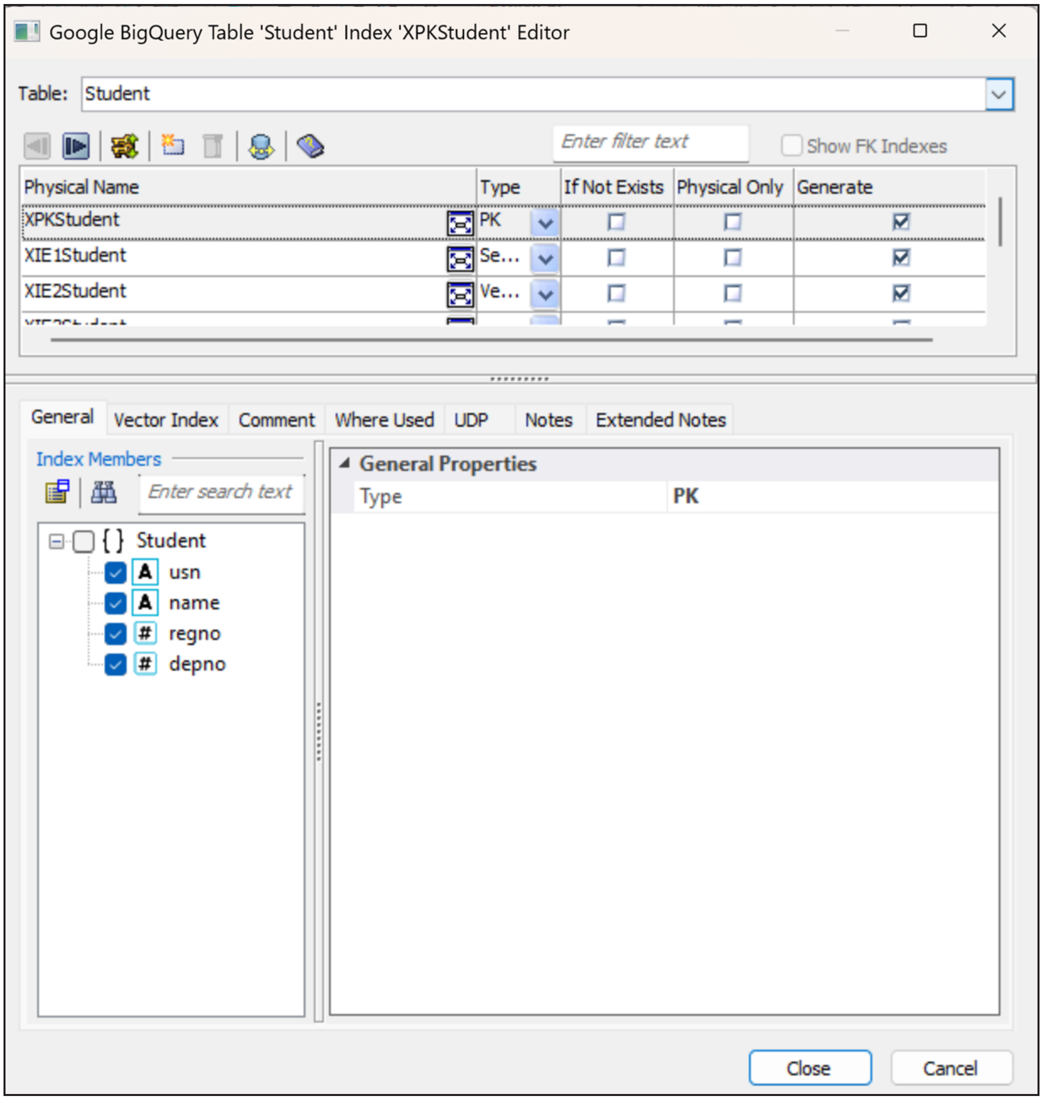
Several enhancements have been implemented for Google BigQuery:

- Primary Key Type
- Comprehensive Column Sorting

Primary Key Type

Google BigQuery models now support the PK (primary key) type and display it in the Index Editor. Earlier, primary key was available only in the Column Editor. The information is synchronized between both editors.

Apart from Primary Key, Search, and Vector are other supported types. You can create multiple search and vector indexes in a table, but each table can generate only one search index and one vector index at a time.



Google BigQuery Table 'Student' Index 'XPKStudent' Editor

Table: Student

☐ Show FK Indexes

Physical Name	Type	If Not Exists	Physical Only	Generate
XPKStudent	PK	☐	☐	☑
XIE1Student	Se...	☐	☐	☑
XIE2Student	Ve...	☐	☐	☑

Index Members

- ☐ { } Student
  - ☑ A usn
  - ☑ A name
  - ☑ # regno
  - ☑ # depno

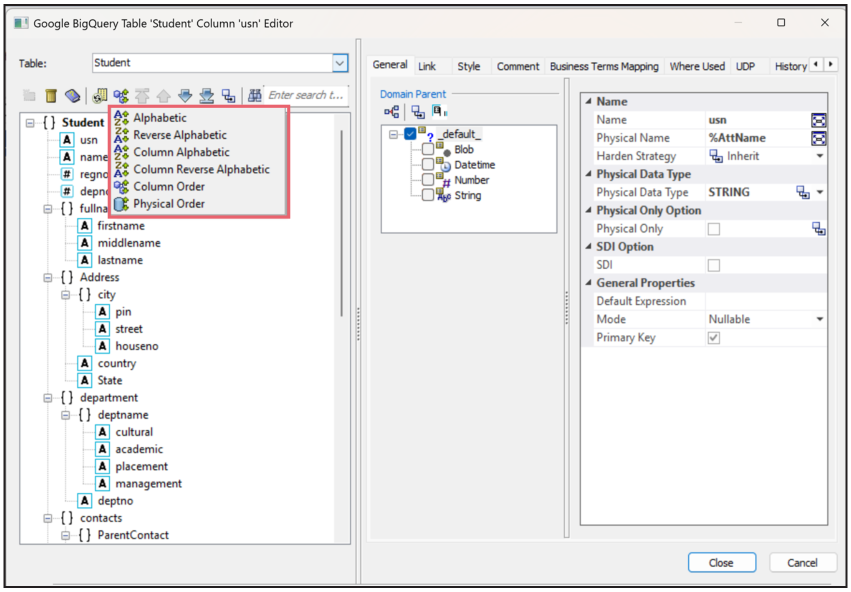
General Properties

Type	PK

Comprehensive Column Sorting

The Sort feature in the Google BigQuery Column Editor now lets you sort columns in the different ways to help you organize and analyze your data.

You can select the method you want using the drop-down menu that opens after you click the Sort button.



Name
- Name: usn
- Physical Name: %AttName
- Harden Strategy: Inherit

Physical Data Type
- Physical Data Type: STRING

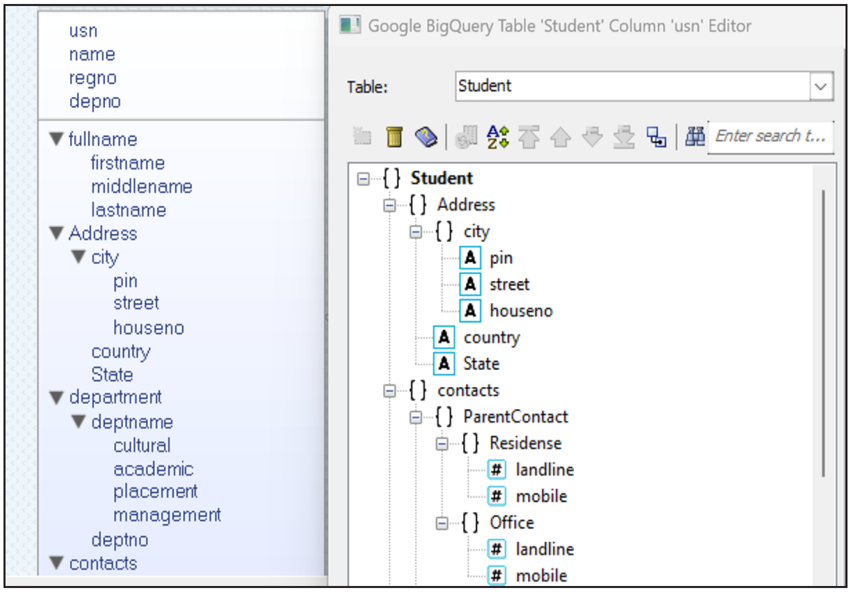
Physical Only Option
- Physical Only: □

SDI Option
- SDI: □

General Properties
- Default Expression:
- Mode: Nullable
- Primary Key: ☑

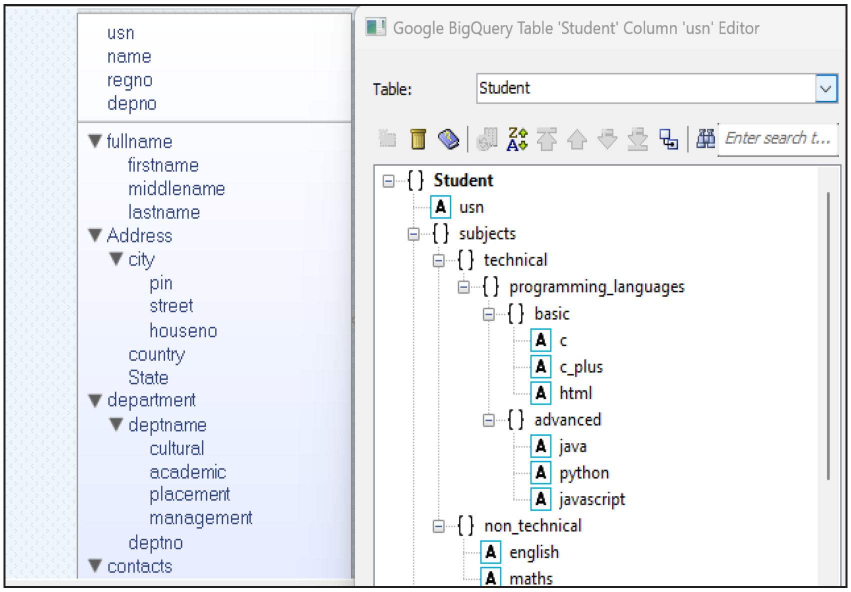
Alphabetic

Specifies that the column list is sorted in alphabetic order. This is applicable only in the Column Editor and is not reflected in the ER diagram.



