Database Actions

Using Guide for On-Premises Oracle Database

Release 24.1
F94252-02
June 2024

Database Actions Using Guide for On-Premises Oracle Database, Release 24.1

F94252-02

Primary Author: Celin Cherian

Contents

4 The REST Pages

13 The Monitoring Pages

A Supported SQL*Plus and SQLcl Commands in SQL Worksheet

Preface

This document provides information about Oracle Database Actions, a web-based interface that provides development and administration features for Oracle Database.

Audience

This online help is intended for those using Oracle Database Actions included with Oracle REST Data Services.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Product Accessibility

Oracle Database Actions provides features to support accessibility. See *Oracle Database Actions Accessibility Guide*.

Related Documents

Oracle REST Data Services Installation and Configuration Guide

Oracle REST Data Services Developer's Guide

Oracle Autonomous Database

Oracle SQL Developer User's Guide

To download release notes, installation documentation, white papers, or other collateral for SQL Developer, go to the Oracle Technology Network (OTN) at

`http://www.oracle.com/technetwork/`

For the PL/SQL page on OTN, see `http://www.oracle.com/technetwork/database/features/plsql/`

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Third-Party License Information

See Third-Party License Information in *Oracle REST Data Services Developer's Guide*

1

Changes in Release 24.1 for *Using Oracle Database Actions*

The new features in this release are:

- New home page for Database Actions

  See The Home Page - Launchpad

2

About Oracle Database Actions

Oracle Database Actions is a web-based interface for on-premises Oracle Database that uses Oracle REST Data Services to provide many of the database development and administration features of desktop-based Oracle SQL Developer. The main features include running SQL statements and scripts in the worksheet, exporting data, creating RESTful web services, managing JSON collections, monitoring databases, and creating Data Modeler diagrams.

> **Note:**
>
> Some features of Database Actions are only available if you sign in as a user with the DBA role. For such features, a "restricted availability" statement appears at the start of the feature description. For example:
>
> Available only if you signed in as a database user with the DBA and PDB_DBA roles.

Database Actions is also available with:

- Oracle Autonomous Database cloud services. See Database Actions
- Oracle APEX. See Accessing SQL Developer Web Directly from APEX

> **See Also:**
>
> - Creating or Editing a User for creating users and assigning roles.
> - ORDS System Requirements for system requirements.

2.1 About the Database Actions User Interface

This section describes the Database Actions user interface.

The Database Actions user interface has three components:

- The Header at the top
- The page body, whose content varies depending on which page you are viewing
- The Status Bar at the bottom

Header

The header contains the Selector icon, a Search field, the help icon, and the user drop-down list.

- **Selector Icon**

  Click the Selector icon to see the main navigation menu slide into view. Click **Oracle Database Actions** in the header to go to the Launchpad page.

- **Search field**

  To enter a search term, click in the Search field or use the shortcut key **Ctrl+K** (**Command+K** for Apple computers). For more information, see Using the Omnisearch Bar.

- **Help Icon**

  Click the help icon to open the contextual or online help for the page you are viewing.

- **User Drop-Down List**

  The user drop-down list shows the database user you are signed in as, and provides the following items when you open it:

  – **Preferences**

    The options are:

    Region

    * **Language**: Select one of the following languages for the user interface: English, German, Spanish, French, Italian, Japanese, Korean, Portuguese, and Chinese
    * **Timezone**: Select **UTC** or **Local time zone** from the drop-down list.

    Code Editor

    Provides global settings that apply to the code editor, such as theme, indentation, font family and line numbers.

    Worksheet

    * **SQL History**: Controls whether the history of commands executed in the code editor is enabled in the browser or not.

  – **Log**: Opens a dialog that shows the list of HTTP calls made during your session.

  – **About**: Opens a dialog providing version information for the database and other components as well as copyright and licensing information.

  – **Sign Out**: Signs you out of your database session.

Status Bar

The status bar contains icons that link to log files. The three icons (Errors, Warnings, Processes) are filters that have been applied to the log file.

Errors, **Warnings**: Displays an Errors or Warnings dialog, which lists log entries from unsuccessful REST calls or from any other problem in the application.

Processes: Displays a Processes dialog, which logs REST calls that are either finished or ongoing.

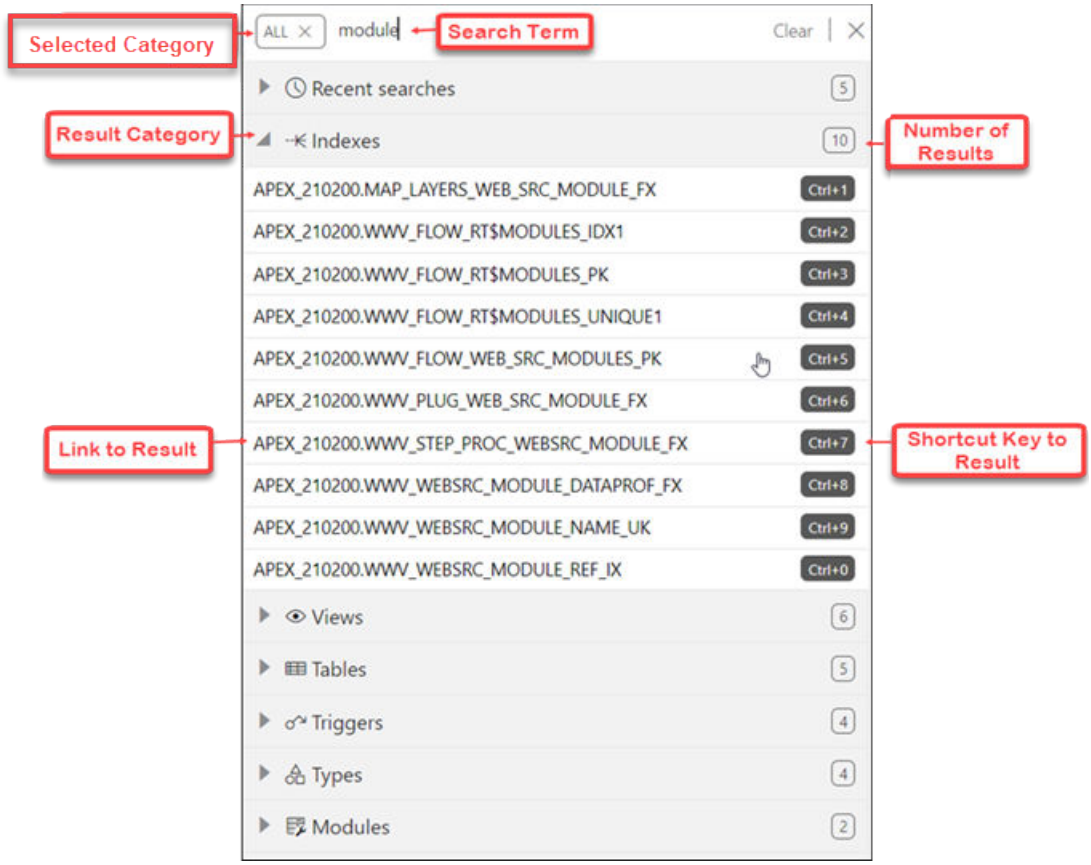
Log notification link: Displays a Log dialog, containing log entries of the following types: Errors, Warnings, Processes, SQL History and SQL Result.

2.1.1 Using the Omnisearch Bar

You can access the Omnisearch bar (Search field) in the header from any page in Database Actions.

To enter a search term, click in the search field located at the top right of the header, or use the shortcut key **Ctrl+K** (**Command+K** for Mac OS systems).

In the Omnisearch bar, you can filter the search entry by selecting a category (such as tables, views, templates, and so on) from the displayed list.



If you do select a category, you will then need to enter the exact search term.

If you do not select a category, **ALL** is selected by default. After you enter the search term, a search is performed across all categories. In some cases, the search term is searched across multiple attributes of a category. The following table lists the attributes that are searched for each category:

Category	Attribute
Tables	Name
Views	Name
Indexes	Name
Packages	Name

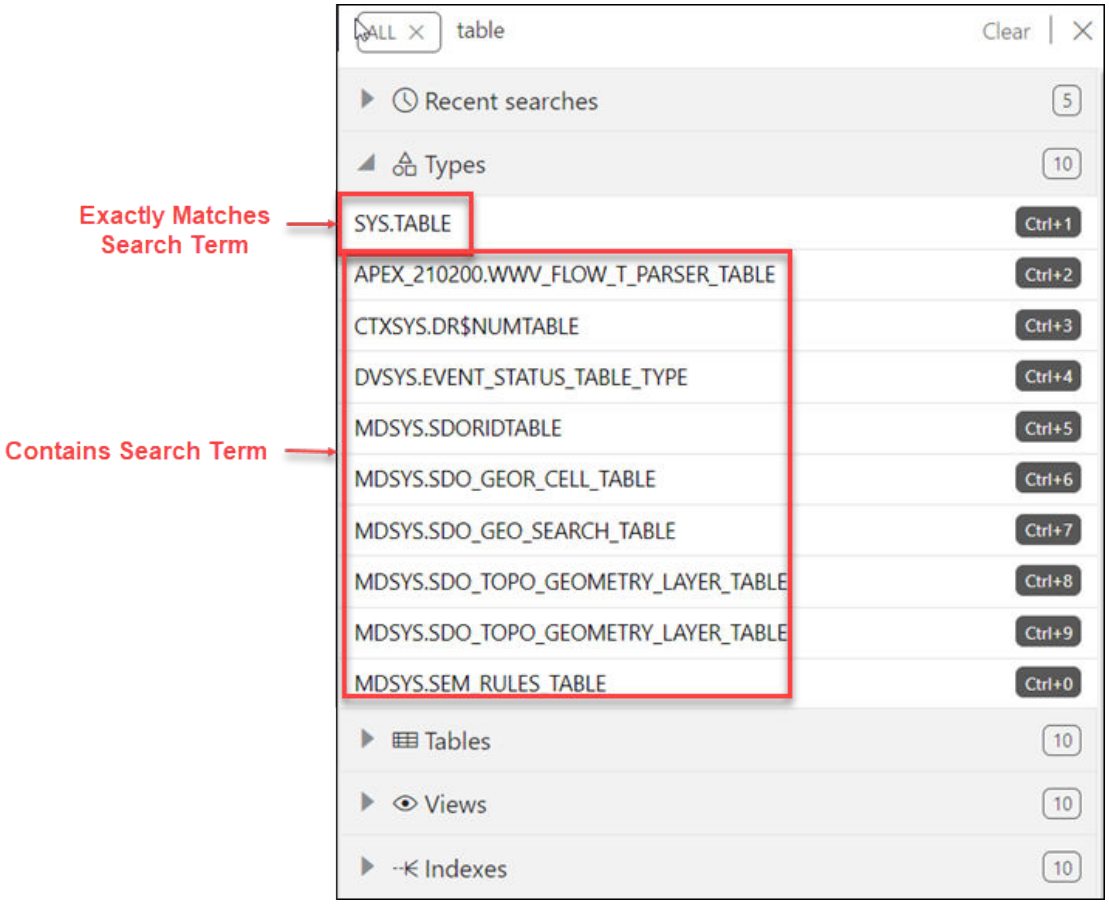
Category	Attribute
Functions	Name
Procedures	Name
Triggers	Name
Types	Name
Sequences	Name
Charts	Name, Comments, URI_Prefix
Dashboards	Name, Comments, URI_Prefix
Modules	Name, Comments, URI_Prefix
Templates	Comments, URI_Prefix
Handlers	Comments
Roles	Name
Privileges	Label, Name, Comments, Description
OAuth Clients	Name, Description
Database Users	Username, Alias (alias is used depending on user permissions)
APEX Workspaces	Workspace, Workspace display name

Displaying Results for the Search Term

In the results displayed, categories are sorted based on the following criteria:

1. Number of exact matches
2. Most number of matched items

The results within a category are displayed as two groups. The first group consists of items that exactly match the search term. The second group consists of items that contain the search term but do not start with it. Within each group, the items are sorted alphabetically.



You can quickly access previous search terms using the **Recent Searches** list.

2.2 Accessing Database Actions

Oracle Database Actions is included with Oracle REST Data Services.

To access Database Actions:

1. Enable **Database Actions** in Oracle REST Data Services. For more information, see Interactive Command-Line Interface Installation in *Oracle REST Data Services Installation and Configuration Guide*.
2. To use Database Actions, you must sign in as a database user whose schema has been REST-enabled.

   Execute the following code as a database user with the DBA role:

```
BEGIN
 ords_admin.enable_schema(
  p_enabled => TRUE,
  p_schema => 'schema-name',
  p_url_mapping_type => 'BASE_PATH',
  p_url_mapping_pattern => 'schema-alias',
  p_auto_rest_auth => TRUE
 );
```

```
 commit;
END;
```

where:

- *schema-name* is the database schema name in all-uppercase.
- *schema-alias* is an alias for the schema name that will appear in the URL the user will use to access Database Actions. Oracle recommends that you do not use the schema name itself as a security measure to keep the schema name from being exposed.
- *p_auto_rest_auth* specifies that the REST /metadata-catalog/ endpoint requires authorization. REST uses the metadata-catalog to get a list of published services on the schema.

3. Sign in to Database Actions. See Signing In to Database Actions

2.3 Signing In to Database Actions

You can sign in to Database Actions by connecting to a single default database or to multiple databases or by using a schema alias.

> **Note:**
>
> - From Oracle REST Data Services, you can access the Database Actions Sign In page from the ORDS landing page (`https://example.com/ords`).
> - From APEX, you can open Database Actions directly from the SQL workshop dropdown menu. See Accessing SQL Developer Web Directly from APEX

The Sign in page has a languages menu on the top right where you can select the language to display.

Default Database

To connect to a single default database:

- In the Database Actions Sign-in page, enter your user name and password and click **Sign in**.

> **Note:**
>
> In the Password field, if the CAPS LOCK key is turned on, an up arrow icon is displayed in the right corner of the field.

ORDS Schema Alias

To connect to a database using an ORDS schema alias, which is different from your user name:

- In the Database Actions Sign-in page, expand **Advanced** and in the Path field, enter the schema alias.

  See Enabling REST to know how to set up a schema alias.

- Enter your user name and password and click **Sign in**.

Multiple Databases

To connect to a database in a database pool:

- In the Database Actions Sign-in page, expand **Advanced** and in the Path field, enter the database pool name specified while installing Oracle REST Data Services.

  See Configuring Oracle REST Data Services for Multiple Databases in *Oracle REST Data Services Installation and Configuration Guide* for more information about database pools.

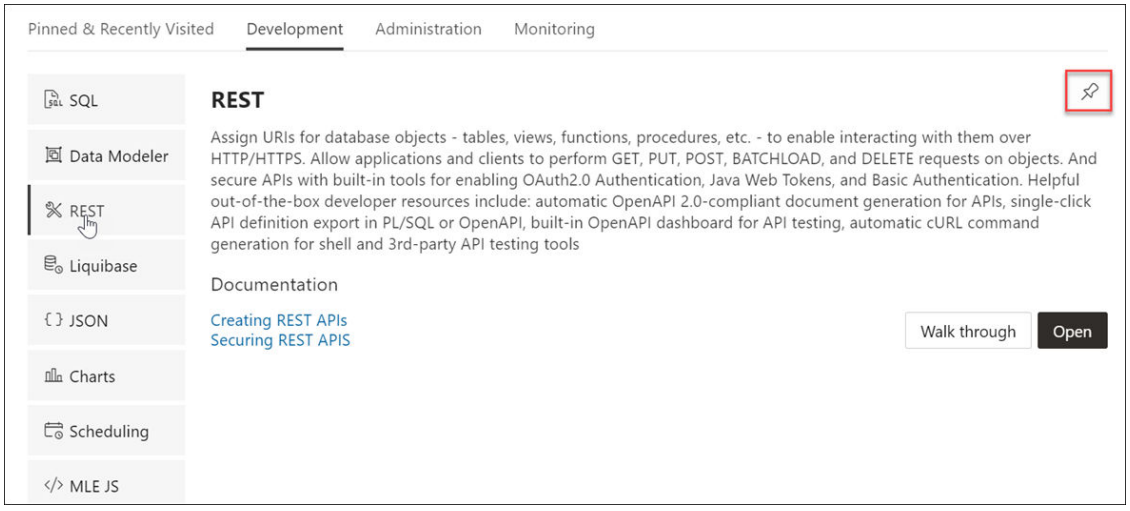
- Enter your user name and password and click **Sign in**.

2.4 The Home Page - Launchpad

The Launchpad page for Database Actions can now dynamically display categories and features based on whether you are logged in to an on-premises, Autonomous Database or OCI environment. For on-premises, the main categories displayed are Pinned and Recently Visited, Development, Administration and Monitoring. For Autonomous Database, additional categories displayed are Data Studio and Downloads. Descriptions for features belonging to these categories are provided below.

The first time you log in, the Development tab is displayed by default. All features that pertain to the Development category are displayed vertically on the left side of the screen. When you hover over a feature name, its description is displayed on the right along with links to specific documentation pages. A new feature is highlighted with a blue label.

In Pinned & Recently Visited, Pinned lists all features that you have pinned (saved). To pin a feature, click the Pin icon displayed on the top right of the feature description. There is no limit on the number of features that can be pinned. **Recently Visited** lists the last seven features that you have accessed.



To navigate to a specific feature such as the REST pages, click **REST** on the left or click **Open** in the feature description part.

The following is a description of each feature displayed on the home page.

Development

- **SQL**: Enter and execute SQL and PL/SQL commands, and create database objects. See The SQL Page
- **Data Modeler**: Create diagrams from existing database schemas, generate DDL statements, and create reports. See The Data Modeler Page
- **APEX**: Link to the Oracle Application Express sign-in page. Application Express is a rapid web application development platform for the Oracle database. See Oracle Application Express documentation
- **REST**: Develop RESTful web services and ensure secure access. See The REST Pages
- **Liquibase**: View changelogs for your schema. See The Liquibase Page
- **Charts**: Create charts and dashboards containing multiple charts using SQL queries. See The Charts and Dashboards Page
- **JSON**: Manage and query JSON collections. JSON is available only if you are signed in as a database user with the SODA_APP role. See The JSON Page
- **MLE JS**: Execute JavaScript code in Oracle Database. See The MLE JS Page
- **Scheduling**: Provide details of scheduled jobs, chains, programs and schedules. See The Scheduling Pages

Administration

Administration is available only if you are signed in as a database user with administrator rights.

- **Database Users**: Perform user management tasks such as create, edit, and REST enable users. Users have access to features based on user privileges. See The Database Users Page
- **Data Pump**: Monitor Data Pump jobs initiated through the available Database API endpoints, the DBMS_DATAPUMP package, or the SQL Developer Data Pump Export and Import wizards. See The Data Pump Page
- **APEX Workspaces**: Create and manage APEX workspaces.

Monitoring

Monitoring is available only if you are signed in as a database user with administrator rights.

- Monitor database activity and performance using various tools.

  Monitoring is available to database users with DBA and PDB_DBA roles. The only exception is the Real Time SQL Monitoring feature, which is also available to non-administrator users for Oracle Database 19c and later versions.

  See The Monitoring Pages

Part I
Development

This part provides information about the following topics:

Topics:

3

The SQL and Data Modeler Pages

Use the SQL page to enter and execute SQL and PL/SQL statements and create database objects. The Data Modeler page enables you to create diagrams from existing schemas, retrieve data dictionary information, generate DDL statements, and export diagrams.

Topics

3.1 The Overview Page

The Overview page contains widgets that provide a general overview of the activity in the SQL and Data Modeler pages.

To navigate to the Overview page, click **Selector** and then under Development, select **Overview**.

- **My Worksheets**: Displays your saved worksheets. You can click the name of the worksheet to open it in the Worksheet page.
- **My Diagrams**: Displays the Data Modeler diagrams that have been saved. You can click the name of the diagram to open it in the Data Modeler page.
- **Recently Modified Objects**: Displays a timeline of the created, modified, and dropped objects in the database. You can zoom in and out using the + and – icons. You can also move horizontally by dragging the cursor to the right or left.
- **Invalid Objects**: Displays the invalid objects in your schema.
- **Table Stats Freshness**: Displays the time period since the tables were last analyzed.

You can right-click the header in Invalid Objects, Table Stats Freshness, My Worksheets, or My Diagrams to manage or sort columns:

- **Columns**: Enables you to select columns to show or hide.
- **Sort**: Displays a dialog box for selecting columns to sort by. For each column, you can specify ascending or descending order, and you can specify that null values be displayed first.

Right-click the body of the display table to count rows or to view records:

- **Count Rows**: Displays the number of rows in the table.
- **Single Record View**: Enables you to view data for a table or view, one record at a time.
- **Copy**: Copies data from a cell or a row or a range of rows. To copy from more than one row, select the rows you want to copy, right-click by pressing the SHIFT or CTRL key, and select **Copy**.

3.2 The SQL Page

The SQL page enables you to enter and execute SQL and PL/SQL statements, and create database objects.

To navigate to the SQL page, do either of the following:

- In the Launchpad page, select the **Development** tab and click **SQL**.
- Click **Selector** to display the navigation menu. Under Development, select **SQL**.

You can use SQL and PL/SQL statements in the worksheet to create a table, insert data, create and edit a trigger, select data from a table, and save that data to a file. Some other features are syntax highlighting and error detection.

The SQL page consists of the left pane for navigating worksheets and objects, the editor for executing SQL statements, and the output pane for viewing the results. These panes are described in the following sections:

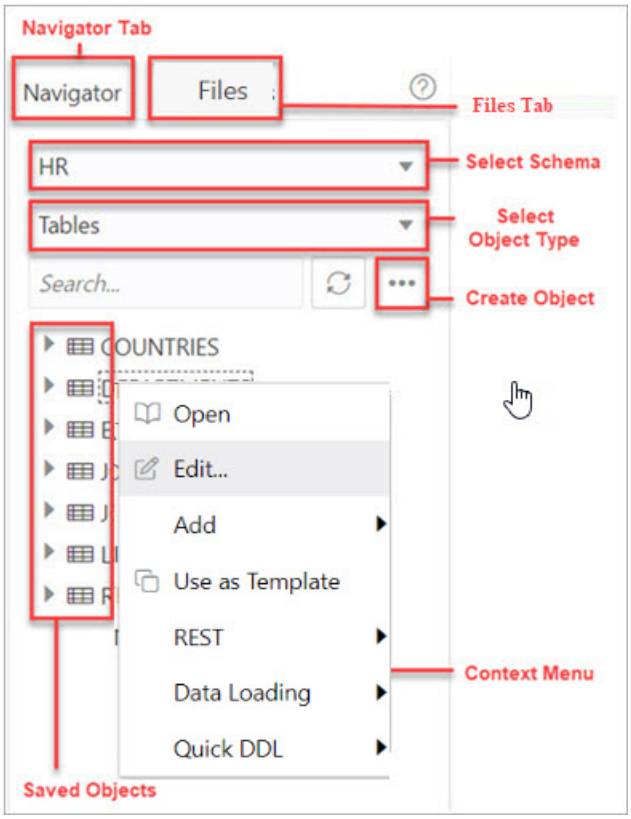
- Object Navigator and Files
- Executing SQL Statements in the Code Editor
- Viewing the SQL Output
- Loading Data

3.2.1 Object Navigator and Files

The Navigator tab in the left pane displays saved objects for the selected schema. The Files tab enables you to view and open files saved in your browser or local system.

The following figure shows the various elements in the left pane.

Figure 3-1 Left Pane in SQL



Navigator Tab

Displays saved objects for the selected schema.

- **Schema and Object Type selector**: Use the drop-down lists to select the schema and filter the results by object type.
- **Search**: Searches the contents of a saved worksheet or search for objects in the Navigator tab by name. The search functionality is not case-sensitive, retrieves all matching entries, and does not require the use of wild card characters.
- **Context menu**: Options in the context menu are:
  - **Open** to browse properties and data relevant to the object type (tables and views).

    The Data pane displays the data for a table, view, or materialized view.

    To edit an entry, double-click a cell to make edits. You can also click and enter the value. When you make an edit, the border of the gutter cell in that row changes to blue.

    The icons available in the Data pane are:

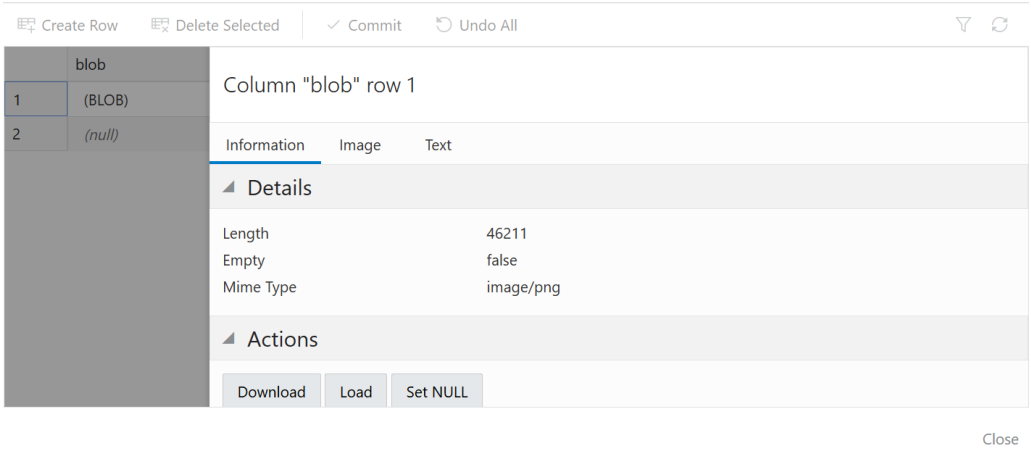
    * **Create Row**: Insert a new row into the database table. When you insert a row using the **Create Row** icon, the row is committed into the database.
    * **Delete Selected**: Mark the selected row for deletion. When you mark a row for deletion, the border of the row changes to red.
    * **Commit**: Commit all changes made to the database.

* **Undo All**: Revert all changes that are marked for commit.

Use the **Filter** icon at the top right corner to filter the column data. You can also right-click a cell to access the context menu to count rows, view a single record, export or copy the cell text to clipboard.

To view a Binary Large Object data type (BLOB), click the pencil icon for a BLOB data type. In the View Value dialog:

* The **Image** tab displays the loaded image, if the loaded BLOB type is an image.
* The **Text** tab displays the text file, if the loaded BLOB type is text .
* The **Information** tab displays the details and allows you to perform the following actions:
  * **Download**: To download the image/text file of BLOB data type .
  * **Load**: To insert an image/text of BLOB data type.
  * **Set NULL**: To delete the object and set the value as NULL.



– **Edit** edits the properties of an existing object.
– **Add** creates an object based on the object type selected.
– **Use as Template** creates an object by using the properties of an existing object as the template.
– **REST**
  * **Enable** enables REST access for the database object. See Enabling REST Access for a Database Object

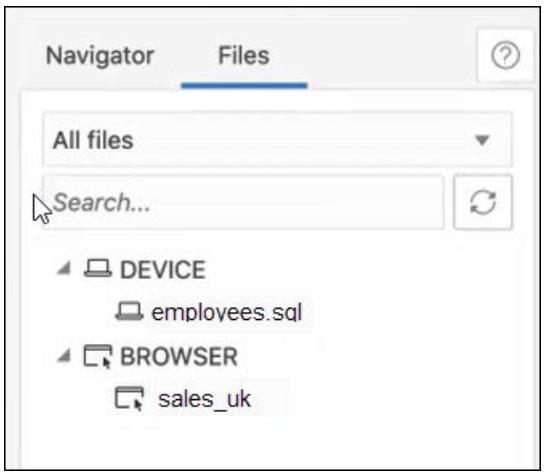
    A REST enabled object is indicated by a **REST Enabled** icon in the Navigator tab.
  * **Disable** disables REST access for the database object after it is enabled. See Disabling REST Access for a Database Object
  * **cURL Command** generates a cURL call for a selected HTTP method for the database object. See Generating cURL Requests for a REST-Enabled Database Object
– **Data Loading** loads data from local files into a table.
– **Quick DDL** generates Data Definition Language statements for the object.

- **Refresh** : Refreshes the objects or worksheets listed in the left pane.
- **Object Submenu** : Opens the Create Object dialog to create a new object based on the type selected in the drop-down list.
- **Help** : Provides contextual help documentation.

Files Tab

Enables you to open files from your browser or local device.

> **Note:**
>
> The DEVICE category in the left pane is displayed only when using a Chromium-based browser in a secure context (HTTPS).



All files: Use the drop-down list to filter files by browser or device.

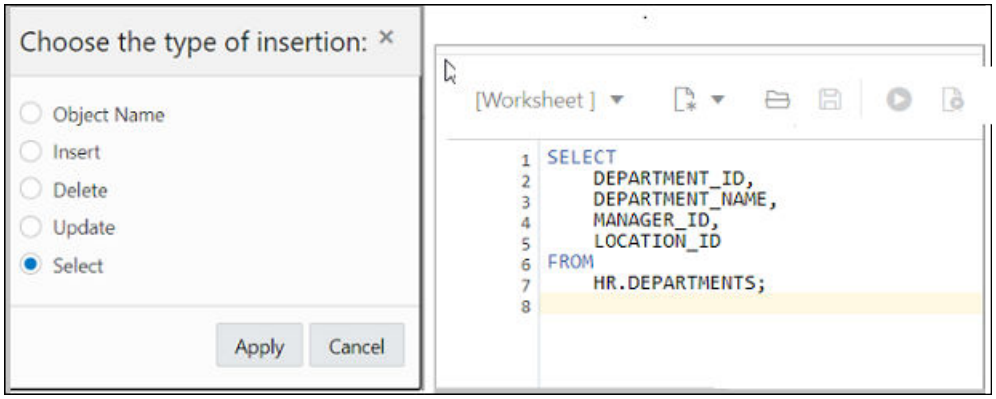
The context menu options for a file are **Open** and **Delete**. In Device, the corresponding option for deleting a file is **Forget**. In this case, the file is not deleted, instead the reference to the file is removed.

Drag and Drop Objects and Files into the Worksheet

You can drag objects from the left pane and drop them into the worksheet editor in the right pane.

- If you drag and drop a table or view, you are prompted to select one of the following SQL statements: Insert, Update, Select, or Delete. For example, if you choose Select, a Select statement is constructed with all columns in the table or view. You can then edit the statement, for example, modifying the column list or adding a WHERE clause.

Figure 3-2 Insert Select Query



If you choose Object Name, the name of the object prefixed by the schema name is added to the worksheet.

- If you drag and drop a function or procedure, you can choose to insert the name or the PL/SQL code of the function or procedure in the worksheet. If you select the PL/SQL code, you can enter the parameters before inserting the code into the worksheet.

3.2.2 Executing SQL Statements in the Code Editor

The code editor in the SQL page enables you to run SQL statements, PL/SQL scripts, and JavaScript code. The main features include in-context code completion, syntax highlighting, and error debugging.

You can enter SQL and PL/SQL statements to specify actions such as creating a table, inserting data, selecting data, or deleting data from a table. SQL keywords are automatically highlighted. For multiple statements, you must terminate:

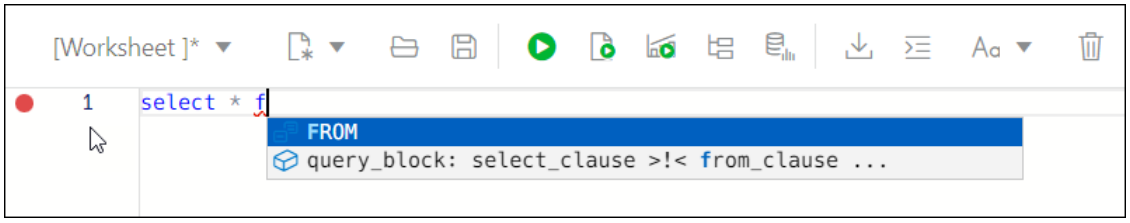
- Each non-PL/SQL statement with either a semicolon (;) or a slash (/) on a new line
- Each PL/SQL statement with a slash (/) on a new line

The **PL/SQL editor** is triggered when opening the following object types: Functions, Procedures, Packages and Types. This editor helps you detect errors in your PL/SQL code during compilation. The output includes details such as the specific line and column where the error is detected, along with a link to go to the relevant position in the code block.

The **JavaScript worksheet mode** supports the Multilingual Engine syntax in Oracle Database release 21c. For more details, see Support for Multilingual Engine.

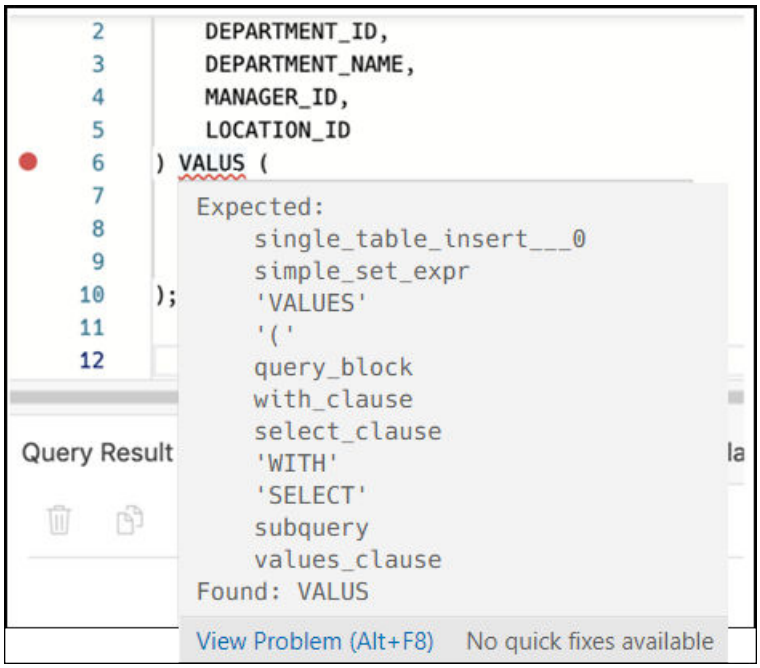
For SQL*Plus and SQLcl statements supported in the worksheet, see Supported SQL*Plus and SQLcl Commands in SQL Worksheet.

If you press **Ctrl+Space**, the editor provides you with a list of possible completions at the insertion point that you can use to autocomplete code that you are editing. This list is based on the code context at the insertion point.



The editor offers a comprehensive list of commands available through the Command Palette. To open the Command Palette, press **Ctrl+Shift+P**. For a list of keyboard shortcut keys, see Keyboard Shortcuts.

An error in the code is signified by a red dot in the left gutter and a squiggle line beneath the specific text. When you hover over it, you see a pop-up displaying possible fixes for resolving the error.



You can set code editor preferences using the **Preferences** option available in the top-right user drop-down list. Some of the code editor options that you can customize are Theme (Light, Dark and High contrast dark), Font size and family, Tab size, Word wrap, Ruler, Line numbers and so on.

The SQL toolbar contains icons for the following operations:

- **Worksheet** drop-down list
  - **Open**: Opens a file from the browser or device.
  - **Open Recent**: Displays the recently accessed files. If there are more than five files in the recent list, then a **More** link is displayed for viewing the additonal files.
  - **Save As**: Saves a file to the browser or device.
- **New File**

  - Select **Worksheet** to create a worksheet.
  - Select **PL/SQL** to create a PL/SQL object type. The editor switches to a PL/SQL mode.
  - Select **JavaScript** to create a JavaScript file. The editor switches to a JavaScript mode and **(JS)** is added to the file name.
- **Open** opens a file from your browser or device. To open a file from your device, in the Open File slider, click **Open File** and browse to select the file, or drag and drop the file into the slider.
- **Run Statement**
  - **Run Statement** executes the selected statements or the statement at the mouse pointer in the worksheet editor. The SQL statements can include bind variables and substitution variables of type VARCHAR2 (although in most cases, VARCHAR2 is automatically converted internally to NUMBER if necessary). A dialog box is displayed for entering variable values.
  - **Run Statement Without Pagination** executes the selected statements without wrapping it inside a `ROW_NUMBER()OVER` analytic function to implement paging. This means the query is run 'as is' in the worksheet, and only fetches the first 256 rows.
- **Run Script**
  - **Run as SQL Script** executes all statements in the worksheet editor using the Script Runner. The SQL statements can include bind variables (but not substitution variables) of type VARCHAR2 (although in most cases, VARCHAR2 is automatically converted internally to NUMBER if necessary). A dialog box is displayed for entering bind variable values.
  - **Run as JavaScript** executes the code as a JavaScript file. This option is used only with JavaScript code. If the JavaScript code is added to a PL/SQL block, then select "Run as SQL Script" to execute the script.
- **Compile** (for PL/SQL toolbar) performs a PL/SQL compilation of the subprogram.
- **Create Chart** creates a chart for the corresponding SQL statement entered in the editor. In contrast, you can partially highlight a subquery and create a chart. A slider window is displayed for entering the chart parameters. For a description of the fields, see Creating or Editing a Chart.
  If the SQL statement is syntactically incorrect or incomplete, an error/warning notification is displayed.
- **Explain Plan** generates the execution plan for the statement (internally executing the EXPLAIN PLAN statement). The execution plan is automatically displayed in the Explain Plan tab in the worksheet output pane. See Viewing the SQL Output
- **Autotrace** runs the statement and collects run time statistics and the actual execution plan. The Autotrace output is displayed in the Autotrace tab in the worksheet output pane. Currently, there are no preferences available.
- **Download Editor Content** downloads the content of the worksheet as a SQL file to the local system.
- **Format** formats the SQL statement in the editor, such as capitalizing the names of statements, clauses, keywords, and adding line breaks and indentation.
- **Clear** removes the statements from the editor.
- **Tour** provides a guided tour of the worksheet highlighting salient features and providing information that is useful if you are new to the interface.
- **Help** provides context-related help and provides a link to the help documentation.