Reverse Alphabetic

Specifies that the column list is sorted in reverse alphabetic order. This is applicable only in the Column Editor and is not reflected in the ER diagram.

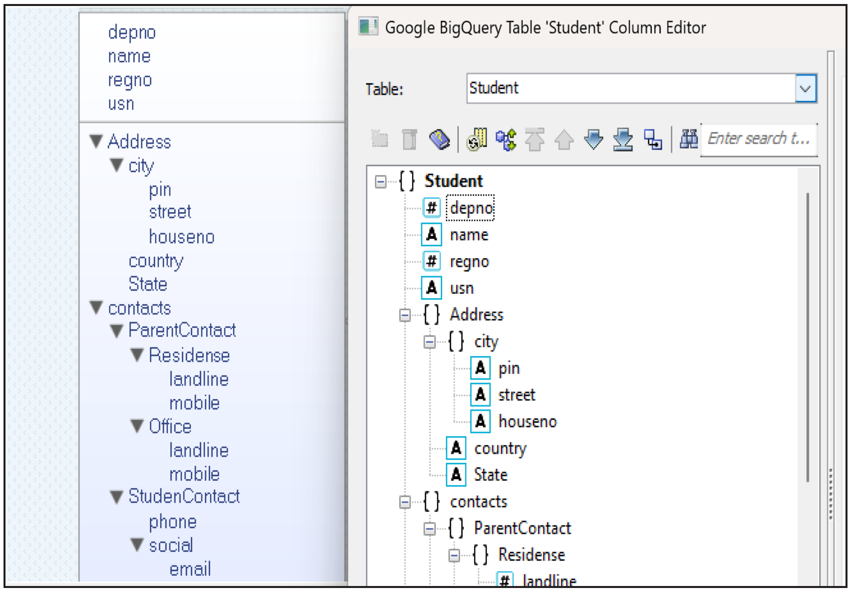


Google BigQuery Table 'Student' Column 'usn' Editor

Table: Student

Column Alphabetic

Specifies that the column list is sorted in alphabetic order. This is applicable in the Column Editor and the ER Diagram.

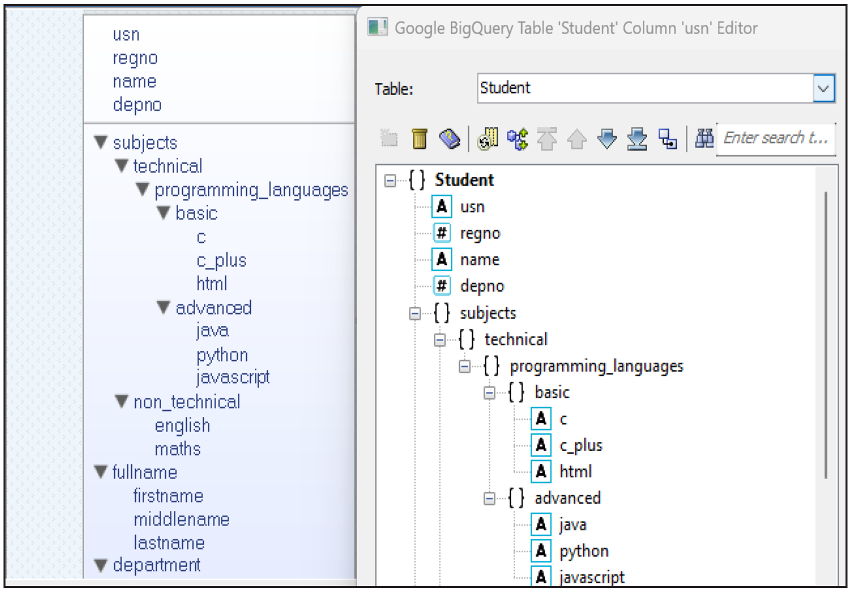


Google BigQuery Table 'Student' Column Editor

Table: Student

Column Reverse Alphabetic

Specifies that the column list is sorted in reverse alphabetic order. This is applicable in the Column Editor and the ER Diagram.



Google BigQuery Table 'Student' Column 'usn' Editor

Table: Student

Column Order

Specifies that the column list is sorted by the current column order. This updates the order for any movements in the sequence of columns.

Physical Order

Specifies that the column list is sorted in physical order. This applies the original column order defined when the table was created.

Primary key columns and non-key columns are sorted separately when you apply column sorting. You can also sort nested columns under each node independently. To sort nested columns under a node, select the node, click **Sort**, and select the sort order from the drop-down menu.

For all NoSQL tables except Google BigQuery only four types of column sorting options are available: Alphabetic, Reverse Alphabetic, Column Alphabetic, and Column Reverse Alphabetic. For Google BigQuery tables, all six sorting options listed above are available.

PostgreSQL Enhancements

Several enhancements have been introduced to PostgreSQL models as follows:

- You can now assign multiple permissions at once to PostgreSQL objects using checkboxes.
- erwin DM now supports permission object for Stored Procedures in PostgreSQL. You can assign execution rights and manage access directly within the model.

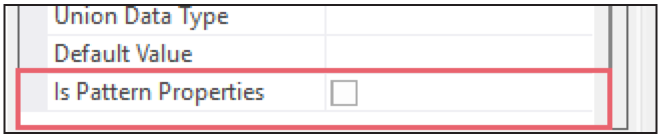
Productivity and UI Enhancements

Several additions and enhancements have been implemented to improve erwin Data Modeler's productivity and usage experience. These enhancements are:

- JSON Field Editor Property
- ER Diagram Property
- PostgreSQL Permission Editor
- Upgraded CDM Models

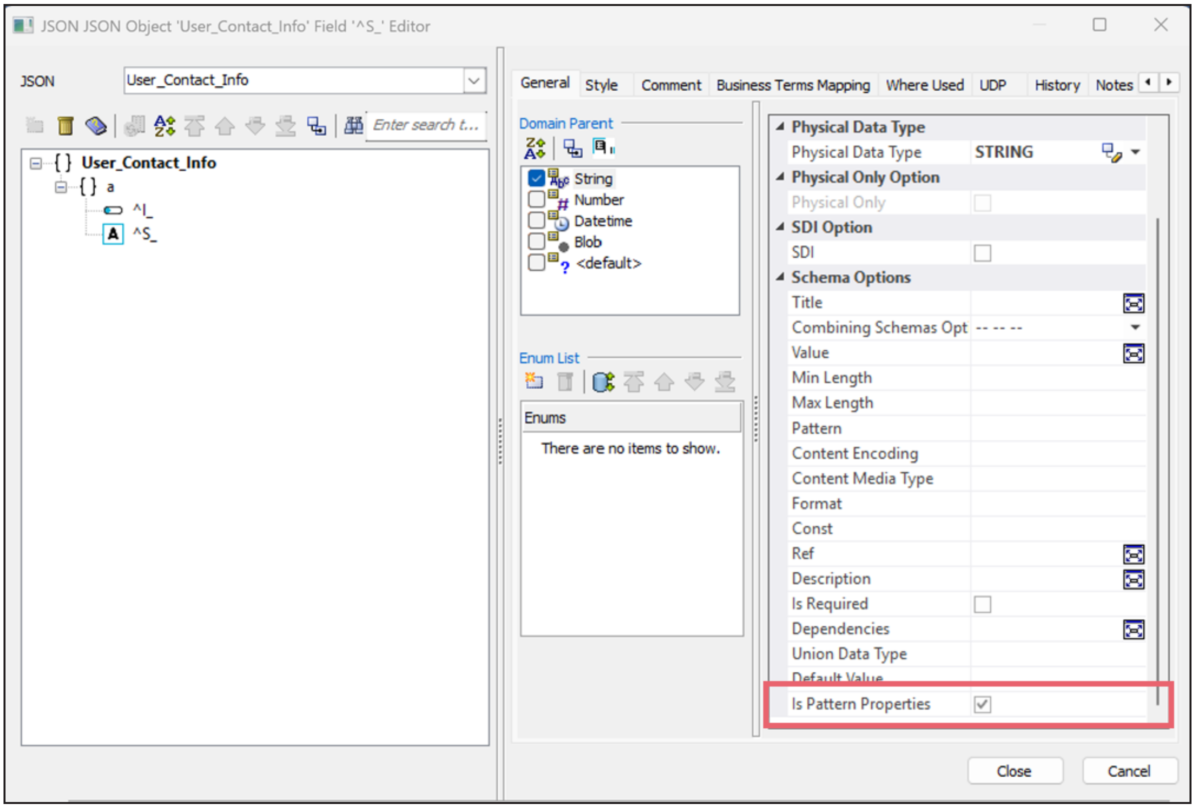
JSON Field Editor Property

The Pattern Properties option has been removed from the JSON Object Editor and added to the Field Editor as the Is Pattern Properties checkbox. When selected, this specifies that the field contains pattern properties that define a regular expression and schema for validating additional properties.