- **Open in Fullscreen** opens the editor in full screen mode.

3.2.2.1 About Session State in Database Actions

In Database Actions, a connection to the database is stateless.

In a stateless environment, each HTTPS request from a client maps to a new database session. Therefore, a session begins and ends with every SQL statement or script execution.

As the session state is not maintained, session attributes do not persist and commands such as ROLLBACK and COMMIT do not apply. If a SQL statement or script executes successfully, an implicit commit is performed. If it executes with an error, an implicit rollback is performed.

Therefore, when needed, include the ROLLBACK and COMMIT commands or session attributes in the PL/SQL code block that is sent to the database for a session.

The only configuration commands that persist during a session in Database Actions are:

- `SET DEF[INE] <ON|OFF|prefix_character>`
- `SET ESC[APE] <ON|OFF|escape_character>`
- `SET TIMI[NG] <ON|OFF>`

3.2.2.2 Keyboard Shortcuts

This section lists the keyboard shortcuts for various commands in the SQL page.

Table 3-1 Keyboard Shortcuts

Windows	MacOS	Description
Ctrl + Enter	Cmd + Enter	Runs the code as query.
Alt + Tab	Option + Tab	Focus next element.
Ctrl+ Esc / Escape	Cmd + Esc / Escape	Remove focus from editor.
Ctrl + Down Arrow	Cmd + Down Arrow	Moves to the next SQL code from history.
Ctrl + Up Arrow	Cmd + Up Arrow	Moves to the previous SQL code from history.
Ctrl + S	Cmd + S	Saves the current worksheet.
Ctrl + O	Cmd + O	Opens the worksheet browser dialog.
Ctrl + I	Cmd + I	Downloads the content of the editor.
F1	Fn + F1	Opens the help topic.
Shift + Esc	Shift + Esc	Focus previous element.
F5	Fn + F5	Runs code as script.
F6	Fn + F6	Shows Autotrace.
F10	Fn + F10	Shows Explain Plan.
F11	Fn + F11	Creates a chart.
Ctrl + B	Cmd + B	Opens the "Convert Case" drop-down list.
Ctrl + F7	Cmd + Fn + F7	Formats code in the editor.
Ctrl + Space	Ctrl + Space	Autocompletes code (shows hints).

The following table lists the keyboard shortcuts for commands in the Command Palette.

Table 3-2 Command Palette Keyboard Shortcuts

Windows	MacOS	Description
Ctrl + Alt + Up	Cmd + Option + Up	Add Cursor Above
Ctrl + Alt + Down	Cmd + Option + Down	Add Cursor Below
Shift + Alt + I	Shift + Option + I	Add Cursors to Line Ends
Ctrl + K Ctrl + C	Cmd + K Cmd + C	Add Line Comment
Ctrl + D	Cmd + D	Add Selection To Next Find Match
Shift + Alt + Down	Shift + Option + Down	Copy Line Down
Shift + Alt + Up	Shift + Option + Up	Copy Line Up
Ctrl + U	Cmd + U	Cursor Undo
	Cmd + Backspace	Delete All Left
	Ctrl + K	Delete All Right
Ctrl + Shift + K	Shift + Cmd + K	Delete Line
Shift + Alt + Right	Shift + Ctrl + Cmd + Right	Expand Selection
Ctrl + F	Cmd + F	Find
Enter	Enter	Find Next
Ctrl + F3	Cmd + Fn + F3	Find Next Selection
Shift + Enter	Shift + Enter	Find Previous
Ctrl + Shift + F3	Shift + Cmd + Fn + F3	Find Previous Selection
	Cmd + E	Find With Selection
Ctrl + Shift + [	Option + Cmd + [	Fold
Ctrl + K Ctrl + 0	Cmd + K Cmd + 0	Fold All
Ctrl + K Ctrl + /	Cmd + K Cmd + /	Fold All Block Comments
Ctrl + K Ctrl + 8	Cmd + K Cmd + 8	Fold All Regions
Ctrl + K Ctrl + 1	Cmd + K Cmd + 1	Fold Level 1
Ctrl + K Ctrl + 2	Cmd + K Cmd + 2	Fold Level 2
Ctrl + K Ctrl + 3	Cmd + K Cmd + 3	Fold Level 3
Ctrl + K Ctrl + 4	Cmd + K Cmd + 4	Fold Level 4
Ctrl + K Ctrl + 5	Cmd + K Cmd + 5	Fold Level 5
Ctrl + K Ctrl + 6	Cmd + K Cmd + 6	Fold Level 6
Ctrl + K Ctrl + 7	Cmd + K Cmd + 7	Fold Level 7
Ctrl + K Ctrl + [	Cmd + K Cmd + [	Fold Recursively
Ctrl + Shift + \	Shift + Cmd + \	Go to Bracket
Ctrl + G	Ctrl + G	Go to Line...
Alt + F8	Option + Fn + F8	Go to Next Problem(Error, Warning, Info)
F8	Fn + F8	Go to Next Problem in Files (Error, Warning, Info)
F7	Fn + F7	Go to Next Symbol Highlight
Shift + Alt + F8	Shift + Option + Fn + F8	Go to Previous Problem (Error, Warning, Info)
Shift + F8	Shift + Fn + F8	Go to Previous Problem in Files (Error, Warning, Info)
Shift + F7	Shift + Fn + F7	Go to Previous Symbol Highlight
Ctrl +]	Cmd +]	Indent Line
Ctrl + Shift + Enter	Shift + Cmd + Enter	Insert Line Above

Table 3-2 (Cont.) Command Palette Keyboard Shortcuts

Windows	MacOS	Description
	Ctrl + J	Join Lines
Ctrl + K Ctrl + D	Cmd + K Cmd + D	Move Last Selection To Next Find Match
Alt + Down	Option + Down	Move Line Down
Alt + Up	Option + Up	Move Line up
F1 (All browsers) Ctrl + Shift + P (Google Chrome only)	Fn + F1	Open Command palette
Ctrl + [	Cmd + [	Outdent Line
Ctrl + K Ctrl + U	Cmd + K Cmd + U	Remove Line Comment
Ctrl + H	Option + Cmd + F	Replace
Ctrl + Shift + .	Shift + Cmd + .	Replace with Next Value
Ctrl + Shift + ,	Shift + Cmd + ,	Replace with Previous Value
Ctrl + Shift + L	Shift + Cmd + L	Select All Occurrences of Find Match
Alt + F1	Option + Fn + F1	Show Accessibility Help
Shift + F10	Shift + Fn + F10	Show Editor Context Menu
Ctrl + K Ctrl + I	Cmd + K Cmd + I	Show Hover
Shift + Alt + Left	Shift + Ctrl + Cmd + Left	Shrink Selection
Shift + Alt + A	Shift + Option + A	Toggle Block Comment
Ctrl + K Ctrl + L	Cmd + K Cmd + L	Toggle Fold
Ctrl + /	Cmd + /	Toggle Line Comment
Ctrl + M	Shift + Ctrl + M	Toggle Tab Key Moves Focus
	Shift + Ctrl + T	Transpose Letters
Ctrl + Space	Ctrl + Space	Trigger Suggest
Ctrl + K Ctrl + X	Cmd + K Cmd + X	Trim Trailing Whitespace
Ctrl + Shift +]	Option + Cmd +]	Unfold
Ctrl + K Ctrl + J	Cmd + K Cmd + J	Unfold All
Ctrl + K Ctrl + 9	Cmd + K Cmd + 9	Unfold All regions
Ctrl + K Ctrl +]	Cmd + K Cmd +]	Unfold Recursively

3.2.2.3 Support for Multilingual Engine

Database Actions provides support for Multilingual Engine (MLE) by enabling you to run JavaScript code in the worksheet.

Prerequisites

For the availability of MLE features in Database Actions, you need the:

- `DBMS_MLE` package in Oracle Database Release 21c and later versions. For more information, see DBMS_MLE in *Oracle Database PL/SQL Packages and Types Reference*.
- `EXECUTE DYNAMIC MLE` and `EXECUTE ON JAVASCRIPT` privileges assigned to you.

You can work with JavaScript code in the worksheet in the following ways:

- Create a JavaScript worksheet

- Execute JavaScript code in a standard worksheet
- Execute JavaScript code as a PL/SQL block

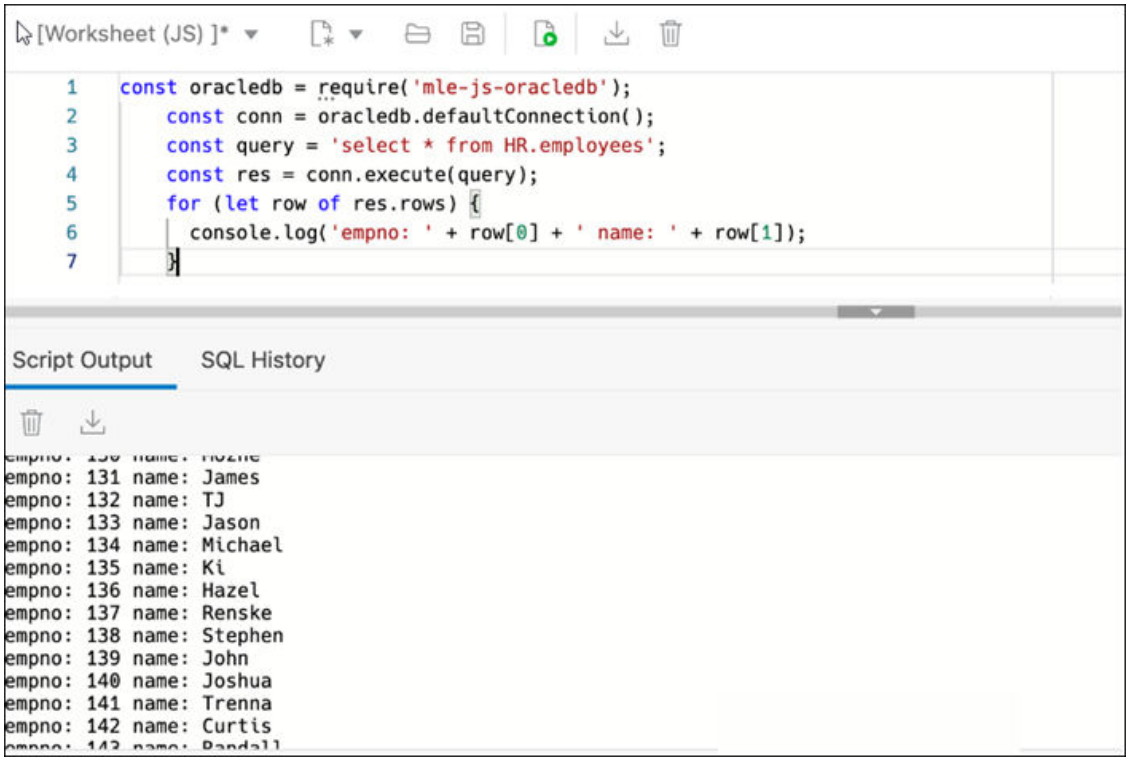
Create a JavaScript Worksheet

You can open the worksheet in JavaScript mode. You will also see the toolbar icons and output tabs change to reflect the JavaScript mode.

To create and save a JavaScript worksheet:

1. In the toolbar, expand the **New File** icon and select **JavaScript**.
2. When you enter code in the worksheet, the JavaScript code is automatically highlighted.
3. Execute the JavaScript code in the worksheet using **Run Script**.
4. Click **Save**.

   In the Files pane, **(JS)** is added to the JavaScript file name. This enables you to quickly identify the JavaScript file.

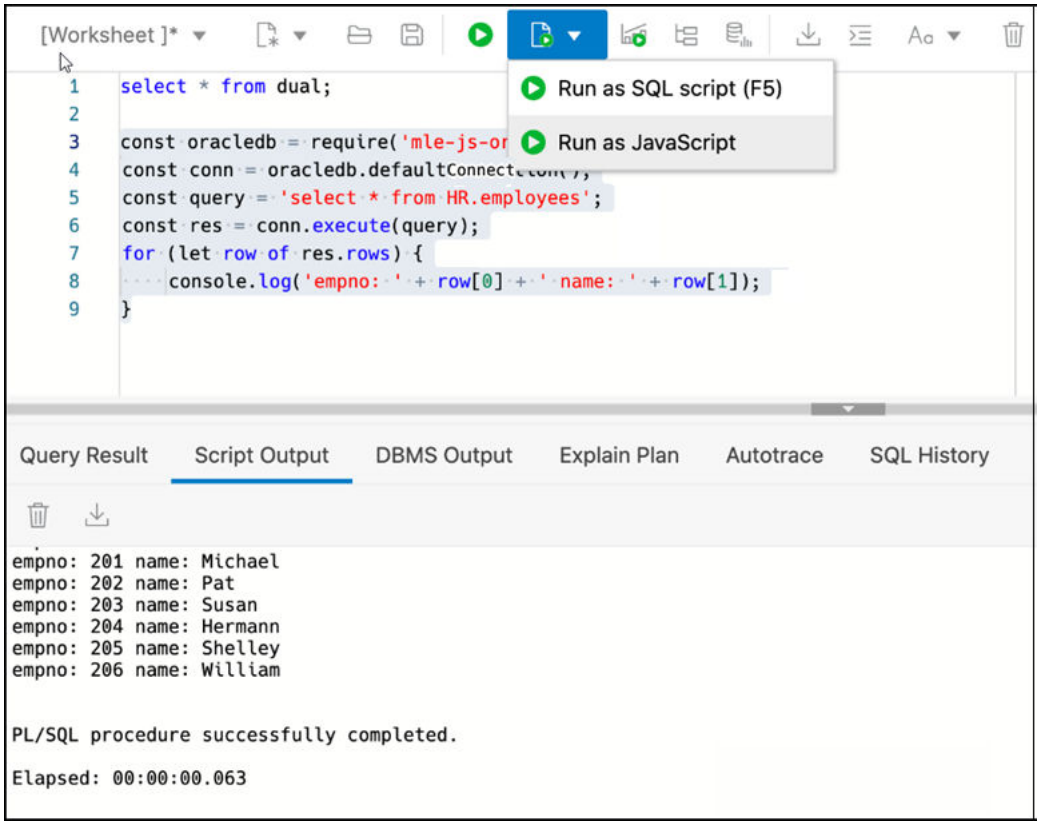


Execute JavaScript code in a standard worksheet

In a standard worksheet (when the worksheet is not in JavaScript mode):

1. Select the JavaScript code to execute.
2. In the worksheet toolbar, expand the **Run Script** icon and select **Run as JavaScript**.

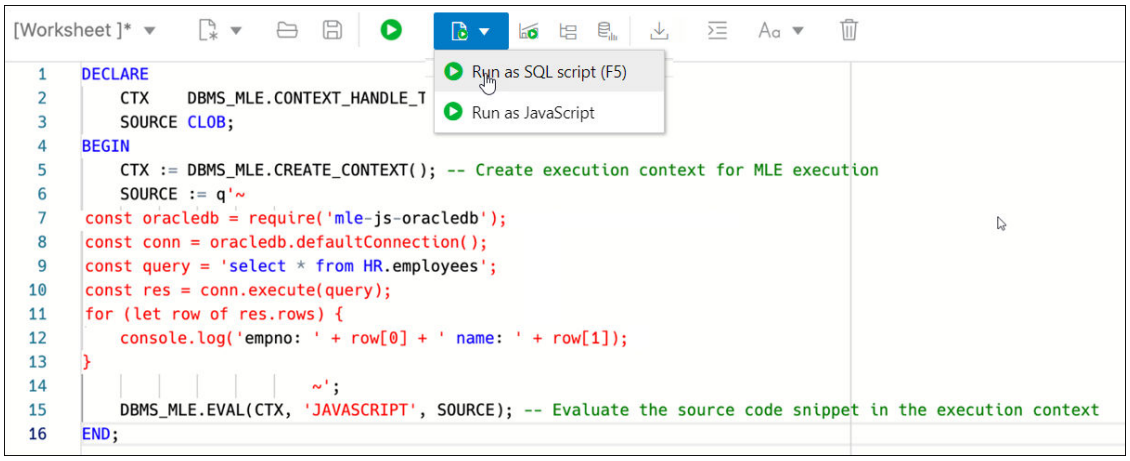
> **Note:**
>
> Selecting **Run as SQL script** results in an error.



Execute JavaScript code as a PL/SQL block

You can use a PL/SQL code block to execute JavaScript code.

From the worksheet toolbar, expand the **Run Script** icon and select **Run as SQL script (F5)**.

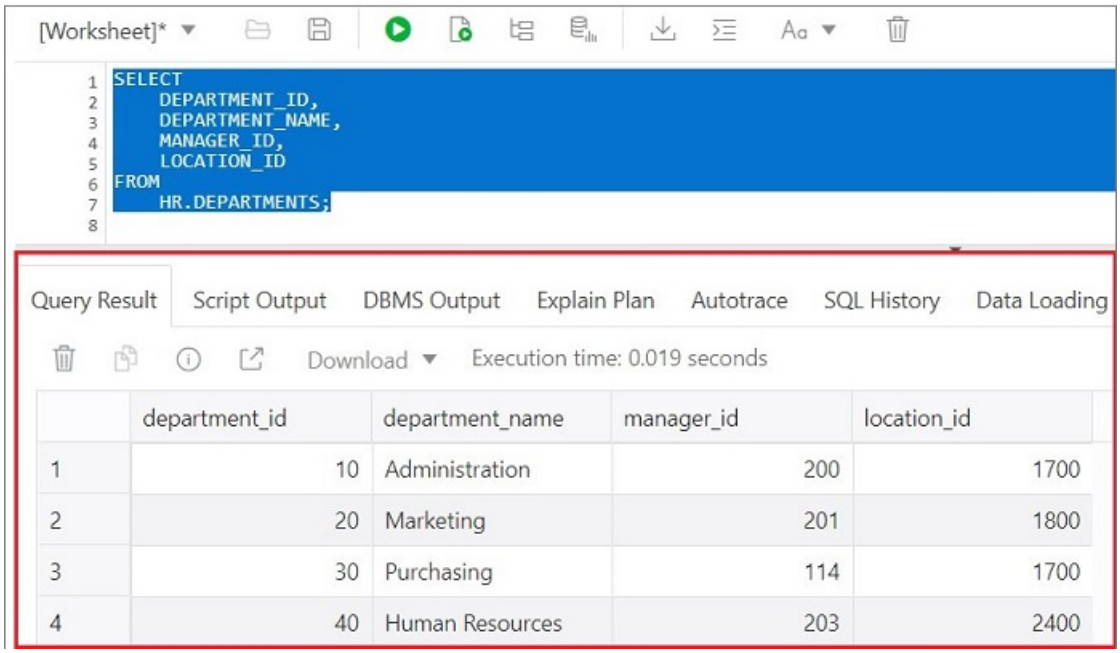


3.2.3 Viewing the SQL Output

The lower right pane in SQL displays the output of the operation executed in the SQL editor.

The following figure shows the output pane in the SQL page.

Figure 3-3 Output Pane



The output pane has the following tabs:

- **Query Result**: Displays the results of the most recent Run Statement operation in a display table.
- **Script Output**: Displays the text output from your statements executed as a script using the script engine.
- **DBMS Output**: Displays the output of DBMS_OUTPUT package statements.
- **Explain Plan**: Displays the plan for your query using the Explain Plan command. The default view is the diagram view. For more information, see Using the Explain Plan Diagram.
- **Autotrace**: Displays the session statistics and execution plan from `v$sql_plan` when executing a SQL statement using the Autotrace feature. Displays the output if you clicked the Autotrace icon.
- **SQL History**: Displays the SQL statements and scripts that you have executed. To re-enter a previously executed query in the worksheet, double-click the query in the history list. You can search for specific statements by clicking the Search icon. The Search functionality is case-sensitive, retrieves all entries that contain the search text, and does not require wildcard characters.

> **Note:**
>
> The executed statements are saved in the database, and are therefore available across all devices. For a read-only database, the SQL statements are saved on your browser.

- **Data Loading**: Displays a report of the total rows loaded and failed for all visible tables (including tables from other schemas).

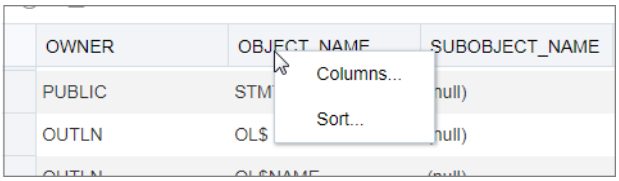
The icons in this pane are:

- **Clear output**: Clears the output.
- **Show info**: Displays the SQL statement for which the output is displayed.
- **Open in new tab**: Opens the query result or explain plan in a new window.
- **Download**: This is applicable only for Query Result. Enables you to download the query result to your local computer in CSV, JSON, XML, or TEXT (.tsv) format.

In the Query Result tab, in the display table, the context menu (right-click) for the row header consists of the following:

- **Columns** enables you to select columns to hide.
- **Sort** displays a dialog box for selecting columns to sort by. For each column, you can specify ascending or descending order, and you can specify that null values be displayed first.

Figure 3-4 Context Menu for Row Header



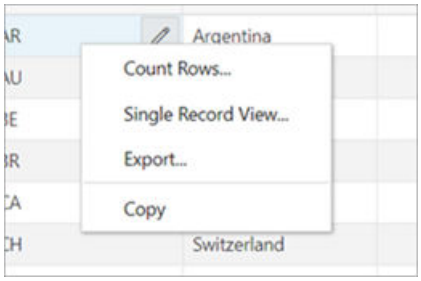
The context menu for the rest of the display table consists of the following commands:

- **Count Rows** displays the number of rows in the result set for your query.
- **Single Record View** enables you to view data for a table or view, one record at a time.
- **Export** generates the file for download based on the format selected, which can be XML, CSV (comma-separated values including a header row for column identifiers), Insert , Delimited, Fixed, HTML, JSON, or TEXT.
  - **Format**: Select the format to export from the drop-down list.
  - **Line Terminator**: Identifies the terminator for each line. The line terminator is not included in the data exported. If the preview page shows the data in one single row, the correct terminator is not specified.
  - **Header**: Controls whether the first row is a header row or the first row of data.
  - **Left and Right Enclosure**: Enclosures are used for character data and are optional. Enclosures are not included in the data exported.

> **Note:**
>
> If a popup blocker is enabled, it will prevent the file from downloading.

- **Copy** copies data from a cell or a row or a range of rows.

Figure 3-5 Context Menu



3.2.3.1 Using the Explain Plan Diagram

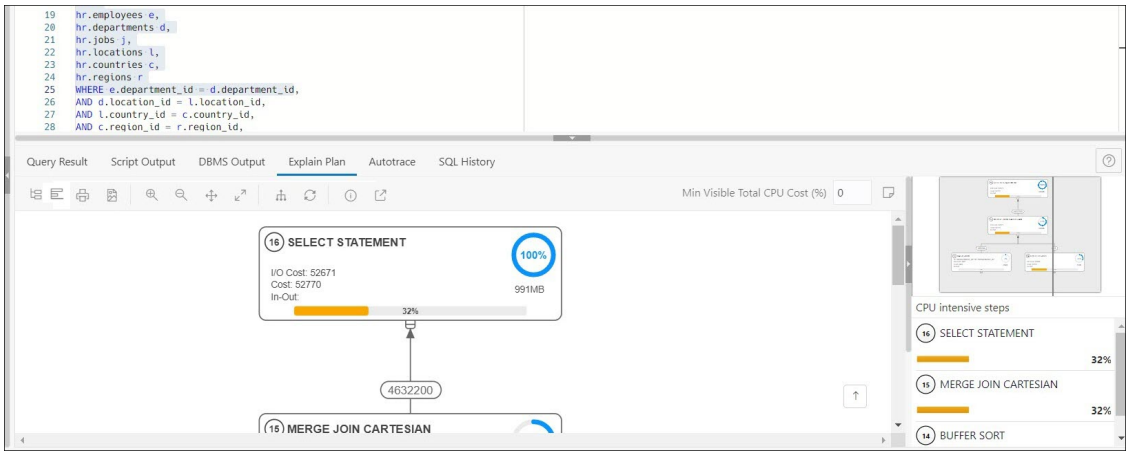
The Explain Plain diagram view is a graphical representation of the contents of `PLAN_TABLE`, which is the default table for results of the EXPLAIN PLAN statement. The hierarchical nature of the steps in the execution plan is depicted in the diagram.

By default, three levels of steps are visible in the diagram. You can use the **+/-** signs at the bottom of each step (available when the step has children) to expand or collapse. To view all steps in the diagram, use **Expand All** in the toolbar.

The diagram also provides the following details:

- Cardinality (number on the arrow to the parent step), which is the number of rows processed
- Operation and options applied in that step
- Execution order, which is the sequential number in the order of execution
- Access predicates CPU cost in percentage (orange bar)
- Total CPU cost for the step in percentage (blue circle)
- Estimated I/O Cost, Bytes processed and Cost metrics

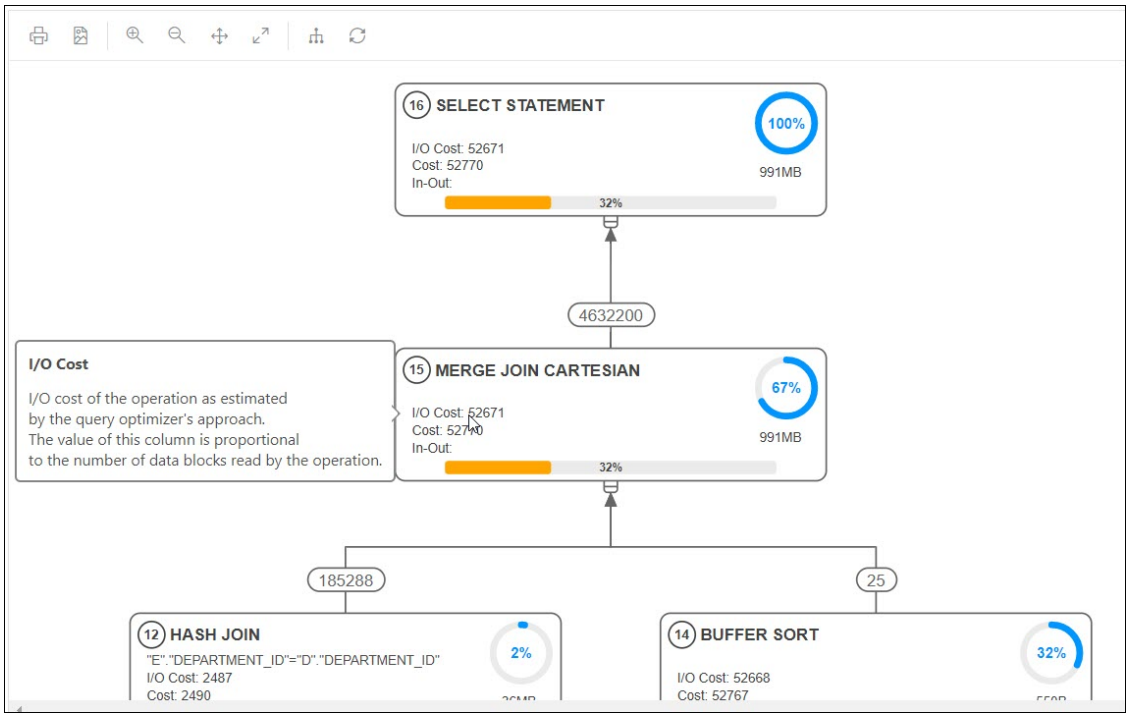
You can see a brief description pop-up when you hover over any of these statistics in a step.



The icons in the toolbar are:

- **Advanced View/Diagram View**: Switches between Advanced and Diagram views. The Advanced View displays data from `PLAN_TABLE` in a mixed tabular/tree view. The Diagram View displays the data in a flowchart format.
- **Chart View/Diagram View**: Switches between Chart and Diagram views. The Chart view displays data from `PLAN_TABLE` as a flame graph.
- **Print Diagram**: Prints the diagram.
- **Save to SVG**: Saves the diagram to file in SVG format.
- **Zoom In, Zoom Out**: If a step is selected in the diagram, clicking the Zoom In icon ensures that it remains at the center of the screen.
- **Fit Screen**: Fits the entire diagram in the visible area.
- **Actual Size**: Sets the zoom factor to 1.
- **Expand All**: Displays all steps in the diagram.
- **Reset Diagram**: Resets the diagram to the initial status, that is, only three levels of steps are displayed.
- **Show Info**: Shows the `SELECT` statement used by the Explain Plan functionality.
- **Open in New Tab**: Opens the diagram view in a new tab for better viewing and navigation. The diagram is limited to the initial SELECT statement.
- **Min Visible Total CPU Cost(%)**: Defines the threshold to filter steps with total CPU cost less than the the provided value.

  Enter a value between 0 and 100. There is no filtering for 0.
- **Plan Notes**: Displays the Explain Plan notes.



Properties

Double-click or press **Enter** on a selected step to open the Properties slider, which provides more information about that step. See PLAN_TABLE in *Oracle Database Reference* for a description of each property.

The Properties slider shows:

- All information for that step extracted from `PLAN_TABLE` in a tabular format. Nulls are excluded.

  You can select **JSON** to view the properties in JSON format.
- Information from `OTHER_XML` column of `PLAN_TABLE`.

  The information is displayed in JSON format.

Navigation

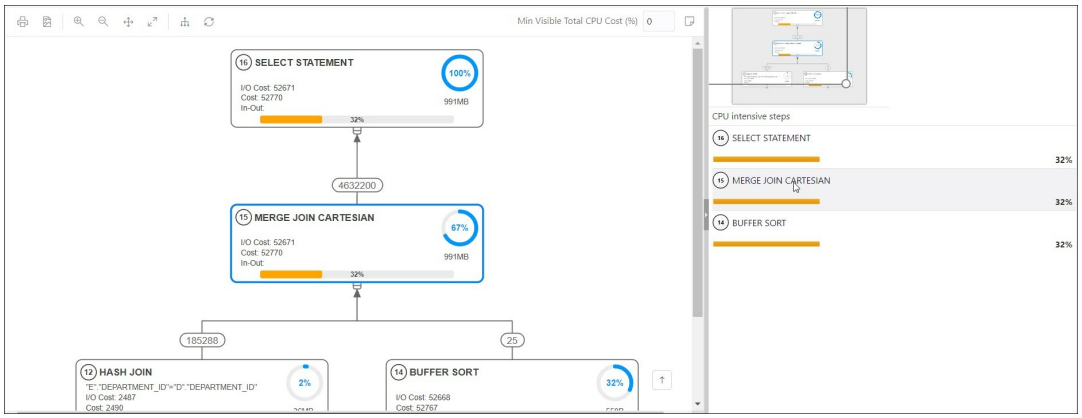
- Press the **Tab** key to move through the steps in the execution order. The selected step has a blue border around it.

  To move in the reverse direction, press the **Shift + Tab** keys.

  If no step is selected, pressing the **Tab** key selects the step with execution number 1.
- Depending on the zoom level, use horizontal and vertical scrollbars to view different parts of diagram.

  Click the left mouse button and hold it to pan the diagram around up and down.

  Use the  icon at the bottom right to scroll to the top of the diagram.
- The Diagram Navigator is at the top right corner and represents a smaller copy of the diagram on a grey background. The rectangle border allows zoom-in and zoom-out operations and moves to show different parts of the diagram.



The diagram navigator shows a list with steps having more than 1% CPU cost in descending order. Click a step in the list to navigate to the same step in the diagram, enabling you to see it in the context of the other steps.

3.2.4 Loading Data

In the SQL page, you can load data from one or more local files into one or more tables.

The file formats that you can load are CSV, XLS, XLSX, TSV, TXT, XML, JSON, and AVRO. For XML, JSON, and AVRO files, see Format Specifications for JSON, AVRO, and XML Files.

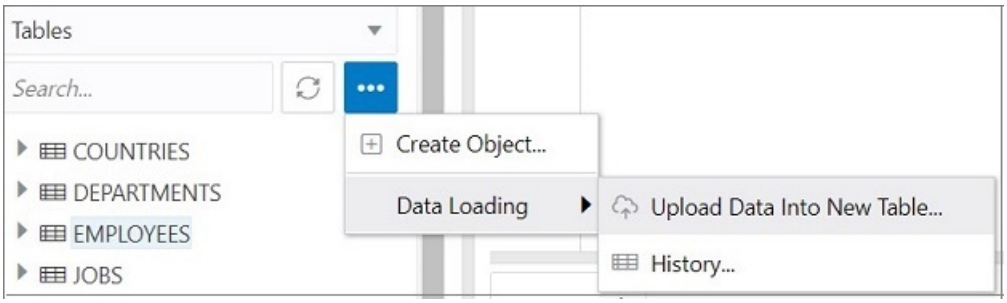
- Loading Data from a Local File to a New Table
- Loading Data from a Local File to an Existing Table
- Loading Data from Multiple Local Files into Mutiple Tables

3.2.4.1 Loading Data from a Local File to a New Table

To load data from a local file to a new table:

1. You can start in one of the following ways:
   - In the Navigator tab, in the left pane, click **Object submenu** ••• , select **Data Loading**, and then select **Upload Data into New Table**.

     Figure 3-6 Upload Data to New Table Option

   - In the Navigator tab, drag and drop the local file into the left pane. When you drag a file into the pane, the following message is displayed `Drop the file here to start`.

   The Upload Data into New Table is displayed. A preview of the data is displayed in a grid format.

2. Click **Show/Hide options** to display options that you can modify for data preview:
   - **Column names**: Select **Get from file** to display column headers in the first row.
   - **Encoding**: An option to select the encoding type is visible when the loaded file is in plain text format (CSV, TSV, or TXT). The default encoding type is UTF-8.
   - **Text enclosure** and **Field delimiter**: These options are visible only when the selected file is in plain text format (CSV, TSV, or TXT). Select or enter the character used in the source file for text enclosure and field delimiter.
   - **Rows to skip**: Enter or use the up and down arrows to select the number of rows to skip.
   - **Preview size**: Enter or use the up and down arrows to select the number of rows to preview.
   - **Limit rows to upload**: If you select this option, you need to specify the rows to load. Use the up and down arrows to select the number of rows to load.

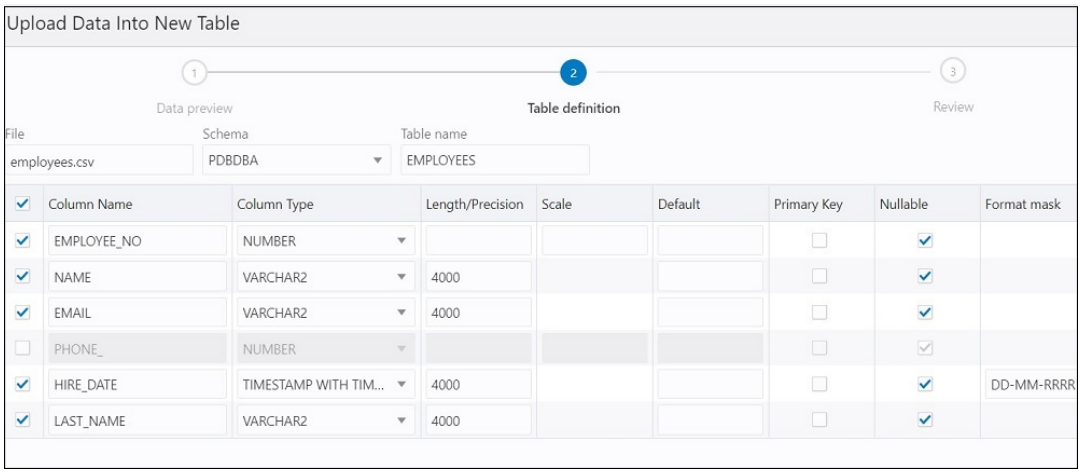
   To remove the options selected and the data preview, click **Clear**.

   After selecting the required options, click **Apply**, and then click **Next**.
3. In Table Definition, do the following:
   - In the **Table Name** field, enter a name for the target table.

- Select the check box at the beginning of a row to add the column to the target table.
- Select or enter values for column attributes such as Column Name, Column Type, Precision, Scale, Default, Primary Key and Nullable.
- The Format Mask column appears for date, timestamp and numeric types of data. For date and timestamp types, you must select a value from the drop-down list or type the value in the Format Mask field. For numeric types, the format mask is optional.