Union Data Type	
Default Value	
Is Pattern Properties	□

You can use this option to define regular expression rules that validate data before processing.



JSON: User_Contact_Info

- { } User_Contact_Info
  - { } a
    - ^I_
    - A ^S_

Domain Parent:

- ☑ String
- □ Number
- □ Datetime
- □ Blob
- □ <default>

Enums: There are no items to show.

Physical Data Type

Physical Data Type	STRING

Physical Only Option

Physical Only	□

SDI Option

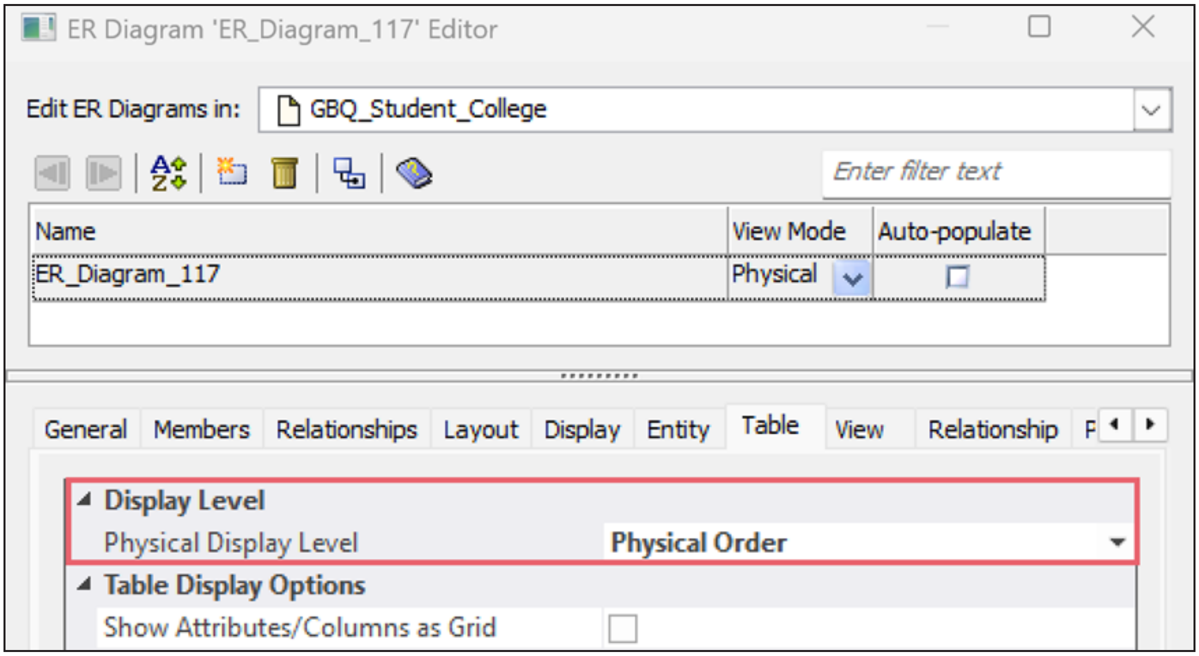
SDI	□

Schema Options

Title	
Combining Schemas Opt	-- -- --
Value	
Min Length	
Max Length	
Pattern	
Content Encoding	
Content Media Type	
Format	
Const	
Ref	
Description	
Is Required	□
Dependencies	
Union Data Type	
Default Value	
Is Pattern Properties	☑

ER Diagram Property

In the ER Diagram Editor, the Physical Order option under Physical Display Level is now available only for SQL databases and Google BigQuery. It is no longer available for other NoSQL databases.


ER Diagram 'ER_Diagram_117' Editor

Edit ER Diagrams in: GBQ_Student_College

Name	View Mode	Auto-populate
ER_Diagram_117	Physical	□

Display Level

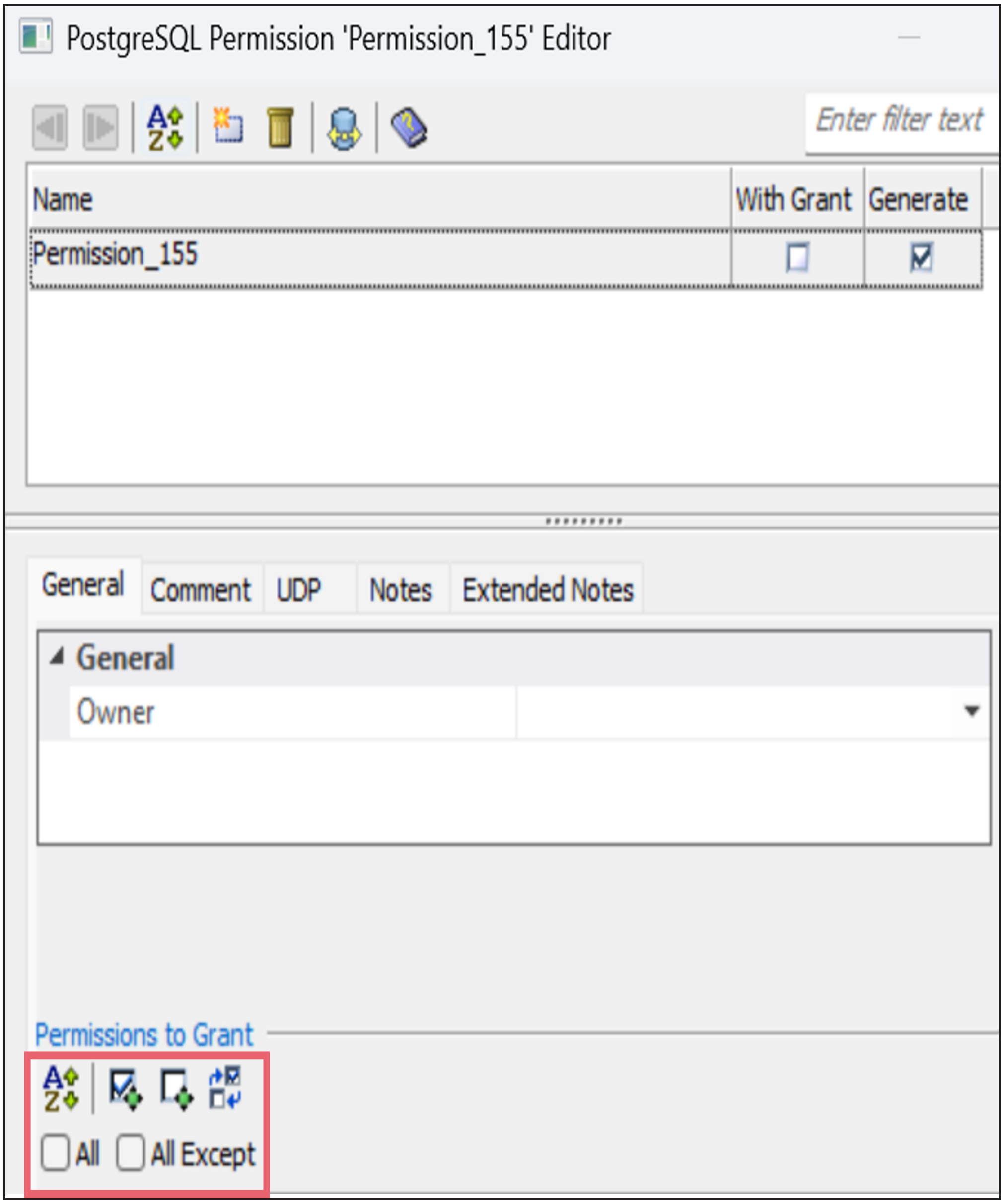
Physical Display Level: Physical Order

Table Display Options

Show Attributes/Columns as Grid □

PostgreSQL Permission Editor

The PostgreSQL Permission Editor now includes new icons such as Sort Items, Select All, Select None, and Toggle Selection, along with the All and All Except options.



Name	With Grant	Generate
Permission_155	☐	☑

General

Owner

☐ All ☐ All Except

Using this options, you can perform the following actions:

- **Sort Items** (): Sort the permission list in alphabetical and reverse alphabetical order.
- **Select All** (): Select all available permissions at once.
- **Select None** (): Deselect all selected permissions.

- **Toggle Selection (⇄)**: Reverse the current selection state of each permission.
- **All**: Select this checkbox to grant all available permissions.
- **All Except**: Select this checkbox to grant all permissions except the ones currently selected.