  For a date and timestamp column, you need to supply a compatible format mask that describes the data being uploaded. For example, if the date data looks like `12-FEB-2021 12.21.30`, you need to supply a date mask of `DD-MON-YYYY HH.MI.SS`. The format mask is automatically determined based on the data in the file. You need to review the suggested format mask and if needed, modify it by entering the format directly into the target cell.

Figure 3-7 Table Definition Step in Upload Data into New Table



Click **Next**.

4. Review the generated DDL code based on the selections made in the previous screens. The mapping of the source to target columns are also displayed.

   Click **Finish**. After the data is successfully loaded, the new table is displayed in the Navigator tab.

5. For a detailed report of the total rows loaded and failed, do one of the following:

   - Right-click the table in the Navigator tab, select **Data Loading**, and then select **History**. This displays the report for a specific table.
   - In the Navigator tab, select Object submenu , select **Data Loading**, and then select **History**. This displays the report for all tables in the schema that is selected in the Navigator tab.
   - In the worksheet output pane, select the **Data Loading** tab. This displays the report for all visible tables (including tables from other schemas).

   A summary of the data loaded is displayed in the History dialog. If any data failed to load, you can view the number of failed rows in the Failed Rows column. Click the failed rows column to open a dialog showing the failed rows.

   In the History dialog, you can also search for files loaded by schema name, table name, or file name. To remove the loaded files, click Remove all history .

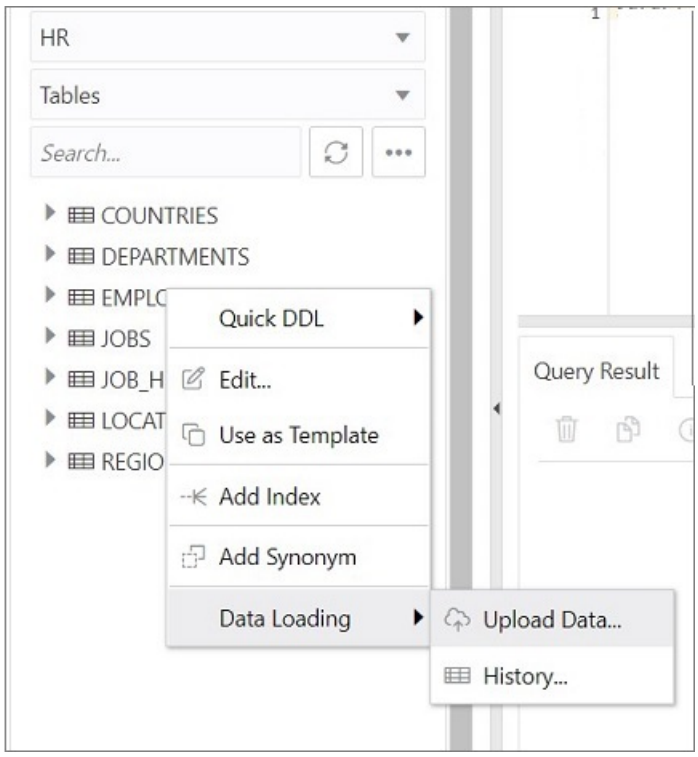
You can also load data from a file to a new table using the steps in Loading Data from Multiple Local Files into Mutiple Tables.

3.2.4.2 Loading Data from a Local File to an Existing Table

To load data from a local file to an existing table:

1. In the Navigator tab, in the left pane, right-click the table that you want to load data into, select **Data Loading**, and then select **Upload Data**.

Figure 3-8 Upload Data Option for an Existing Table



The Import data dialog is displayed.

2. Drag and drop the file from your system into the dialog, or click **Select Files** to browse for the file and open it.

   A preview of the data is displayed in a grid format.

3. Click **Show/Hide options** to display options that you can modify for data preview:
   - **Column names**: Select **Get from file** to display column headers in the first row.
   - **Encoding**: An option to select the encoding type is visible when the loaded file is in plain text format (CSV, TSV, or TXT). The default encoding type is UTF-8.
   - **Text enclosure** and **Field delimiter**: These options are visible only when the selected file is in plain text format (CSV, TSV, or TXT). Select or enter the character used in the source file for text enclosure and field delimiter.
   - **Rows to skip**: Enter or use the up and down arrows to select the number of rows to skip.

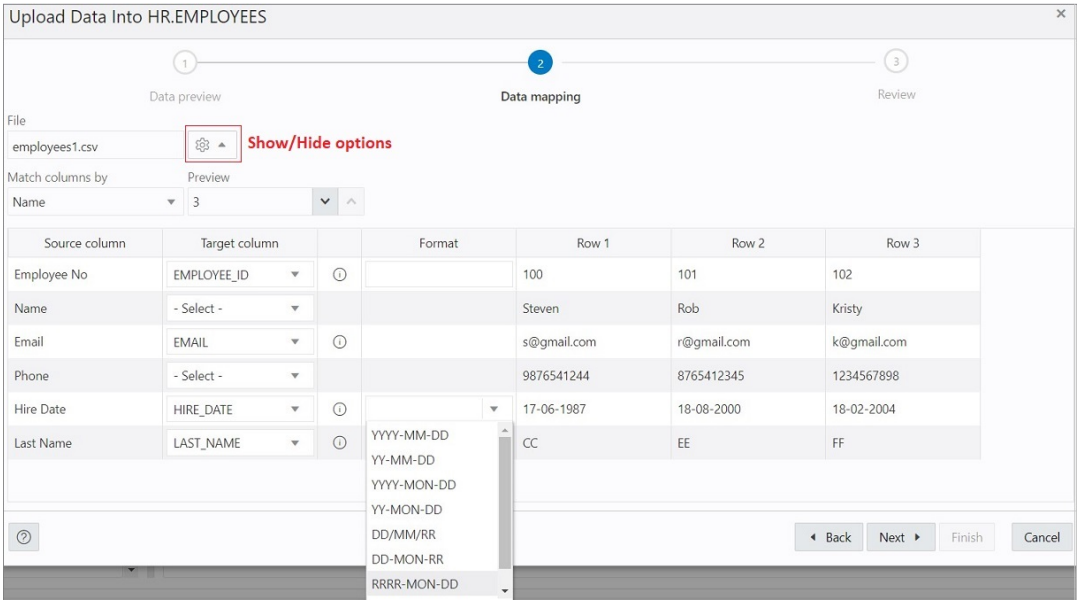
- **Rows to load**: Enter or use the up and down arrows to select the number of rows to load.
- **Preview size**: Enter or use the up and down arrows to select the number of rows to preview.

To remove the options selected and the data preview, click **Clear**.

After selecting the required options, click **Apply**, and then click **Next**.

4. In Data mapping, match the data in the file to the appropriate columns in the target table. By default, the matching is done using column name.

Figure 3-9 Data Mapping



To modify, click **Show/Hide options**  . In Match columns by:

- Select **Name** to match columns based on the name of the column in the target table.
- Select **Position** if you want to match columns based on the position of the column in the target table.
- Select **None** to remove the current selections and to select the target column for each source column from the drop-down list.

> **Note:**
>
> Based on the data in the file, attempts are made to automatically retrieve the correct format mask of date-based columns. If this is incorrect, you can change the suggested format by entering it directly into the target cell.

If there are any issues to resolve, you see a notification such as 

on the top right of the dialog.

Click **Next**.

5. A summary of the previous screens is displayed. Click **Finish**.

   The data will start uploading to the target table. After it is completed, an entry is added to the Log with the status of the operation. To view the Log, click the timestamp notification at the bottom of the page. If the operation is successful, a `Data Import Completed` notification is displayed.

6. For a detailed summary of the upload process, right-click the table in the Navigator tab, select **Data Loading**, and then select **History**. A summary of the data loaded is displayed in the Data Loading History dialog.

   If any data failed to load, you can view the number of rows in the Failed Rows column. Click the column to open a dialog showing the failed rows.

   In the Data Loading History dialog, you can also search for files loaded by schema name, table name, or file name. To remove the loaded files, click Remove all history .

   You can also load data from a file to an existing table using the steps in Loading Data from Multiple Local Files into Mutiple Tables.

3.2.4.3 Loading Data from Multiple Local Files into Mutiple Tables

To load data into multiple tables from multiple files:

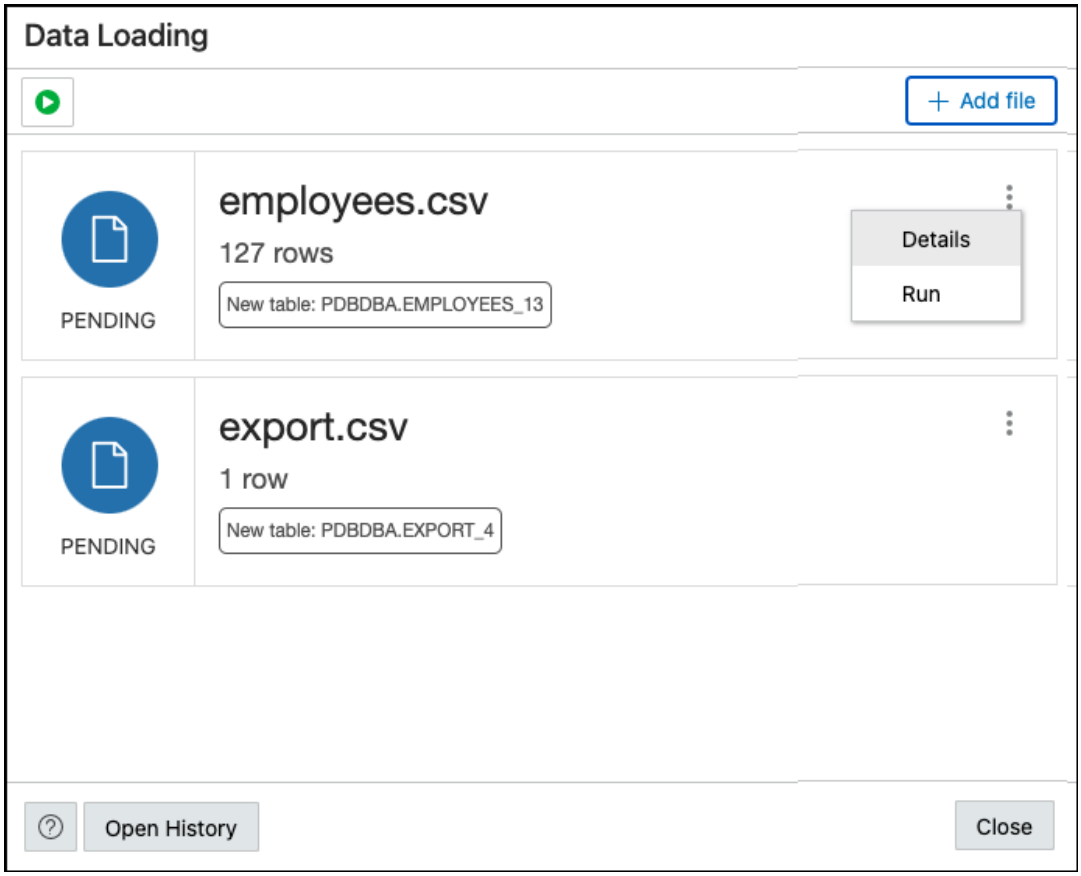
1. In the SQL page, on the top right, click **Data Load**.

   The Data Loading slider appears.

2. Click **Add File** to browse for one or more files and add them concurrently. Alternatively, you can drag and drop the files into the Data Loading slider.

   The added files are displayed as cards.

3. To open a file, click the name of the file or click at the top right corner of the card and select **Details**.



In the Details page, a preview of the file is displayed. Click **Next**. To go to the previous page, click **All Files**.

> **Note:**
>
> A preview of the entire file is shown the first time the file is loaded. Subsequently if any changes are made to the settings, the preview is set at a maximum of 10 rows. For a description of the settings, see step 2 in Loading Data from a Local File to a New Table.

4. In the Target Details page, the **Actions** field contains the following two options:
   - **Create New Table** to create a table. By default, this option is selected and the **Table Name** field is prefilled.
   - **Append to Existing Table** to add the file to an existing table. The data loader attempts to match the file name to an existing table name. Expand **Table Rows Preview** to preview the existing table.

   For a description of the mapping options, see step 3 in Loading Data from a Local File to a New Table and step 4 in Loading Data from a Local File to an Existing Table.

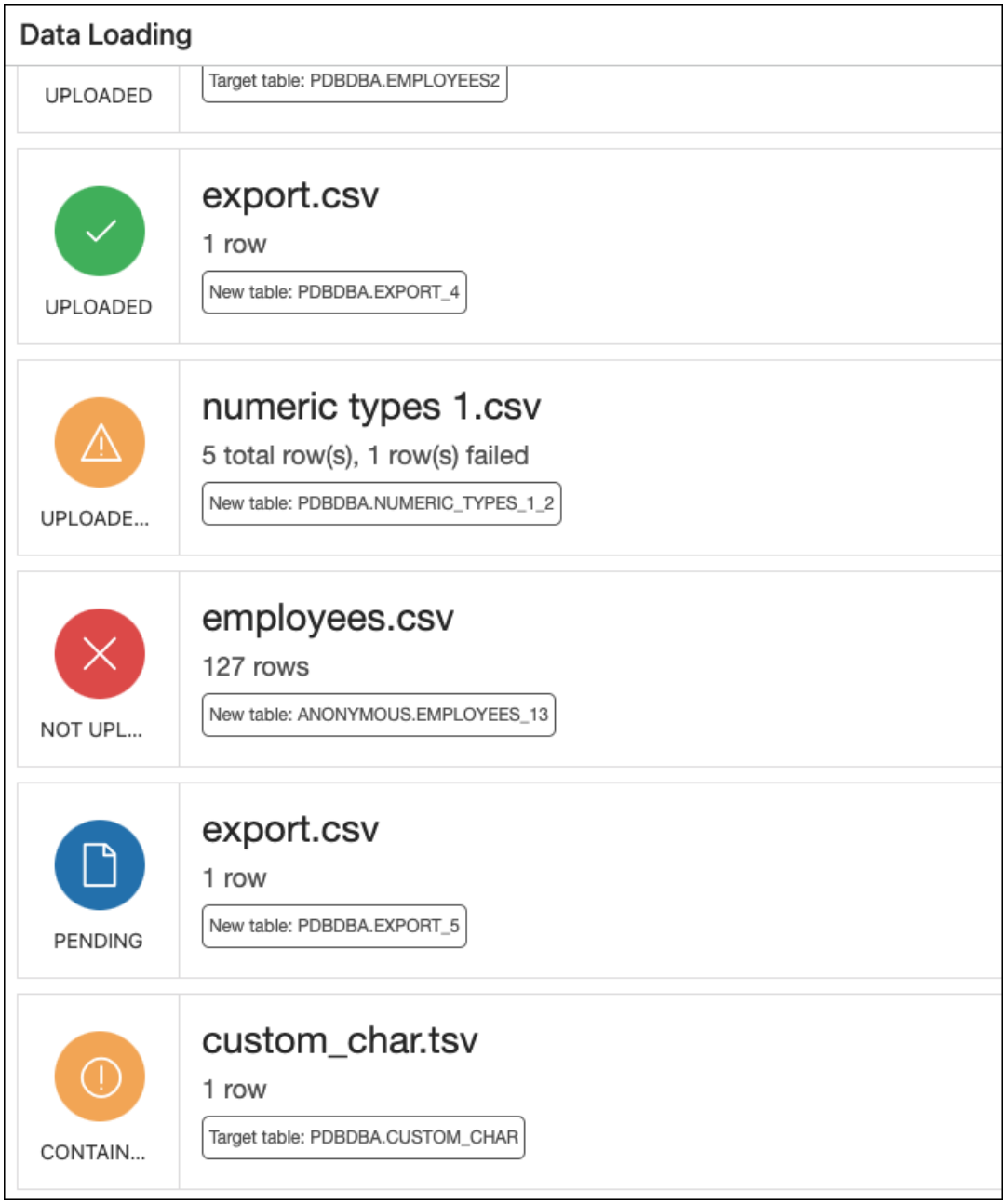
   Click **All Files** to return to the initial Data Loading page.
5. To load data into a table, at the top right of the card, click ⋮ and select **Run**.

To load data into multiple tables simultaneously, click  **Run All**.

The files that are successfully uploaded are displayed with a green colour icon and a check mark and the status `UPLOADED`.

Some other possible statuses are:

- `UPLOADED WITH WARNING` in yellow.
- `NOT UPLOADED BECAUSE OF ERRORS` in red.
- `UPLOADED WITH ERRORS` where you can see the number of failed rows.
- `CONTAINS WARNING(S)` indicates that something in the settings is wrong. This file will not be loaded until the warnings are resolved.



6. After a file is loaded, go to the Details slider to see the third step named **Results**.

   The Results step shows the total number of rows loaded along with the number of failed rows.

   To view all previous files that have been loaded along with their loading statuses, click **Open History**. You can filter by schema and table name.

3.2.4.4 Format Specifications for JSON, AVRO, and XML Files

Data has to be stored in a particular format for JSON, AVRO, and XML files to load them successfully into a table.

The format specifications are described in the following sections.

3.2.4.4.1 JSON and AVRO Files

For JSON and AVRO files, the conversion for primitive types to table columns is supported only for top-level data. Nested objects are saved as JSON strings such as VARCHAR2 (JSON) or CLOB (JSON).

> **Note:**
>
> JSON check constraints are available only for Oracle Database 12c and later releases.

Consider the following JSON file as an example:

```
[
  {
    "ItemNumber": 1,
    "Description": "One Magic Christmas",
    "Part": {
      "UnitPrice": 19.95,
      "UPCCode": 13131092899
    },
    "Quantity": 9,
    "Total": 179.55
  },
  {
    "ItemNumber": 2,
    "Description": "Lethal Weapon",
    "Part": {
      "UnitPrice": 17.95,
      "UPCCode": 85391628927
    },
    "Quantity": 5,
    "Total": 89.75
  }
]
```

The AVRO schema for this file:

```
{
     "type": "array",
     "items": {
          "type": "record",
          "fields": [
               {
                    "name": "ItemNumber",
                    "type": "int"
               },
               {
                    "name": "Description",
                    "type": "string"
               },
               {
                    "name": "Part",
                    "type": {
                         "type": "record",
                         "fields": [
                              {
                                   "name": "UnitPrice",
                                   "type": "float"
                              },
                              {
                                   "name": "UPCCode",
                                   "type": "float"
                              }
                         ]
                    }
               },
               {
                    "name": "Quantity",
                    "type": "int"
               },
               {
                    "name": "Total",
                    "type": "float"
               }
          ]
     }
}
```

Load the JSON file using "Upload Data" in the SQL page, and it is converted to the following table with two rows. `part` is a nested object that is assigned the column type CLOB (JSON) during data mapping.

	itemnumber	description	part	quantity	total
1	1	One Magic Christmas	{"UnitPrice":19.95,"UPCCode":131310...	9	179.55
2	2	Lethal Weapon	{"UnitPrice":17.95,"UPCCode":853916...	5	89.75

3.2.4.4.2 XML Files

This section lists the specifications for loading XML files.