Upgraded CDM Models

CDM models have been upgraded to ensure compatibility with new features and improved performance.

erwin Mart Portal Enhancements

erwin Mart Portal has undergone the following enhancements:

- DM Connect for DI-Logical Names Export Jobs
- Productivity Enhancements

DM Connect for DI

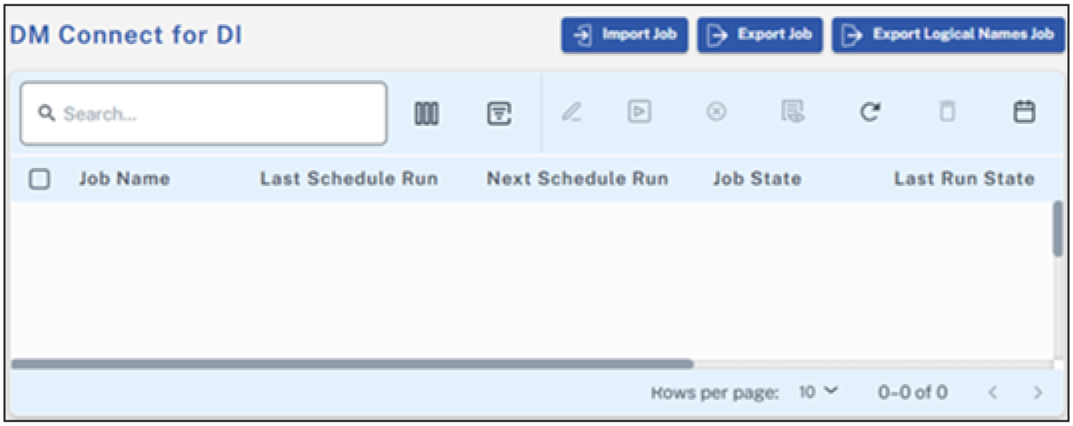
The DM Connect for DI feature has been upgraded to support erwin DI 15.0. Also, you can now map logical names to business terms in the Business Glossary Manager via DM Connect for DI module.

An Export BGM job converts logical names to an erwin DI-compatible format and exports them to the Business Glossary Manager as Business Terms.

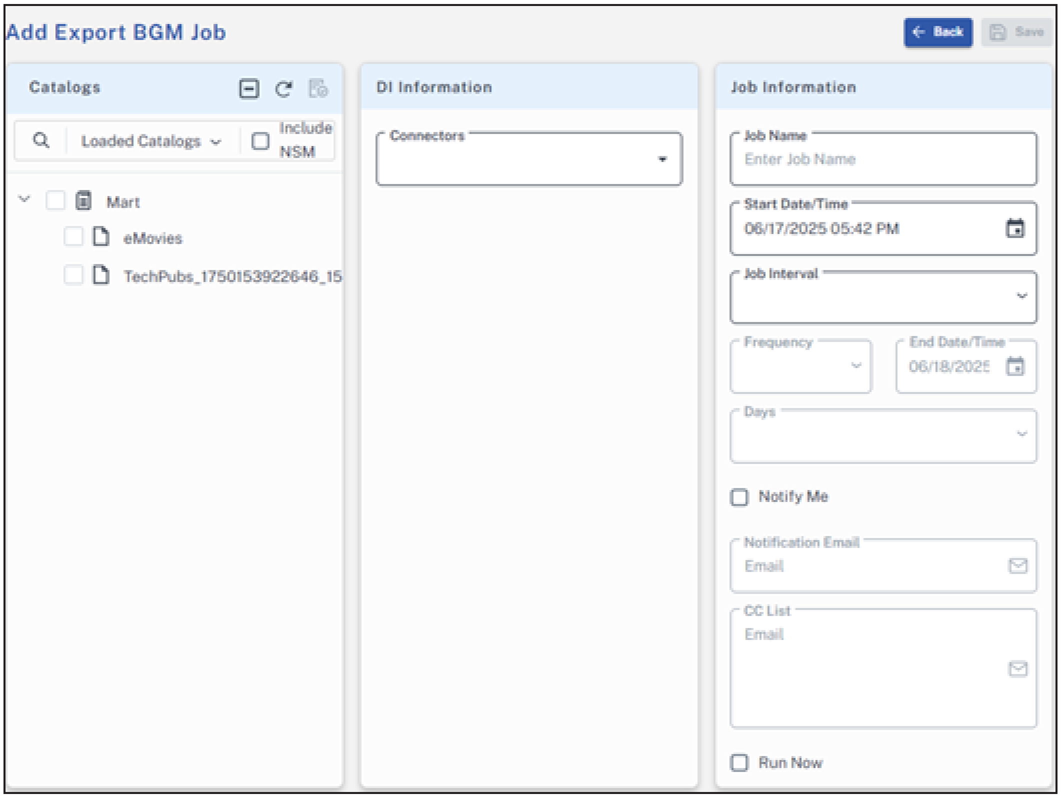
For more information about data sharing between erwin Data Modeler (erwin DM) and erwin Data Intelligence(erwin DI), refer to the Data Sharing topic.

To schedule logical name export jobs, follow these steps:

1. In the header pane, click ▦ and then click **DM Connect for DI**.
   The DM Connect for DI page opens.



2. Click **Export Logical Names Job**.
   The Add Export BGM Job page appears.



Job Name: Enter Job Name

Start Date/Time: 06/17/2025 05:42 PM

End Date/Time: 06/18/2025

Notification Email: Email

CC List: Email

3. Set up job parameters as follows:

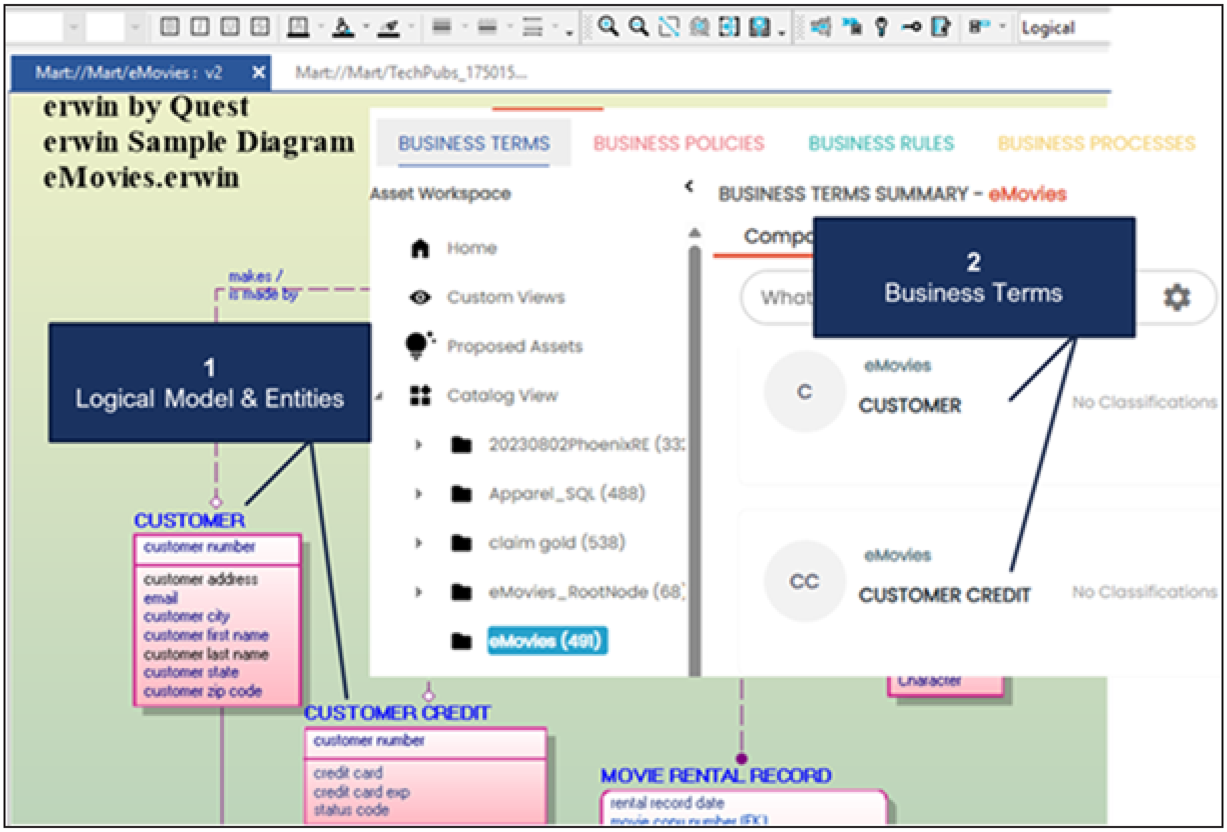
Tab	Field	Description
Catalogs	Catalog Tree	Select models from catalog to export. Before you select models, you can use the All Catalogs or Loaded Catalogs to display all available catalogs or only the expanded catalogs respectively. Apart from that, after you select catalogs, you can click to view only the selected catalogs in the Catalogs section.
	Include NSM	Select whether naming standards must be exported. A catalog named by NSM file is created under **Business Glossary Manager** > **DM NSM Files** custom

Tab	Field	Description
		asset. Ensure that the DM NSM Files asset is available in the Business Glossary Manager.
DI Information	Connectors	Select a configuration to use for the export job.

Tab	Field	Description
Job Information	Job Name	Specify a job name.
	Start Date/Time	Select the date and time at which the job must start.
	Job Interval	Select a suitable frequency at which the job must run. You can set the job to run once or recur daily, weekly, monthly, or yearly. You can also set up custom recurrence for jobs.
	Frequency	Select the hourly frequency at which the job should run. This property is available only when you set the Job Interval to Recurring.
	End Date/Time	If you set up recurring jobs, select the date and time at which the recurrence must end.
	Days	Select the days of the week on which the job should run. The days available here depend on the End Date/Time. This property is available only when you set the Job Interval to Recurring.
	Notify Me	Select the check box to receive a notification when the job status changes. This enables the Notification Email and CC List fields.
	Notification Email	Specify the email address at which you want to receive the notification.
	CC List	Specify a semi-colon-separated list of email addresses that must receive the job notification.
	Run Now	Select the check box to run the job immediately.