- Attributes will have their own columns

  If the XML data is structured as:

```
<?xml version="1.0"?>
<catalog>
  <book id="bk102">
    <author>Ralls, Kim</author>
    <title>Midnight Rain</title>
    <genre>Fantasy</genre>
    <publisher>John Doe</publisher>
   </book>
</catalog>
```

  The generated columns in the table are `id`, `author`, `title`, `genre`, and `publisher`.

- Two or more levels of nesting are needed to parse the data

  In the following example, the data that needs to be parsed will not be located because it has only one level of nesting (catalog).

```
<?xml version="1.0"?>
<catalog>
  <author>Ralls, Kim</author>
  <title>Midnight Rain</title>
  <genre>Fantasy</genre>
  <publisher>John Doe</publisher>
</catalog>
```

  However, the following examples will work:

```
<?xml version="1.0"?>
<catalog>
  <book id="bk102">
    <author>Ralls, Kim</author>
    <title>Midnight Rain</title>
    <genre>Fantasy</genre>
    <publisher>John Doe</publisher>
  </book>
</catalog>
```

  or

```
<?xml version="1.0"?>
<catalog>
  <bookstore>
    <book id="bk102">
      <author>Ralls, Kim</author>
      <title>Midnight Rain</title>
      <genre>Fantasy</genre>
```

```
      <publisher>John Doe</publisher>
    </book>
  </bookstore>
</catalog>
```

or

```
<?xml version="1.0"?>
<catalog>
  <bookstore>
    <shelf>
      <book id="bk102">
        <author>Ralls, Kim</author>
        <title>Midnight Rain</title>
        <genre>Fantasy</genre>
        <publisher>John Doe</publisher>
      </book>
    </shelf>
  </bookstore>
</catalog>
```

- Special characters such as hyphen (-) and period (.) in tag names are replaced by underscore (_) in the column name

  XML tag names can contain hyphens and periods. Since the parser is converting XML to JSON, these characters become invalid object keys.

```
<?xml version="1.0"?>
<catalog>
  <book id="bk102">
    <author-name>Ralls, Kim</author-name>
    <title.1>Midnight Rain</title.1>
    <genre>Fantasy</genre>
    <publisher>John Doe</publisher>
   </book>
</catalog>
```

  The generated columns are `id`, `author_name`, `title_1`, `genre`, and `publisher`.

- First-level only-text tags are ignored

```
<?xml version="1.0"?>
<catalog>
  <library> New Age Library </library>
  <book id="bk102">
    <author>Ralls, Kim</author>
    <title>Midnight Rain</title>
    <genre>Fantasy</genre>
    <publisher>John Doe</publisher>
  </book>
</catalog>
```

  The `<library>` tag is ignored and only the content of the `<book>` tag is taken into account. The generated columns are `id`, `author`, `title`, `genre`, and `publisher`.

- First-level repetitive data is interpreted as an array of values

```
<?xml version="1.0" encoding="UTF-8"?>
<items id="orders">
    <item_number>1</item_number>
    <description>One Magic Christmas</description>
    <part>
      <unit_price>19.95</unit_price>
      <upccode>13131092899</upccode>
    </part>
    <quantity>9</quantity>
    <total>179.55</total>
    <item_number>2</item_number>
    <description>Lethal Weapon</description>
    <part>
      <unit_price>17.95</unit_price>
      <upccode>85391628927</upccode>
    </part>
    <quantity>5</quantity>
    <total>89.75</total>
</items>
```

The generated columns include `item_number`, `description`, `part`, and each column will have only one row with the following values respectively ([`1,2`], [`"One Magic Christmas","Lethal Weapon"`], [{`" unit_price":19.95,"upccode":13131092899}`, {`"unit_price":17.95,"upccode":85391628927`}] and so on for the remaining columns.

- Tags containing values and attributes are converted to object

```
<?xml version="1.0"?>
<catalog>
  <book id="bk102">
    <author country="ca">Ralls, Kim</author>
    <title>Midnight Rain</title>
    <genre>Fantasy</genre>
    <publisher>John Doe</publisher>
  </book>
</catalog>
```

The `<author>` tag is converted to a column and will have an object as value that is structured as:

```
{
      "_":"Ralls, Kim",
      "country":"ca"
    }
```

Note that the value of the tag has underscore ("_") as key and the attributes as "attribute_name": "attribute_value".

3.3 The Data Modeler Page

The Data Modeler page provides an integrated version of Oracle SQL Developer Data Modeler with basic reporting features. You can create diagrams from existing schemas, retrieve data dictionary information, generate DDL statements, and export diagrams.

To navigate to the Data Modeler page, do either of the following:

- In the Launchpad page, select the **Development** tab and click **Data Modeler**.
- Click **Selector** to display the navigation menu. Under Development, select **Data Modeler**.

The Data Modeler page consists of the left pane for navigating objects and diagrams, the editor pane for working with relational diagrams, and the right pane for viewing the properties of the object selected in the diagram. These panes are described in the following sections:

- Navigating Diagrams and Objects
- About the Data Modeler Editor

Related Topics

- *Oracle SQL Developer Data Modeler User's Guide*

3.3.1 Navigating Diagrams and Objects

The **Diagrams** tab lists the Data Modeler diagrams that have been saved.

When you right-click a diagram, you have options to open, save, delete, and view properties. Figure 3-10 shows a thumbnail preview that is displayed when you click the name of a diagram in the left pane.

Figure 3-10 Diagram Preview in Data Modeler

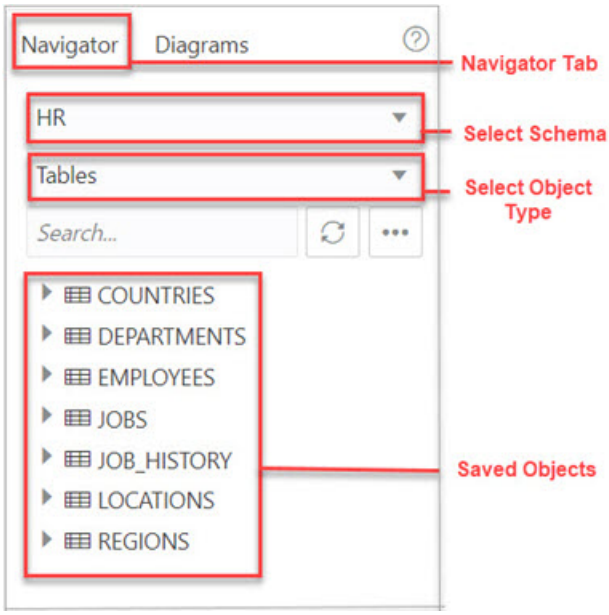
- To create a new diagram, in the Diagrams tab, click New Diagram .
- To open an existing diagram, in the Diagrams tab, right-click the diagram and click **Open**.

> **Note:**
>
> Diagrams are stored in the database, in the Database Actions user schema. When you use Data Modeler for the first time, a `OSDDMW_DIAGRAMS` table is created in your schema, to store the diagrams.
>
> Oracle SQL Developer Data Modeler can import from or export to the `OSDDMW_DIAGRAMS` table if a connection is provided. For more information, see Sharing Diagrams with SQL Developer Data Modeler.

The **Navigator** tab lists the objects that are available for each schema. You can select the schema and object type from the drop-down lists. For a particular schema, if a tables object is selected, the left pane displays all the tables that belong to the particular schema. You can expand a table to view its columns. Figure 3-11 indicates the important elements in the Navigator tab.

In a schema, you can search for objects in the Navigator tab by entering a few characters. You can also search the contents of a saved diagram. The search functionality is not case-sensitive, retrieves all matching entries, and does not require the use of wildcard characters.

Figure 3-11 Navigator Tab



The context menu for a table or view consists of:

Add Object to Diagram: Adds the selected object to the selected diagram. Alternatively, you can drag and drop an object into a selected diagram.

Add Object with dependencies to Diagram: Adds the selected object along with parent and child tables related to the object to the selected diagram.

Add Object as Star Schema to Diagram: Adds the selected object to the diagram and searches the data dictionary for foreign keys and implied foreign keys related to the object. The related tables or views are added to the diagram and the star schema layout is applied. See Implied Foreign Keys

Edit, Use as Template: Opens the Table Properties Dialog. Use to edit an existing object (such as a table or view) for a specific schema, or create a new object by using an existing one for the initial content. See The Table Properties Dialog and The View Properties Dialog.

Add/Edit Sequence: Opens the Sequence Properties Dialog. Use to create or edit a sequence for a selected schema. See The Sequence Properties Dialog.

Add Index: Opens the Index Properties Dialog. Use to create an index for a table. See The Index Properties Dialog.

Add Synonym: Opens the Synonym Properties Dialog. Use to create a synonym for a table or view in a selected schema. See The Synonym Properties Dialog

3.3.2 About the Data Modeler Editor

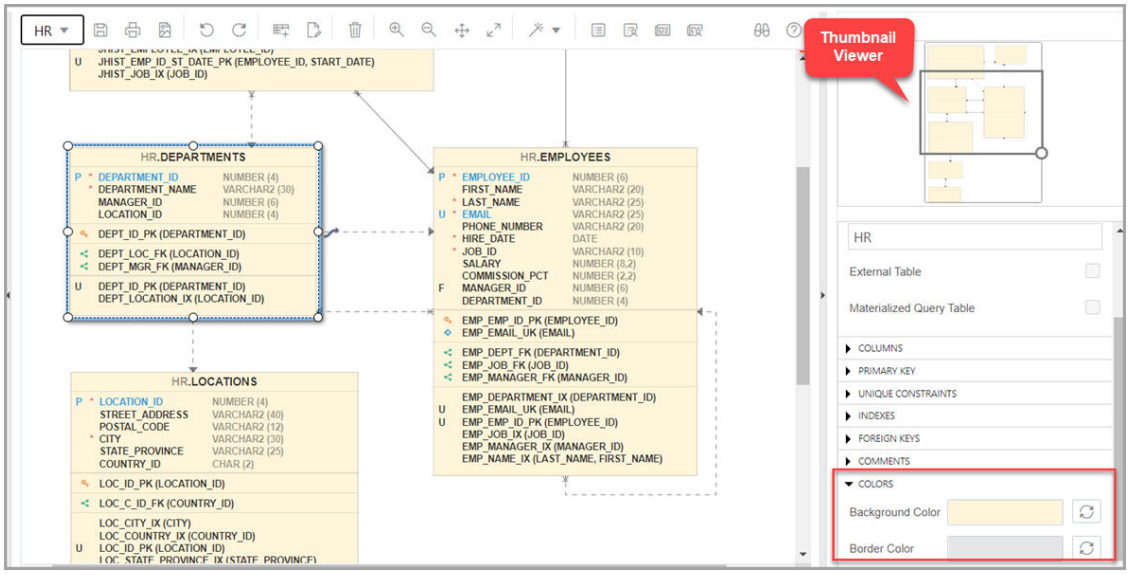
You can create and work with relational diagrams in the editor pane.

When an object is selected in the diagram, you can inspect the properties of the object in the right pane. For a table, the properties displayed are Columns, Primary Key, Unique Constraints, Indexes, Foreign Keys, Comments, and Colors. The only properties that you can edit are background and border color. Click Reset to return to the default colors.

Figure 3-12 shows the properties of an object selected in the editor pane.

You can drag the thumbnail viewer in the upper right corner to display the appropriate area in the editor pane. This is useful when you are working with long diagrams in the editor pane.

Figure 3-12 Inspect Properties of Object Selected in Editor Pane



When creating diagrams, you can do the following:

- Select an object and then drag to move it around.
- Adjust or move objects with the relationships intact.
- Add elbows to relationship lines to avoid intersecting with lines from other objects. Right-click the relationship line and drag to create the elbow. Click Remove vertex to restore the original shape.
- Resize objects by dragging the handles that are positioned around the box.
- Select and then right-click an object for the following options:
  - **DDL Preview**: Shows the DDL statements for the object.
  - **Update**: Updates any actions performed on the object.
  - **Delete**: Deletes the object.
  - **Edit**: Opens the Table Properties Dialog for editing a table object or the View Properties Dialog for a view object.
  - **Implied Foreign Keys**: Opens the Implied Foreign Keys Dialog. Use this option to define implied foreign keys for the object. See Implied Foreign Keys
  - **View JSON Data Guides**: Opens the JSON data guide diagram for the table or view. See About Viewing JSON Data Guides

The icons on the toolbar are:

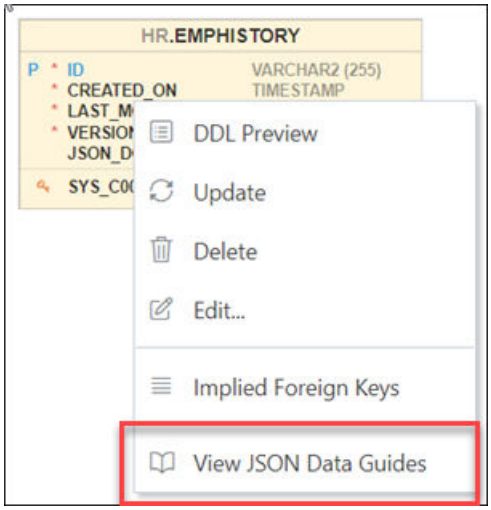
- **Save Diagram**: Saves the currently selected diagram. Diagrams are stored in a table that is created in the schema of the user.
- **Print Diagram**: Prints the selected diagram.

- **Save to SVG**: Saves the currently selected diagram to an image file in SVG format.
- **Add Objects to Diagram**: A dialog is displayed where you can select one or more objects of type tables or views from a specific schema into the selected diagram. Type * to list all the objects of an object type (tables or views) in the schema. You can also search by typing a few characters.
- **Add Note**: Adds notes to the selected diagram. Select the note to add text and to see the associated properties in the right pane, such as Font Size and Colors.
  - To enter text, expand TEXT and type information. Click **Apply**.
  - To select the background, border, or text color for the note, click the box and select the required color. To revert to the default color, click Reset .
- **Delete**: Deletes the selected object or objects from the diagram. To select multiple objects, press the CTRL key and select the objects.
- **Zoom In**: Displays more detail, and potentially fewer objects, in the currently selected diagram.
- **Zoom Out**: Displays less detail, and potentially more objects, in the currently selected diagram.
- **Fit Screen**: Makes all relevant objects fit in the window for the currently selected diagram, adjusting the sizes of shapes and text labels as needed.
- **Actual Size**: Adjusts the shapes and text labels in the currently selected diagram to the default sizes.
- **Layout: Auto Layout**: Rearranges the objects in the diagram to a layout that may be more meaningful and attractive. If you do not like the rearrangement, you can restore the previous layout by clicking Undo.
- **Layout: Star Layout**: Rearranges the objects in the diagram to a star schema layout, where the fact table is in the center and the associated dimension tables surround the fact table.
- **DDL Preview**: Shows the DDL statements for the object. You have the option to save or send the DDL statements to the worksheet. To choose what you want to display, click **Options**.
- **DDL Preview for Current Schema**: Shows the DDL statements for the current schema. You can send the DDL statements to the worksheet.
- **Diagram Report**: Generates a data dictionary report of everything in the diagram.
- **Schema Report**: Generates a data dictionary report of everything in the schema.
- **Help**: Displays the help for the Data Modeling editor.

3.3.2.1 About Viewing JSON Data Guides

A JSON data guide represents the JSON schema for documents in one column with JSON content. A table can have more than one column with JSON content.

The View JSON Data Guides option is available in the context menu for a table or view in a Data Modeler diagram.



This is applicable for tables, views and external tables that have columns with JSON content. Columns with JSON content are identified and data guides are retrieved for each such column. This process is quicker with the existence of a JSON search index.

The JSON schema (JSON Data Guide) is presented visually like an entity-relationship diagram. Arrays are presented as one-to-many relationships, contained objects as one-to-one relationships, and "oneOf" constructs as a box that surrounds possible choices. There is a column selector at the top right part of the page enabling you to select a column with JSON content for diagram presentation.

The following example is a JSON schema and its representation in a data guide diagram.