4. Click **Save**.
   The job is added to the calendar with its **Job State** set to Scheduled.

The job runs according to the schedule and exports logical names to Business Glossary Manager. For example, the logical names, Customer and Customer Credit, from the eMovies model are saved as business terms in the Business Glossary Manager.



Productivity Enhancements

Following productivity enhancements are available in erwin Mart Portal15.0:

- **erwin Mart Portal database configuration**:You can now connect to SQL Server and Azure SQL database via Microsoft Entra authentication. Thus, using identities managed within Microsoft Entra ID, instead of traditional SQL Server login and passwords.
- **erwin Mart Portal Advanced configuration**: Following options have been added to the Advanced settings of erwin Mart Portal configuration:
  - Update Mart Portal Path
  - Update ER360 Path

For more information, refer to the Configuring erwin Mart Portal topic.

The Is GitHub Enterprise option has been removed from the UI, its behavior is now determined by the system based on the domain type selected during source control repository configuration.

erwin ER360 Features and Enhancements

erwin ER360 includes the following enhancements in this release.

- Metadata Indexing is now automated via the new Index Metadata page.
- Global Search Enhancements
- Worksheet Enhancements include the following features:
    - MetaQL Support
    - User-defined Properties
    - Advanced Filters
- Diagram Enhancements

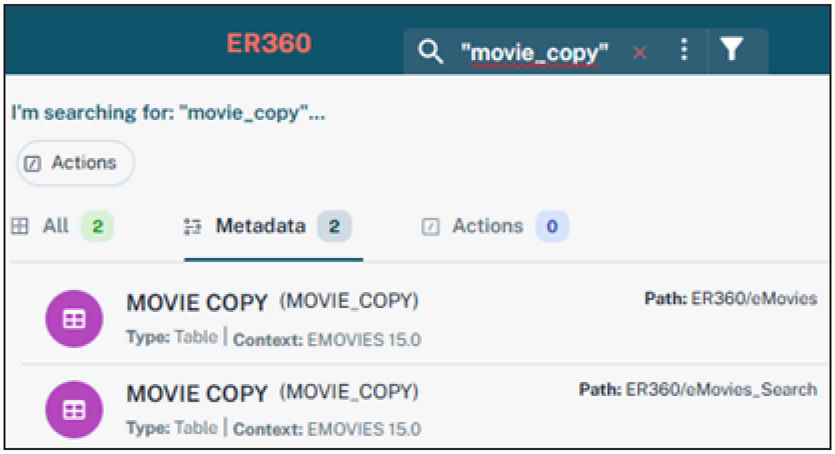
Global Search Enhancements

Global search now supports advanced query syntax that enables granular and targeted search.

Apart from the usual search string, you can use the following advanced search queries to search metadata with precision:

- **Exact words**: Use this format to search metadata objects that match the exact words in the search phrase without any special characters and words. The syntax for this search format is "*<word1>_<word2>*".

  For example, "movie_copy" returns all metadata that exactly match the search phrase.



"movie_copy"

I'm searching for: "movie_copy"...

MOVIE COPY (MOVIE_COPY) Path: ER360/eMovies

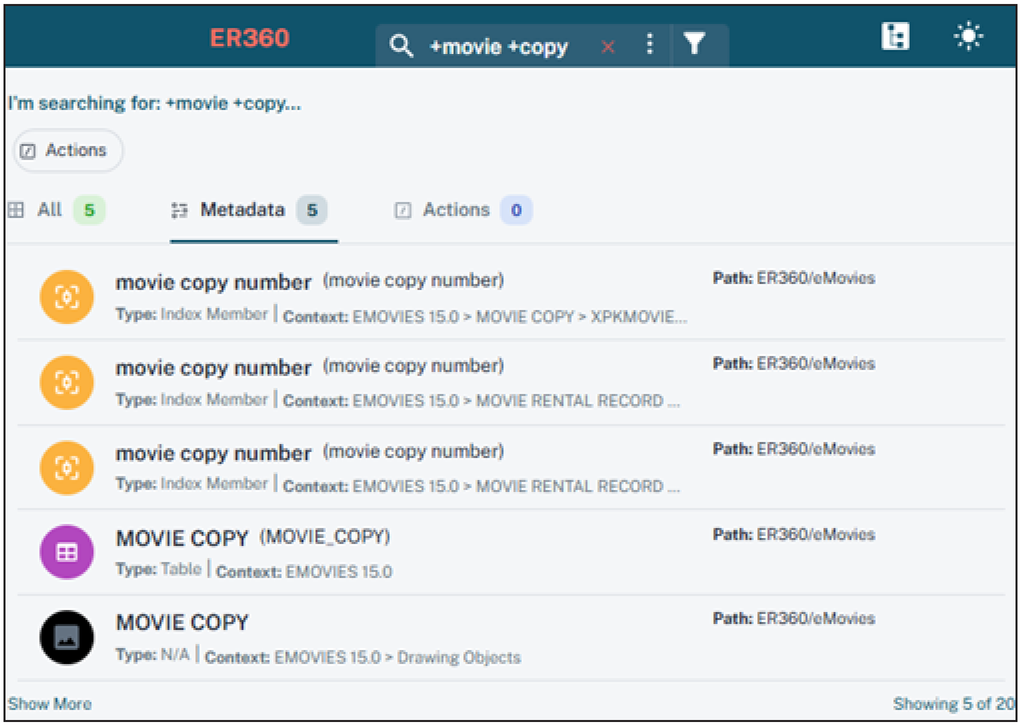
Type: Table | Context: EMOVIES 15.0

MOVIE COPY (MOVIE_COPY) Path: ER360/eMovies_Search

Type: Table | Context: EMOVIES 15.0

- **All words**: Use this format to search metadata objects that match all words preceded by the + (plus) sign in the search phrase. The syntax for this search format is +*<word1>* +*<word2>*.

  For example, the search phrase +movie +copy returns metadata that contains the words movie and copy in addition to any other words in the name of the metadata object.



+movie +copy

I'm searching for: +movie +copy...

movie copy number (movie copy number) Path: ER360/eMovies

Type: Index Member | Context: EMOVIES 15.0 > MOVIE COPY > XPKMOVIE...

movie copy number (movie copy number) Path: ER360/eMovies

Type: Index Member | Context: EMOVIES 15.0 > MOVIE RENTAL RECORD ...

movie copy number (movie copy number) Path: ER360/eMovies

Type: Index Member | Context: EMOVIES 15.0 > MOVIE RENTAL RECORD ...

MOVIE COPY (MOVIE_COPY) Path: ER360/eMovies

Type: Table | Context: EMOVIES 15.0

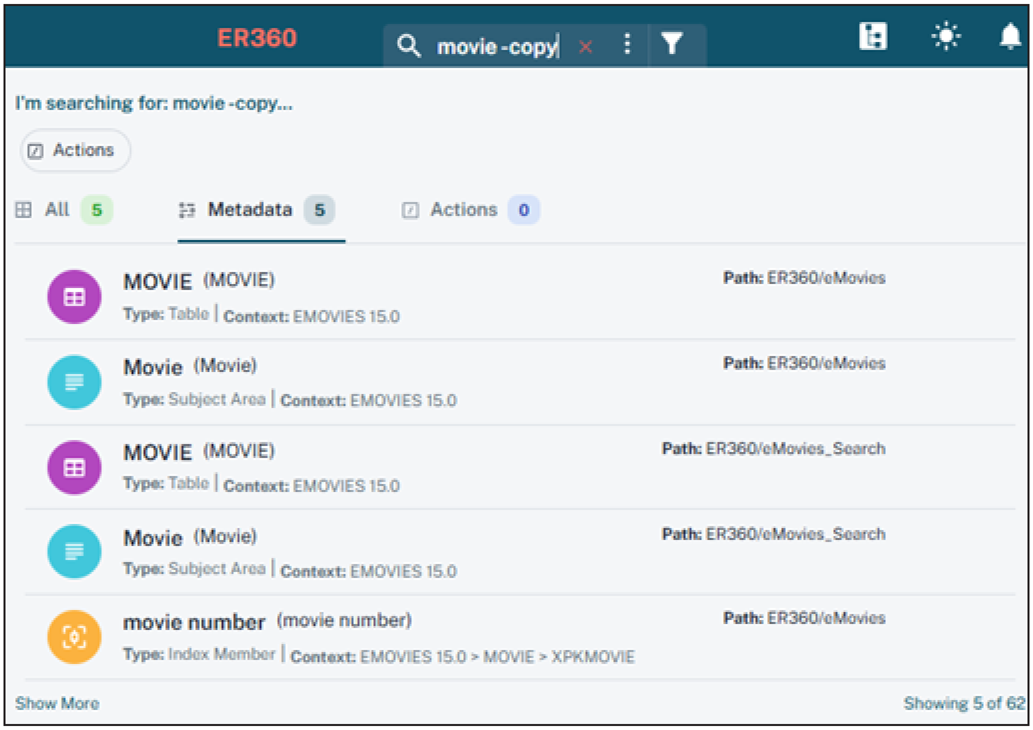
MOVIE COPY Path: ER360/eMovies

Type: N/A | Context: EMOVIES 15.0 > Drawing Objects

Showing 5 of 20

- **Exclude words**: Use this format to search metadata objects that do not contain all the words except the word preceded by the - (minus) sign. The syntax for this search format is *<word1>* -*<word2>*.

  For example, the search phrase movie -copy returns metadata that contains the words movie but not copy.



I'm searching for: movie -copy...

MOVIE (MOVIE) Path: ER360/eMovies
Type: Table | Context: EMOVIES 15.0

Movie (Movie) Path: ER360/eMovies
Type: Subject Area | Context: EMOVIES 15.0

MOVIE (MOVIE) Path: ER360/eMovies_Search
Type: Table | Context: EMOVIES 15.0

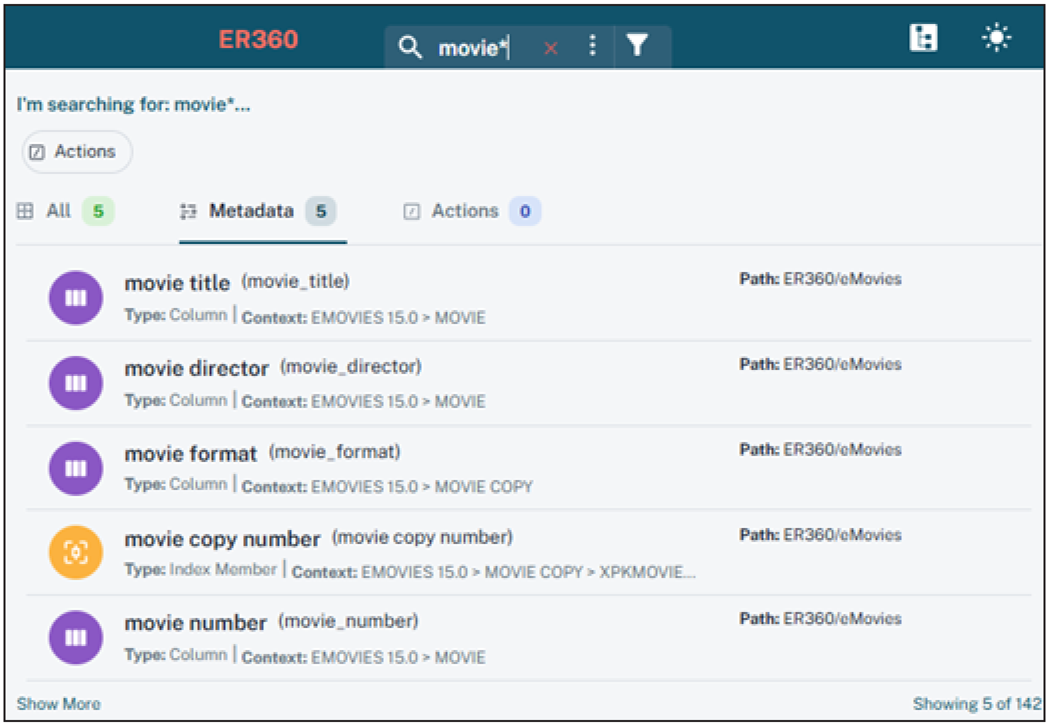
Movie (Movie) Path: ER360/eMovies_Search
Type: Subject Area | Context: EMOVIES 15.0

movie number (movie number) Path: ER360/eMovies
Type: Index Member | Context: EMOVIES 15.0 > MOVIE > XPKMOVIE

Showing 5 of 62

- **Wild cards**: Use this format to search metadata objects that contain the search string words preceded or succeeded by any other words or characters. The syntax for the this search format is *<word>** OR **<word>*.

  For example, the search phrases movie* returns all metadata that start with the word movie.



ER360 movie*

I'm searching for: movie*...

Actions

All 5 | Metadata 5 | Actions 0

movie title (movie_title)
Type: Column | Context: EMOVIES 15.0 > MOVIE
Path: ER360/eMovies

movie director (movie_director)
Type: Column | Context: EMOVIES 15.0 > MOVIE
Path: ER360/eMovies

movie format (movie_format)
Type: Column | Context: EMOVIES 15.0 > MOVIE COPY
Path: ER360/eMovies

movie copy number (movie copy number)
Type: Index Member | Context: EMOVIES 15.0 > MOVIE COPY > XPKMOVIE...
Path: ER360/eMovies

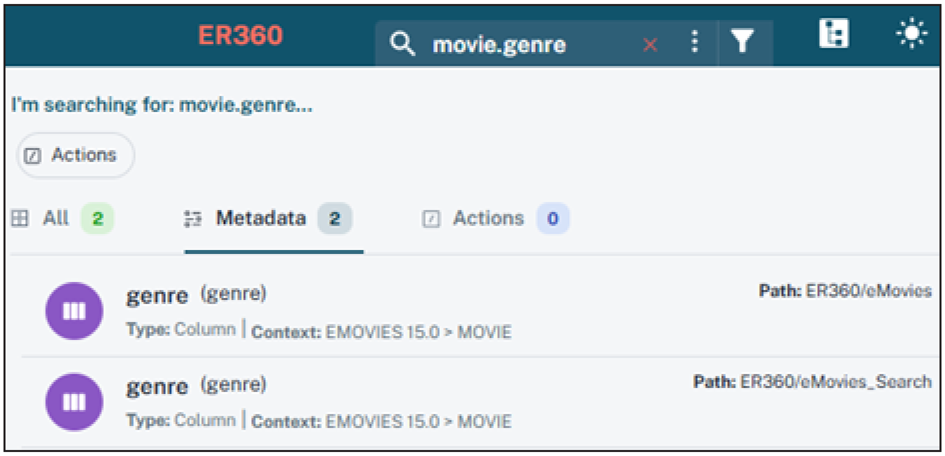
movie number (movie_number)
Type: Column | Context: EMOVIES 15.0 > MOVIE
Path: ER360/eMovies

Show More

Showing 5 of 142

- **Parent-child metadata**: Use this format to search parent-child metadata objects. The syntax for the this search format is *<parentname>*.*<childname>*.

  For example, the search phrase movie.genre returns all metadata where the parent entity is movie and the attribute is genre.



ER360 movie.genre

I'm searching for: movie.genre...

Actions

All 2 | Metadata 2 | Actions 0

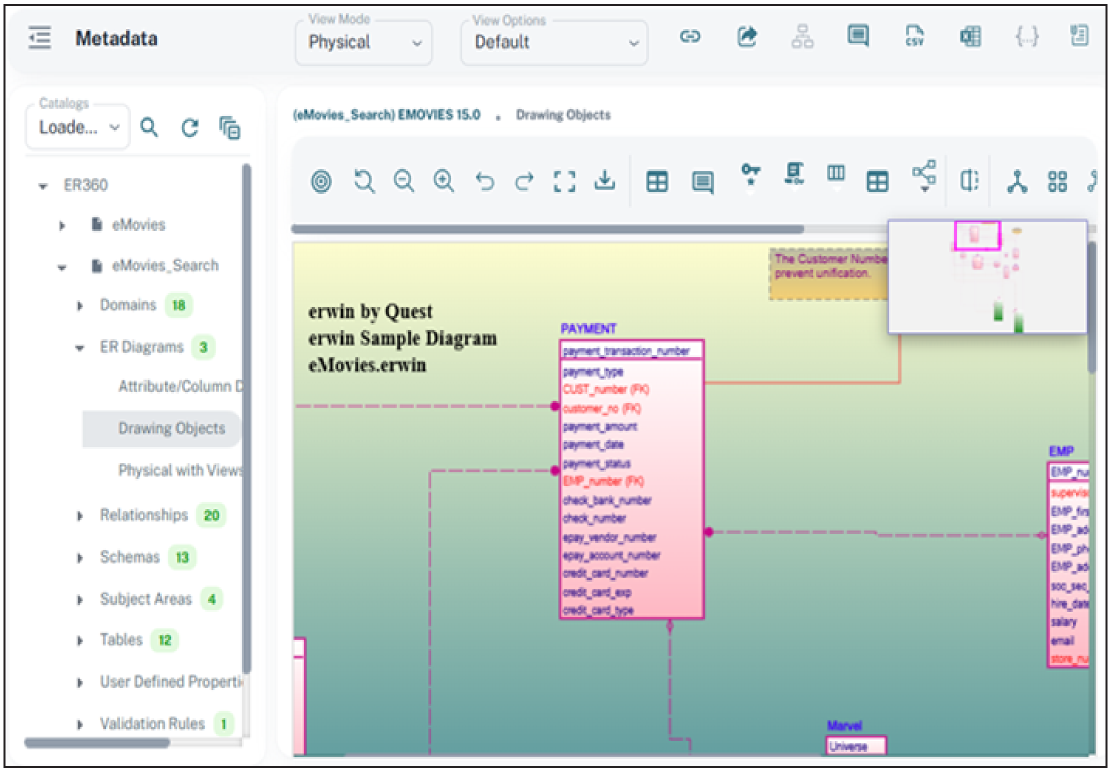
genre (genre)
Type: Column | Context: EMOVIES 15.0 > MOVIE
Path: ER360/eMovies

genre (genre)
Type: Column | Context: EMOVIES 15.0 > MOVIE
Path: ER360/eMovies_Search

Diagram Enhancements

Several enhancements have been made to the diagram view:

- The appearance and structure of models harvested from erwin Data Modeler (erwin DM) is remain visually consistent in ER360, ensuring seamless continuity between Logical and Physical Views.



View Mode: Physical

View Options: Default

Catalogs: Loade...

- Super Type-Sub Type relationship are now available, allowing users to clearly model generalization/specialization hierarchies in logical models.
- All object properties, including User-defined properties (UDPs), are now available under Object Properties.