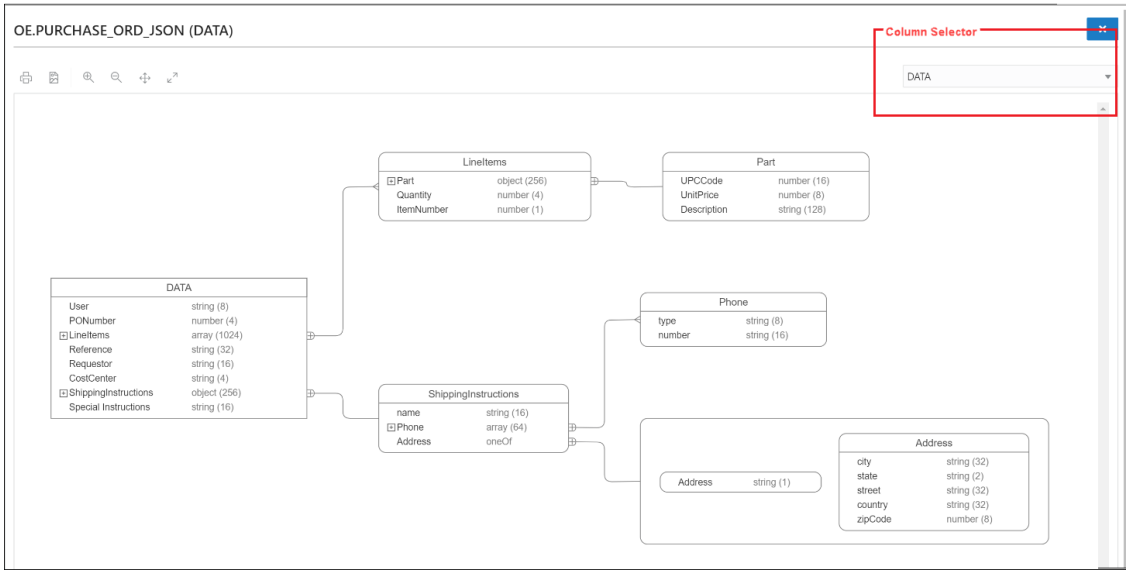
```
{
  "type" : "object",
  "properties" :
  {
    "User" :
    {
      "type" : "string",
      "o:length" : 8,
      "o:preferred_column_name" : "DATA$User"
    },
    "PONumber" :
    {
      "type" : "number",
      "o:length" : 4,
      "o:preferred_column_name" : "DATA$PONumber"
    },
    "LineItems" :
    {
      "type" : "array",
      "o:length" : 1024,
      "o:preferred_column_name" : "DATA$LineItems",
      "items" :
      {
        "properties" :
        {
          "Part" :
```

```
        {
          "type" : "object",
          "o:length" : 256,
          "o:preferred_column_name" : "DATA$Part",
          "properties" :
          {
            "UPCCode" :
            {
              "type" : "number",
              "o:length" : 16,
              "o:preferred_column_name" : "DATA$UPCCode"
            },
            "UnitPrice" :
            {
              "type" : "number",
              "o:length" : 8,
              "o:preferred_column_name" : "DATA$UnitPrice"
            },
            "Description" :
            {
              "type" : "string",
              "o:length" : 128,
              "o:preferred_column_name" : "DATA$Description"
            }
          }
        },
        "Quantity" :
        {
          "type" : "number",
          "o:length" : 4,
          "o:preferred_column_name" : "DATA$Quantity"
        },
        "ItemNumber" :
        {
          "type" : "number",
          "o:length" : 1,
          "o:preferred_column_name" : "DATA$ItemNumber"
        }
      }
    }
  },
  "Reference" :
  {
    "type" : "string",
    "o:length" : 32,
    "o:preferred_column_name" : "DATA$Reference"
  },
  "Requestor" :
  {
    "type" : "string",
    "o:length" : 16,
    "o:preferred_column_name" : "DATA$Requestor"
  },
  "CostCenter" :
  {
    "type" : "string",
```

```
    "o:length" : 4,
    "o:preferred_column_name" : "DATA$CostCenter"
  },
  "ShippingInstructions" :
  {
    "type" : "object",
    "o:length" : 256,
    "o:preferred_column_name" : "DATA$ShippingInstructions",
    "properties" :
    {
      "name" :
      {
        "type" : "string",
        "o:length" : 16,
        "o:preferred_column_name" : "DATA$name"
      },
      "Phone" :
      {
        "type" : "array",
        "o:length" : 64,
        "o:preferred_column_name" : "DATA$Phone",
        "items" :
        {
          "properties" :
          {
            "type" :
            {
              "type" : "string",
              "o:length" : 8,
              "o:preferred_column_name" : "DATA$type"
            },
            "number" :
            {
              "type" : "string",
              "o:length" : 16,
              "o:preferred_column_name" : "DATA$number"
            }
          }
        }
      },
      "Address" :
      {
        "oneOf" :
        [
          {
            "type" : "string",
            "o:length" : 1,
            "o:preferred_column_name" : "DATA$Address"
          },
          {
            "type" : "object",
            "o:length" : 128,
            "o:preferred_column_name" : "DATA$Address_1",
            "properties" :
            {
              "city" :
```

```
                    {
                      "type" : "string",
                      "o:length" : 32,
                      "o:preferred_column_name" : "DATA$city"
                    },
                    "state" :
                    {
                      "type" : "string",
                      "o:length" : 2,
                      "o:preferred_column_name" : "DATA$state"
                    },
                    "street" :
                    {
                      "type" : "string",
                      "o:length" : 32,
                      "o:preferred_column_name" : "DATA$street"
                    },
                    "country" :
                    {
                      "type" : "string",
                      "o:length" : 32,
                      "o:preferred_column_name" : "DATA$country"
                    },
                    "zipCode" :
                    {
                      "type" : "number",
                      "o:length" : 8,
                      "o:preferred_column_name" : "DATA$zipCode"
                    }
                  }
                }
              ]
            }
          }
        },
        "Special Instructions" :
        {
          "type" : "string",
          "o:length" : 16,
          "o:preferred_column_name" : "DATA$SpecialInstructions"
        }
      }
    }
```

Figure 3-13 JSON Data Guide Diagram



> **See Also:**
>
> JSON Data Guide in *Oracle Database JSON Developer's Guide*

3.4 Creating and Editing Database Objects

You can create and edit objects using Create and Edit Object wizards available from the Navigator tab in the SQL and Data Modeler pages.

The wizards for creating and editing various object types are described in the following sections:

- The Table Properties Dialog
- The Index Properties Dialog
- The Sequence Properties Dialog
- The View Properties Dialog
- The Synonym Properties Dialog
- Implied Foreign Keys
- The Materialized View Log Properties Dialog

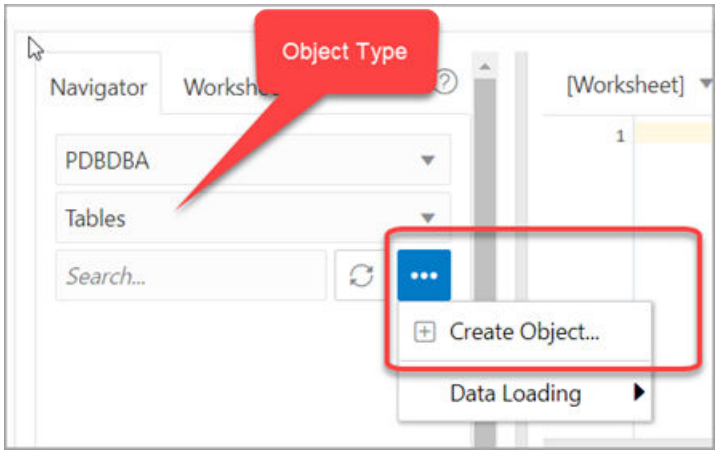
3.4.1 The Table Properties Dialog

The Table Properties Dialog is displayed when you create a table, edit an existing table, or create a table using an existing one as a template.

You can open the Table Properties dialog from the Navigator tab in SQL or Data Modeler.

To create a table for a specific schema, in the Navigator tab, select **Tables** from the object type drop-down list, click **Object submenu** ••• , and select **Create Object**.

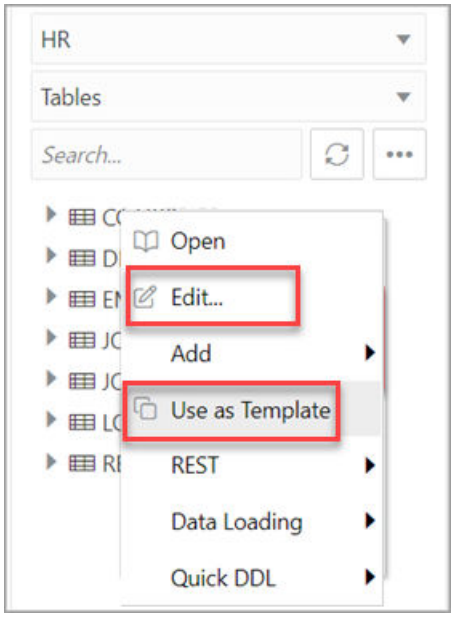
Figure 3-14 Create Object



To create a table from an existing one for a specific schema, right-click the table object in the Navigator tab, and select **Use as Template** as shown in Figure 3-15.

To edit a table for a specific schema, right-click a table object in the Navigator tab, and select **Edit**.

Figure 3-15 Edit Option for Existing Table



The table properties are grouped in several panes.

If you are editing an existing table, you can visit the panes in any order. If you click Create before you finish creating the table, right-click the table name, select **Edit**, and continue creating the table.

> **Note:**
>
> Editing a partitioned table is not recommended. To identify whether a table is partitioned or not, right-click the table name and select **Edit**. If the table is partitioned, a warning message will be displayed.

Schema: Database schema in which to create the table. By default, a new table is created in the existing schema or the schema that you are logged into.

Name: Name for the table.

The different panes in the dialog are described in the following sections:

- Columns Pane
- Primary Key Pane
- Unique Keys Pane
- Indexes Pane
- Foreign Keys Pane
- Table Constraints Pane
- Comments Pane
- Storage Pane
- External Table Properties Pane
- Materialized View Pane
- DDL Pane
- Output Pane

Related Topics

- *Oracle Database SQL Language Reference*

3.4.1.1 Columns Pane

Specifies properties for each column in the table.

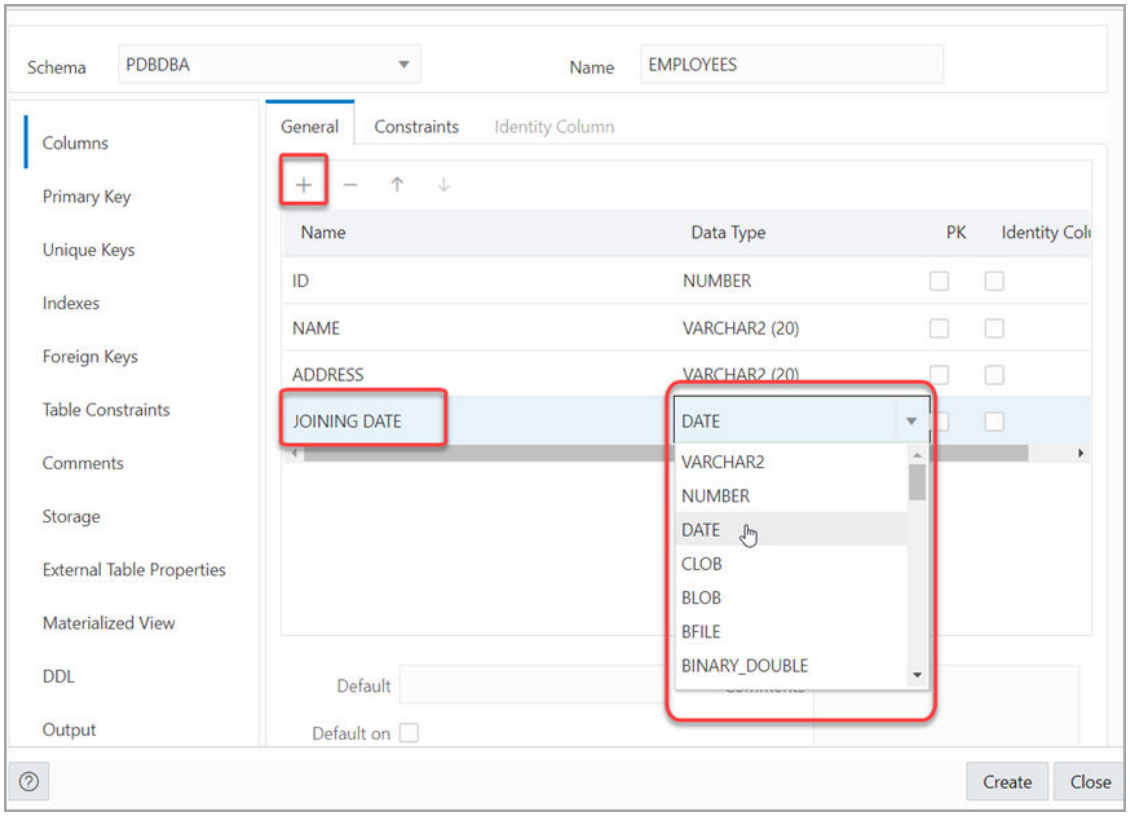
General tab

Lists the columns available in the table.

To add a column, click **Add Column** (+). A new row is added to the table below. Select the row and enter the details for the column.

To delete a column, select the row and click **Remove Column** (-). To move a column up or down in the table, select it and use the up-arrrow and down-arrow icons.

Figure 3-16 Add a Column



- **Name**: Name for the column.
- **Datatype**: Data type for the column.
- **Default**: If no value is specified, the default value inserted into the column when a row is inserted.
- **Default on NULL**: Applicable for Oracle Database 12c and later releases. If this option is selected, when a row is inserted into the table and the value specified for the column is NULL, the default value is inserted into the column.
- **Expression**: Expression for computing the value in the column.
- **Comments**: Optional descriptive comments about the column. Use this field to provide descriptions for the attributes.

In the table:

- **PK**: If this option is selected, the column becomes the primary key.
- **Identity Column**: If this option is selected, the column becomes an identity column. This is applicable only for Oracle Database 12c and later releases. For more details, see the Identity Column tab.

Constraints tab

Displays the Not Null and Check Constraints for a column. A check constraint requires values in a column to comply with a specified condition.

- **Not Null Constraint: Name**: Name for the Not Null constraint.

- **Not Null Constraint: Not Null**: If this option is selected, the column must contain data. You cannot specify no value or an explicit null value for this column when you insert a row. If this option is not checked, the column can contain either data or no data. A primary key column cannot be null.
- **Check Constraint: Name**: Name for the check constraint definition.
- **Check Constraint: Constraint**: Condition that must be met for a column to fulfill the check constraint. You can use any valid CHECK clause (without the CHECK keyword). For example, to indicate that the value in a numeric column named RATING must be from 1 to 10, you can specify: rating >=1 and rating <= 10.
- **Enabled**: If this option is selected, the constraint is checked when data is entered or updated in the column.
- **Deferrable**: If this option is selected, you can defer checking the validity of the constraint until the end of a transaction.
- **Initially Immediate**: If this option is selected, the constraint is checked whenever you add, update, or delete data from the column.
- **Validate**: If this option is selected, the existing data is checked to see if it conforms to the constraint.

Identity Column tab

Applicable for Oracle Database 12c and later releases. The Identity Column tab lists the properties of the identity column. This tab becomes available only after the Identity Column checkbox is selected for the column in the General tab. An identity column is an autoincrement column that can be used to identify a table row. Only one identity column can be specified for a table.

- **Generate**: Always means that values cannot be explicitly included for the identity column in INSERT OR UPDATE statements, By Default means values for the identity column are generated automatically if no values are specified explicitly, By Default on Null means values are generated for the column only when a NULL value is supplied.
- **Start with**: Starting value of the sequence.
- **Increment**: Interval between successive numbers in a sequence.
- **Min value**: Lowest possible value for the sequence. The default is 1 for an ascending sequence and -(10^26) for a descending sequence.
- **Max value**: Highest possible value for the sequence. The default is 10^27 for an ascending sequence and -1 for a descending sequence.
- **Cache** and **Cache size**: Cache causes sequence values to be preallocated in cache, which can improve application performance; Cache size indicates the number of sequence values preallocated in cache. No Cache causes sequence values not to be preallocated in cache.
- **Cycle**: Indicates whether the sequence "wraps around" to reuse numbers after reaching its maximum value (for an ascending sequence) or its minimum value (for a descending sequence). If cycling of values is not enabled, the sequence cannot generate more values after reaching its maximum or minimum value.
- **Order**: Indicates whether sequence numbers are generated in the order in which they are requested. If No Order is specified, sequence numbers are not guaranteed to be in the order in which they were requested.

3.4.1.2 Primary Key Pane

Specifies the primary key for the table.

The primary key is the column, or set of columns, that uniquely identifies each row in the table. If the Primary Key checkbox is selected for a column in the General tab, the corresponding fields are automatically populated in the Primary Key pane. You can make changes to the properties as required.

An index is automatically created on the primary key.

- **Name**: Name of the constraint to be associated with the primary key definition.
- **Enabled**: If this option is checked, the primary key constraint is enforced: that is, the data in the primary key column (or set of columns) must be unique and not null.
- **Index**: Name of the index to which the primary key refers.
- **Tablespace**: Name of the tablespace associated with the index.
- **Available Columns**: Lists the columns that are available to be added to the primary key definition. You can select multiple attributes, if required, for the primary key.
- **Selected Columns**: Lists the columns that are included in the primary key definition.

To add a column to the primary key definition, select it in Available Columns and click the Add (**>**) icon; to remove a column from the primary key definition, select it in Selected Columns and click the Remove (**<**) icon. To move all columns from available to selected (or the reverse), use the Add All (**>>**) or Remove All (**<<**) icon. To move a column up or down in the primary key definition, select it in Selected Columns and use the arrow buttons.

3.4.1.3 Unique Keys Pane

Specifies one or more unique constraints for the table.

A unique constraint specifies a column, or set of columns, whose data values must be unique: each data value must not be null, and it must not be the same as any other value in the column.

To add a unique constraint, click the Add button; to delete a unique constraint, select it and click the Remove button.

- **Name**: Name of the unique constraint.
- **Enabled**: If this option is selected, the unique constraint is enforced.
- **Rely**: If this option is selected, the constraint in NOVALIDATE mode is taken into account during query rewrite.
- **Deferrable**: If this option is selected, in subsequent transactions, constraint checking can be deferred until the end of the transaction using the SET CONSTRAINT(S) statement.
- **Initially Immediate**: If this option is selected, the constraint is checked at the end of each subsequent SQL statement.
- **Validate**: If the option is selected, the existing data is checked to see if it conforms to the constraint.
- **Index**: Name of the index to which the unique key refers.
- **Tablespace**: Name of the tablespace associated with the index.
- **Available Columns**: Lists the columns that are available to be added to the unique constraint definition.

- **Selected Columns**: Lists the columns that are included in the unique constraint definition.

To add a column to the unique constraint definition, select it in Available Columns and click the Add (**>**) icon; to remove a column from the unique constraint definition, select it in Selected Columns and click the Remove (**<**) icon. To move all columns from available to selected (or the reverse), use the Add All (**>>**) or Remove All (**<<**) icon. To move a column up or down in the unique constraint definition, select it in Selected Columns and use the arrow buttons.

3.4.1.4 Indexes Pane

Lists the indexes defined for the table.

To add an index, click Add Index (+); to delete an index, select it and click Remove Index (-).

- **Name**: Name of the index.
- **Type**: The type of Oracle index. `Non-unique` means that the index can contain multiple identical values; `Unique` means that no duplicate values are permitted; `Bitmap` stores rowids associated with a key value as a bitmap.
- **Tablespace**: Name of the tablespace for the index.
- **Expression**: A column expression is an expression built from columns, constants, SQL functions, and user-defined functions. When you specify a column expression, you create a function-based index.
- **Available Columns** and **Selected Columns**: Columns selected for the index. To select a column, click the column in the Available Columns box, and then click the click the Add Selected Columns icon to move it to the Selected Columns box.

3.4.1.5 Foreign Keys Pane

Specifies one or more foreign keys for the table.

A foreign key specifies a column ("local column"), whose data values match values in the primary key or unique constraint of another table.