Metadata Indexing

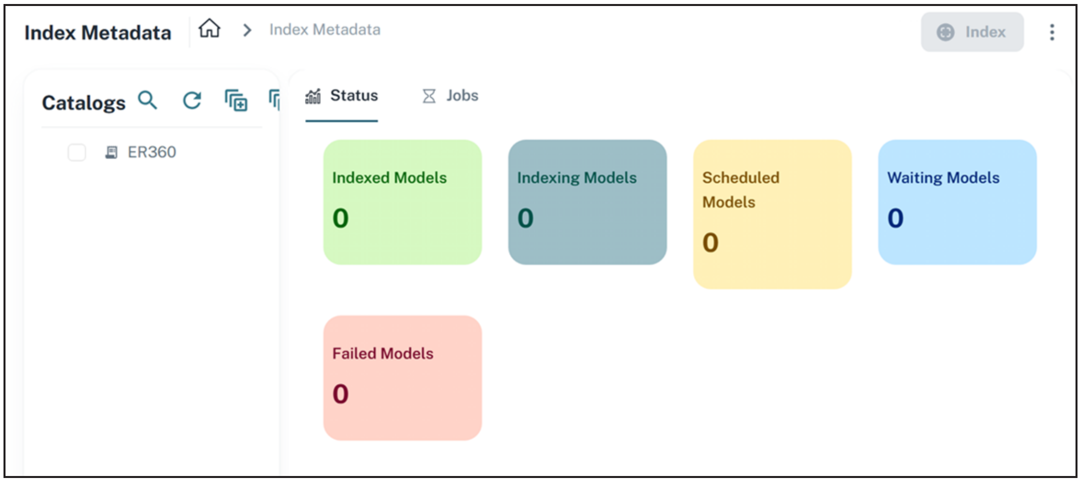
An index job is now automatically created when you harvest a model from erwin Mart Portal to erwin ER360. To support this, a new Index Metadata page has been added. You can also manually initiate an index job from this page.

View Index Jobs

To view index jobs, follow these steps:

1. Under **Application Menu**, click IM.

   The Index Metadata page opens.

   Indexed Models 0

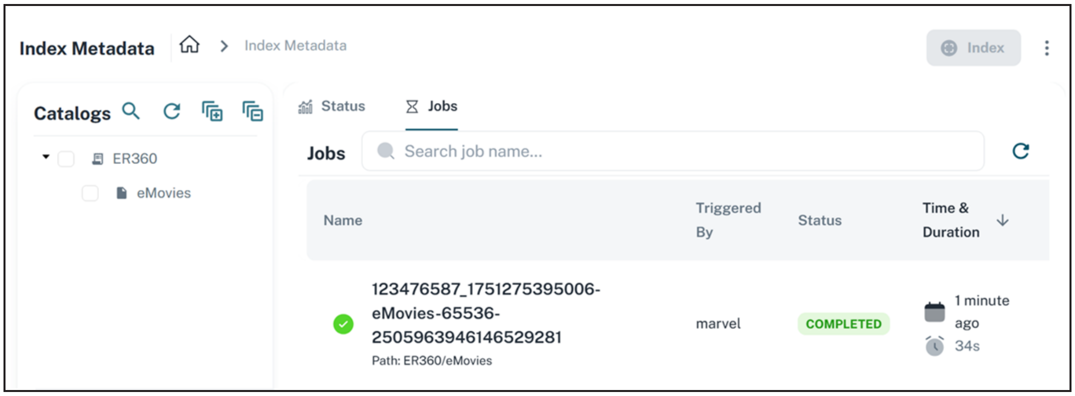
   Indexing Models 0

   Scheduled Models 0

   Waiting Models 0

   Failed Models 0

2. Click the **Jobs** tab. When you harvest a model, you can see the index job for the harvested model.



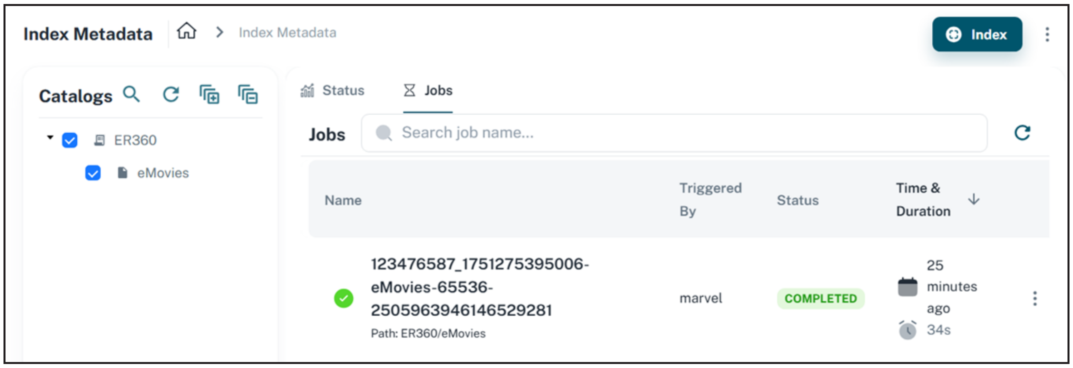
Name	Triggered By	Status	Time & Duration
123476587_1751275395006-eMovies-65536-2505963946146529281 Path: ER360/eMovies	marvel	COMPLETED	1 minute ago 34s

For information about harvesting, refer to the Harvesting Catalogs to erwin ER360 topic.

Index Metadata

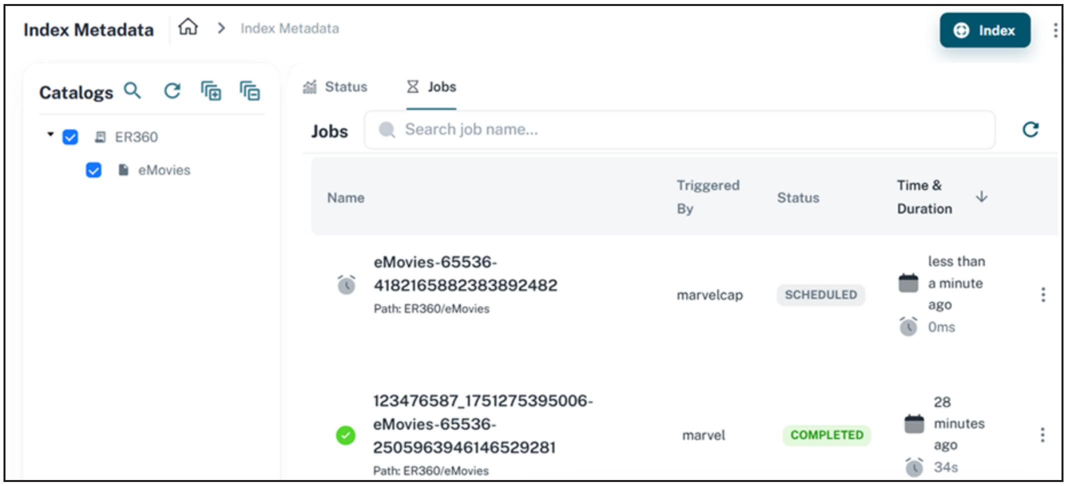
To index metadata, follow these steps:

1. On the Jobs tab, select a model under Catalogs.



Name	Triggered By	Status	Time & Duration
123476587_1751275395006-eMovies-65536-2505963946146529281 Path: ER360/eMovies	marvel	COMPLETED	25 minutes ago 34s

2. Click **Index**. An index job is scheduled.



Name	Triggered By	Status	Time & Duration
eMovies-65536-4182165882383892482 Path: ER360/eMovies	marvelcap	SCHEDULED	less than a minute ago 0ms
123476587_1751275395006-eMovies-65536-2505963946146529281 Path: ER360/eMovies	marvel	COMPLETED	28 minutes ago 34s

Once the indexing job is completed, you can search for the updated metadata or objects.

Worksheet Enhancements

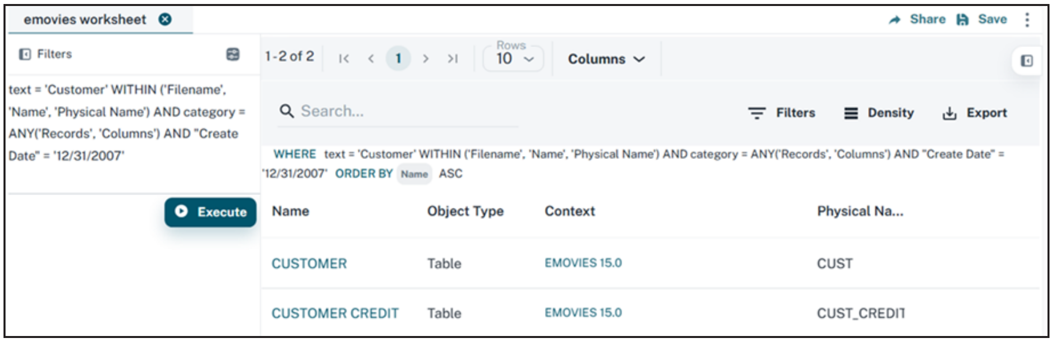
erwin ER360 introduces the following Worksheet features:

- **MetaQL Support**: You can now use MetaQL, a lightweight, SQL-like query language, to make your filter logic in data models easier to understand, share, and audit.

  To view and edit a MetaQL query, follow these steps:

  1. In the Filters pane, click .

     The MetaQL query for the applied filters is displayed.

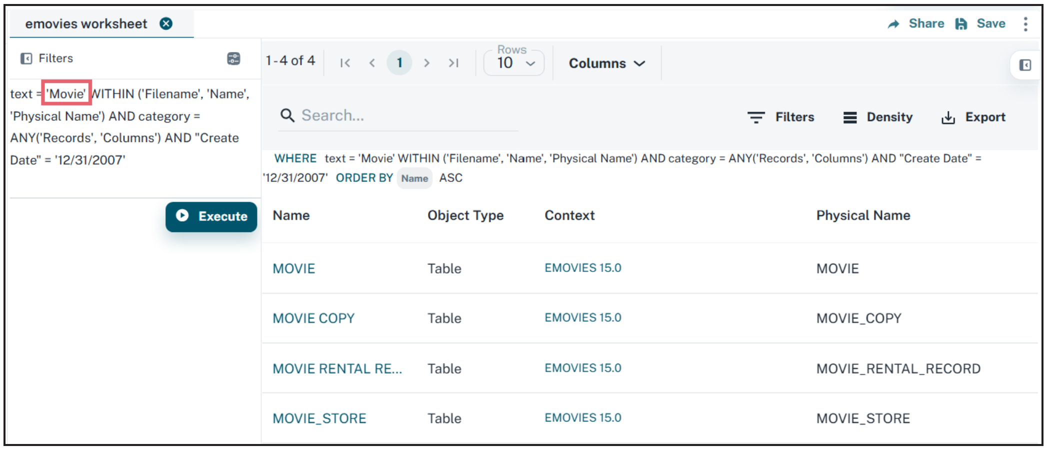
     text = 'Customer' WITHIN ('Filename', 'Name', 'Physical Name') AND category = ANY('Records', 'Columns') AND "Create Date" = '12/31/2007'

     WHERE text = 'Customer' WITHIN ('Filename', 'Name', 'Physical Name') AND category = ANY('Records', 'Columns') AND "Create Date" = '12/31/2007' ORDER BY Name ASC

Name	Object Type	Context	Physical Na...
CUSTOMER	Table	EMOVIES 15.0	CUST
CUSTOMER CREDIT	Table	EMOVIES 15.0	CUST_CREDIT

  2. Update the filters in the query as required, and then Click **Execute**.

     For example, change the text value from 'Customer' to 'Movie'.

     text = 'Movie' WITHIN ('Filename', 'Name', 'Physical Name') AND category = ANY('Records', 'Columns') AND "Create Date" = '12/31/2007'

     WHERE text = 'Movie' WITHIN ('Filename', 'Name', 'Physical Name') AND category = ANY('Records', 'Columns') AND "Create Date" = '12/31/2007' ORDER BY Name ASC

Name	Object Type	Context	Physical Name
MOVIE	Table	EMOVIES 15.0	MOVIE
MOVIE COPY	Table	EMOVIES 15.0	MOVIE_COPY
MOVIE RENTAL RE...	Table	EMOVIES 15.0	MOVIE_RENTAL_RECORD
MOVIE_STORE	Table	EMOVIES 15.0	MOVIE_STORE

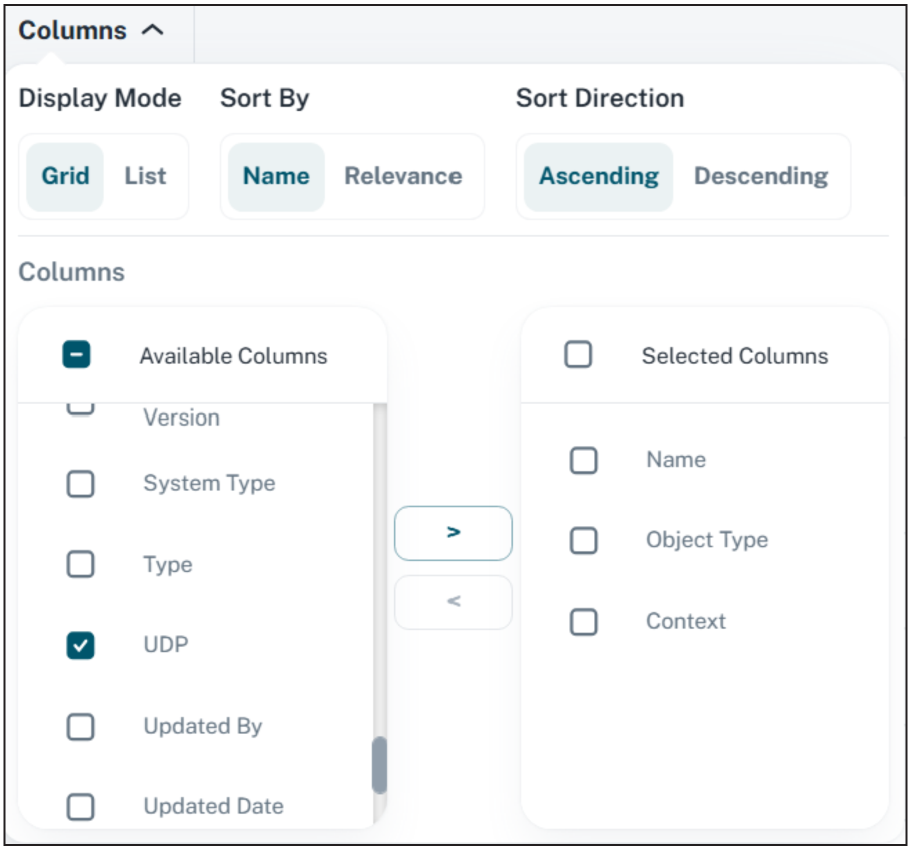
     You can view the filtered data.

- **Advanced Filters**: Additional filters are improved to enable you to refine searches for metadata objects with enhanced control. You can filter records using an extensive range of common attributes, object attributes, and user-defined properties (UDPs) to generate more accurate and relevant results.

  The following table lists a few examples:

Common Attributes	Object Attributes	UDP
Certification Count	Comment	Create Date
Certified By	Data Type	Color
Comment Count	Definition	Data Steward
Commented By	Name	Attribute Owner
Created By	Default Value	Date Created
Created Date	Nullable	
Endorsed By		
Endorsement Count		
Parent Name		
Updated By		
Updated Date		
Warned By		
Warning Count		

- **User-defined Properties**: You can now view and add user-defined properties to filter results using the Column option. These user-defined properties are also visible in the Properties pane.



Columns

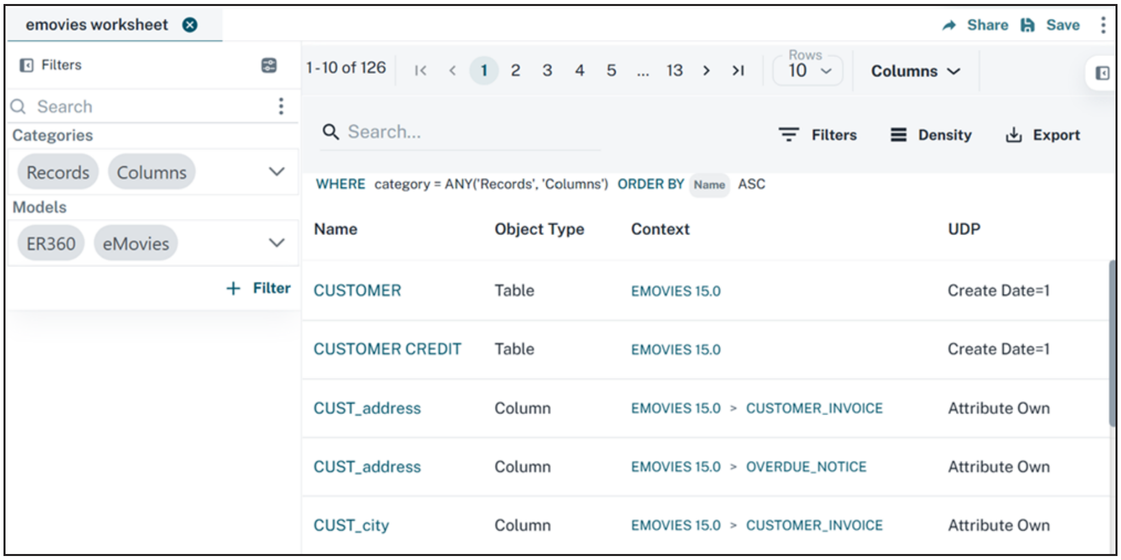
Display Mode: [✓ Grid] List

Sort By: [✓ Name] Relevance

Sort Direction: [✓ Ascending] Descending

Columns

Available Columns	Selected Columns
□ Version	□ Name
□ System Type	□ Object Type
□ Type	□ Context
☑ UDP	
□ Updated By	
□ Updated Date	



Categories: Records, Columns

Models: ER360, eMovies

WHERE category = ANY('Records', 'Columns') ORDER BY Name ASC

Name	Object Type	Context	UDP
CUSTOMER	Table	EMOVIES 15.0	Create Date=1
CUSTOMER CREDIT	Table	EMOVIES 15.0	Create Date=1
CUST_address	Column	EMOVIES 15.0 > CUSTOMER_INVOICE	Attribute Own
CUST_address	Column	EMOVIES 15.0 > OVERDUE_NOTICE	Attribute Own
CUST_city	Column	EMOVIES 15.0 > CUSTOMER_INVOICE	Attribute Own