- **Name**: Name of the foreign key definition.
- **Enabled**: If this option is checked, the foreign key is enforced.
- **Rely**, **Deferrable**, **Initially Immediate**, **Validate**: See the description of these fields in the Unique Keys pane.
- **Referenced Constraint: Schema**: Name of the schema containing the table with the primary key or unique constraint to which this foreign key refers.
- **Referenced Constraint: Table**: Name of the table with the primary key or unique constraint to which this foreign key refers.
- **Referenced Constraint: Constraint**: Name of the primary key or unique constraint to which this foreign key refers.
- **Referenced Constraint: On Delete**: Action to take automatically when a row in the referenced table is deleted and rows with that value exist in the table containing this foreign key: `NO ACTION` (shown by a crossing line in diagrams) performs no action on these rows; `CASCADE` (shown by an "X") deletes these rows; `SET NULL` (shown by a small circle) sets null all columns in those rows that can be set to a null value.
- **Assocations: Local Column**: Lists the column in the currently selected (local) table that is included in the foreign key definition. For each referenced column in the foreign key definition, select the name of a column in the edited table.

- **Associations: Referenced Column**: For each local column, identifies the column in the other (foreign) table that must have a value matching the value in the local column.

3.4.1.6 Table Constraints Pane

Specifies one or more check constraints for the table.

A check constraint specifies a condition that must be met when a row is inserted into the table or when an existing row is modified.

- **Name**: Name of the check constraint definition.
- **Check Condition**: Condition that must be met for a row to fulfil the check constraint. You can use any valid CHECK clause (without the CHECK keyword). For example, to indicate that the value in a numeric column named RATING must be from 1 to 10, you can specify rating >=1 and rating <= 10.
- **Enabled**: If this option is checked, the check constraint is enforced.

3.4.1.7 Comments Pane

Enter descriptive comments in this pane. This is optional.

3.4.1.8 Storage Pane

Enables you to specify storage options for the table.

When you create or edit a table or an index, you can override the default storage options.

- **Organization**: Specifies that the table is stored and organized with (Index) or without an index (Heap) or as an external table (External).
- **Tablespace**: Name of the tablespace for the table or index.
- **Logging**: ON means that the table creation and any subsequent INSERT operations against the table are logged in the redo log file. OFF means that these operations are not logged in the redo log file.
- **Row Archival**: YES enables in-database archiving, which allows you to archive rows within the table by marking them as invisible.

3.4.1.9 External Table Properties Pane

Specifies options for an external table.

An external table is a read-only table whose metadata is stored in the database but whose data in stored outside the database.

External Table

- **Access Driver Type**: Specifies the type of external table.
  - ORACLE_LOADER: Extracts data from text data files. This is the default access driver, which loads data from external tables to internal tables.
  - ORACLE_DATAPUMP: Extracts data from binary dump files. This access driver can perform both loads and unloads.
  - ORACLE_BIGDATA: Extracts data from Oracle Big Data Appliance.
  - ORACLE_HDFS: Extracts data stored in a Hadoop Distributed File System (HDFS).

  - ORACLE_HIVE: Extracts data stored in Apache HIVE.

- **Default Directory**: Specifies the default directory to use for all input and output files that do not explicitly name a directory object. The location is specified with a directory object, not a directory path.
- **Access Params**: Assigns values to the parameters of the specific access driver for the external table. Access parameters are optional.
  - OPAQUE_FORMAT_SPEC: The opaque_format_spec specifies all access parameters for the ORACLE_LOADER, ORACLE_DATAPUMP, ORACLE_HDFS, and ORACLE_HIVE access drivers. For descriptions of the access parameters, see *Oracle Database Utilities*. Field names specified in the opaque_format_spec must match columns in the table definition, else Oracle Database ignores them.
  - USING CLOB: Enables you to derive the parameters and their values through a subquery. The subquery cannot contain any set operators or an ORDER BY clause. It must return one row containing a single item of data type CLOB.
- **Reject Limit**: The number of conversion errors that can occur during a query of the external data before an Oracle Database error is returned and the query is aborted.
- **Project Column**: Determines how the access driver validates the rows of an external table in subsequent queries.
  - ALL: Processes all column values, regardless of which columns are selected, and validates only those rows with fully valid column entries. If any column value raises an error, such as a data type conversion error, the row is rejected even if that column was not referenced in the select list of the query.
  - REFERENCED: Processes only those columns in the select list of the query.

  The ALL setting guarantees consistent result sets. The REFERENCED setting can result in different numbers of rows returned, depending on the columns referenced in subsequent queries, but is faster than the ALL setting. If a subsequent query selects all columns of the external table, then the settings behave identically.
- **Location**: Specifies the data files for the external table. Use the Add (+) icon to add each location specification.
  - For ORACLE_LOADER and ORACLE_DATAPUMP, the files are named in the form `directory:file`. The directory portion is optional. If it is missing, then the default directory is used as the directory for the file. If you are using the ORACLE_LOADER access driver, then you can use wildcards in the file name. An asterisk (*) signifies multiple characters and a question mark (?) signifies a single character.
  - For ORACLE_HDFS, LOCATION is a list of Uniform Resource Identifiers (URIs) for a directory or for a file. There is no directory object associated with a URI.
  - For ORACLE_HIVE, LOCATION is not used. Instead, the Hadoop HCatalog table is read to obtain information about the location of the data source (which could be a file or another database).

Opaque Format Spec

Specifies all access parameters for the ORACLE_LOADER, ORACLE_DATAPUMP, ORACLE_HDFS, and ORACLE_HIVE access drivers.

For example:

```
RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
    TERRITORY AMERICA
```

```
    BADFILE log_file_dir:'ext_1v3.bad'
    LOGFILE log_file_dir:'ext_1v3.log'
    FIELDS TERMINATED BY "|" OPTIONALLY ENCLOSED BY '^' LDRTRIM
    ( PROD_ID,
      CUST_ID ,
      TIME_ID DATE(10) "YYYY-MM-DD",
      CHANNEL_ID ,
      PROMO_ID ,
      QUANTITY_SOLD ,
      AMOUNT_SOLD ,
      UNIT_COST ,
      UNIT_PRICE
    )
```

and the full statement:

```
CREATE TABLE SH.SALES_TRANSACTIONS_EXT
    (
      PROD_ID NUMBER ,
      CUST_ID NUMBER ,
      TIME_ID DATE ,
      CHANNEL_ID NUMBER ,
      PROMO_ID NUMBER ,
      QUANTITY_SOLD NUMBER ,
     AMOUNT_SOLD NUMBER (10,2) ,
     UNIT_COST NUMBER (10,2) ,
     UNIT_PRICE NUMBER (10,2)
     )
     ORGANIZATION EXTERNAL
     (
     TYPE ORACLE_LOADER
     DEFAULT DIRECTORY DATA_FILE_DIR
     ACCESS PARAMETERS
     (
     RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
    TERRITORY AMERICA
    BADFILE log_file_dir:'ext_1v3.bad'
    LOGFILE log_file_dir:'ext_1v3.log'
    FIELDS TERMINATED BY "|" OPTIONALLY ENCLOSED BY '^' LDRTRIM
    ( PROD_ID ,
      CUST_ID ,
      TIME_ID DATE(10) "YYYY-MM-DD",
      CHANNEL_ID ,
      PROMO_ID ,
      QUANTITY_SOLD ,
      AMOUNT_SOLD ,
      UNIT_COST ,
      UNIT_PRICE
    )
    )
     LOCATION ( "DATA_FILE_DIR":'sale1v3.dat')
     )
     REJECT LIMIT 100
 ;
```

CLOB Subquery

Type or copy and paste the query.

> **Note:**
>
> For more information about the external table fields, see *Oracle Database Utilities* and *Oracle Database SQL Language Reference*

3.4.1.10 Materialized View Pane

Specifies options for a materialized view.

Query: Contains the SQL code for the query part of the view definition. Type or copy and paste the query.

General

- **On Pre-built Table**: If **Yes**, an existing table is registered as a preinitialized materialized view. This option is particularly useful for registering large materialized views in a data warehousing environment. The table must have the same name and be in the same schema as the resulting materialized view, and the table should reflect the materialization of a subquery.
- **Reduced Precision**: **Yes** authorizes the loss of precision that will result if the precision of the table or materialized view columns do not exactly match the precision returned by the subquery. If **No** , the precision of the table or materialized view columns must exactly match the precision returned by the subquery, or the create operation will fail.
- **For Update**: Select **Yes** to allow a subquery, primary key, object, or rowid materialized view to be updated. When used in conjunction with Advanced Replication, these updates will be propagated to the master.
- **Real Time MV**: Select **Yes** to create a real-time materialized view or a regular view. A real-time materialized view provides fresh data to user queries even when the materialized view is not in sync with its base tables due to data changes. Instead of modifying the materialized view, the optimizer writes a query that combines the existing rows in the materialized view with changes recorded in log files (either materialized view logs or the direct loader logs). This is called on-query computation.
- **Query Rewrite**: If **Enable**, the materialized view is enabled for query rewrite, which transforms a user request written in terms of master tables into a semantically equivalent request that includes one or more materialized views.
- **Build**: Specifies when to populate the materialized view. **Immediate** indicates that the materialized view is to be populated immediately. **Deferred** indicates that the materialized view is to be populated by the next refresh operation. If you specify Deferred, the first (deferred) refresh must always be a complete refresh; until then, the materialized view has a staleness value of unusable, so it cannot be used for query rewrite.
- **Use Index**: If **Yes**, a default index is created and used to speed up incremental (fast) refresh of the materialized view. If **No**, this default index is not created. (For example, you might choose to suppress the index creation now and to create such an index explicitly later.)
- **Index Tablespace**: Specifies the tablespace in which the materialized view is to be created. If a tablespace is not selected, the materialized view is created in the default tablespace of the schema containing the materialized view.

- **Cache**: If **Yes**, the blocks retrieved for this table are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed. This setting is useful for small lookup tables. If **No**, the blocks are placed at the least recently used end of the LRU list.

Refresh Clause

- **Refresh**: Select Yes to enable refresh operations.
- **Refresh Type**: The method of refresh operation to be performed:
  - Complete Refresh: Executes the defining query of the materialized view, even if a fast refresh is possible.
  - Fast Refresh: Uses the incremental refresh method, which performs the refresh according to the changes that have occurred to the master tables. The changes for conventional DML changes are stored in the materialized view log associated with the master table. The changes for direct-path INSERT operations are stored in the direct loader log.
  - Force Refresh: Performs a fast refresh if one is possible; otherwise, performs a complete refresh.
- **Action**: The type of refresh operation to be performed:
  - On Demand: Performs a refresh when one of the DBMS_MVIEW refresh procedures are called.
  - On Commit: Performs a fast refresh whenever the database commits a transaction that operates on a master table of the materialized view. This may increase the time taken to complete the commit, because the database performs the refresh operation as part of the commit process.
  - Specify: Performs refresh operations according to what you specify in the Start on and Next fields.
- **Start Date**: Starting date and time for the first automatic refresh operation. Must be in the future.
- **Next Date**: Time for the next automatic refresh operation. The interval between the Start on and Next times establishes the interval for subsequent automatic refresh operations. If you do not specify a value, the refresh operation is performed only once at the time specified for Start on.
- **With**: Refresh type, which determines the type of materialized view:
  - Primary Key: Creates a primary key materialized view, which allows materialized view master tables to be reorganized without affecting the eligibility of the materialized view for fast refresh.
  - Row ID: Creates a rowid materialized view, which is useful if the materialized view does not include all primary key columns of the master tables.
- **Default Storage**: If Yes, DEFAULT specifies that Oracle Database will choose automatically which rollback segment to use. If you specify DEFAULT, you cannot specify the rollback_segment. DEFAULT is most useful when modifying, rather than creating, a materialized view.
- **Storage Type**: MASTER specifies the remote rollback segment to be used at the remote master site for the individual materialized view. LOCAL specifies the remote rollback segment to be used for the local refresh group that contains the materialized view. This is the default.
- **Rollback Segment**: Enter the name of the rollback segment.

- **Using Constraint**: If this option is checked, more rewrite alternatives can be used during the refresh operation, resulting in more efficient refresh execution. The behavior of this option is affected by whether you select Enforced or Trusted.
  - Enforced: Causes only enforced constraints to be used during the refresh operation.
  - Trusted: Enables the use of dimension and constraint information that has been declared trustworthy by the database administrator but that has not been validated by the database. If the dimension and constraint information is valid, performance may improve. However, if this information is invalid, then the refresh procedure may corrupt the materialized view even though it returns a success status.

3.4.1.11 DDL Pane

You can review and save the SQL statements that are generated when creating or editing the object. If you want to make any changes, go back to the relevant panes and make the changes there.

- For a new table, click **CREATE** to view the generated DDL statements.
- When you edit table properties, click **UPDATE** to view the generated ALTER statements. For a new table, the UPDATE tab will not be available.

When you are finished, click **Apply**.

3.4.1.12 Output Pane

Displays the results of the DDL commands. If there are any errors, go to the appropriate pane, fix the errors, and run the commands again. You can save to a text file or clear the output.

3.4.2 The Index Properties Dialog

The Index Properties dialog box is displayed when you create or edit an index.

To create an index for a selected schema, in SQL, in the Navigator tab, select **Indexes** from the object type drop-down list, click Object submenu , and select **Create Object**.

To edit an index for a selected schema, right-click a table object in the Navigator tab, and select **Edit**.

Definition pane

- **Schema**: Database schema that owns the table associated with the index.
- **Table**: Name of the table associated with the index.
- **Schema**: Database in which to create the index.
- **Tablespace**: Tablespace for the index.
- **Name**: Name of the index.
- **Type**: The type of Oracle index.
  - **Non-unique** means that the index can contain multiple identical values.
  - **Unique** means that no duplicate values are permitted.
  - **Bitmap** stores rowids associated with a key value as a bitmap.

- **Expression**: A column name or column expression. A column expression is an expression built from columns, constants, SQL functions, and user-defined functions. When you specify a column expression, you create a function-based index.
- **Available Columns**: Columns available in the table.
- **Selected Columns**: Columns selected for the index. Click Add Selected Columns > to move columns from the Available Columns list.
- **Order**: **ASC** for an ascending index (index values sorted in ascending order); **DESC** for a descending index (index values sorted in descending order).

DDL pane

You can review and save the SQL statements that are generated when creating or editing the index. If you want to make any changes, go back to the Definition pane and make the changes there.

- For a new index, click **CREATE** to view the generated DDL statements.
- When you edit index properties, click **UPDATE** to view the generated ALTER statements. For a new index, the UPDATE tab will not be available.

When you are finished, click **Apply**.

Output pane

Displays the results of the DDL commands. If there are any errors, go to the Definition pane, fix the errors, and run the commands again. You can save to a text file or clear the output.

3.4.3 The Sequence Properties Dialog

The Sequence Properties Dialog is displayed when you create or edit a sequence.

To create a sequence for a selected schema, in SQL, in the Navigator tab, select **Sequences** from the object type drop-down list, click Object submenu ••• , and select **Create Object**.

To edit a sequence for a selected schema, right-click a sequence object in the Navigator tab and select **Edit**.

A sequence is an object from which multiple users may generate unique integers. You can use sequences to automatically generate primary key values.

Properties pane

- **Schema**: Database schema in which to create the sequence.
- **Name**: Name of the sequence.
- **Start with**: Starting value of the sequence.
- **Increment**: Interval between successive numbers in a sequence.
- **Min value**: Lowest possible value for the sequence. The default is 1 for an ascending sequence and -(10^26) for a descending sequence.
- **Max value**: Highest possible value for the sequence. The default is 10^27 for an ascending sequence and -1 for a descending sequence.
- **Cache** and **Cache size**: `Cache` causes sequence values to be preallocated in cache, which can improve application performance; Cache size indicates the number of sequence values preallocated in cache. `No Cache` causes sequence values not to be preallocated in cache.

- **Cycle**: Indicates whether the sequence "wraps around" to reuse numbers after reaching its maximum value (for an ascending sequence) or its minimum value (for a descending sequence). If cycling of values is not enabled, the sequence cannot generate more values after reaching its maximum or minimum value.
- **Order**: Indicates whether sequence numbers are generated in the order in which they are requested. If **No Order** is specified, sequence numbers are not guaranteed to be in the order in which they were requested.

DDL pane

You can review and save the SQL statements that are generated when creating or editing the sequence. If you want to make any changes, go back to the Properties pane and make the changes there.

- For a new sequence, click **CREATE** to view the generated DDL statements.
- When you edit a sequence, click **UPDATE** to view the generated ALTER statements. For a new sequence, the UPDATE tab will not be available.

When you are finished, click **Apply**.

Output pane

Displays the results of the DDL commands. If there are any errors, go to the Properties pane, fix the errors, and run the commands again. You have save to a text file or clear the output.

3.4.4 The View Properties Dialog

The View Properties Dialog is displayed when you create or edit a view.

You can open the View Properties Dialog from the Navigator tab in SQL or Data Modeler.

To create a view for a selected schema, in SQL, in the Navigator tab, select **Views** from the object type drop-down list, click Object submenu , and select **Create Object**.

To create a view from an existing template for a selected schema, in the Navigator tab, select the view to create from, right-click and select **Use as Template**.

To edit a view for a selected schema, right-click a view object in the Navigator pane, and select **Edit**.

Schema: Database schema in which to create the view.

Name: Name of the view.

The different panes in the dialog are described in the following sections:

SQL Query pane

Enter or copy and paste the SQL query for the view, using the SELECT and FROM keywords along with the syntax needed to retrieve the desired information. A semicolon is not required after the query.

Columns pane

Click Refresh Columns to automatically populate the columns in this pane. You can edit the columns by selecting the required row and making changes in the Header Alias and Comments fields.

Storage pane

- **Force on Create**: Select **Yes** to create the view regardless of whether the base tables of the view or the referenced object types exist or the owner of the schema containing the view has privileges on them. These conditions must be true before any SELECT, INSERT, UPDATE, or DELETE statements can be issued against the view. If the view definition contains any constraints, CREATE VIEW ... FORCE fails if the base table does not exist or the referenced object type does not exist. CREATE VIEW ... FORCE also fails if the view definition names a constraint that does not exist.
- **Query Restriction**: **Read Only** prevents the view from being used to add, delete, or change data in the underlying table. **Check Option** prohibits any changes to the underlying table that would produce rows that are not included in this view.

Use the Primary Key, Unique Keys, Foreign Keys, and Comments panes to add or edit properties as required.

DDL pane

Based on the inputs provided, the DDL statements are generated. You can review and save the SQL statements. If you want to make any changes, go back to the relevant pane and make the changes there.

- For a new view, click **CREATE** to view the generated DDL statements.
- When you edit a view, click **UPDATE** to view the generated ALTER statements. For a new view, the UPDATE tab will not be available.

When you are finished, click **Apply**.

Output pane

Displays the results of the DDL commands. If there are any errors, go to the respective pane, fix the errors, and run the commands again. You can save to a text file or clear the output.

3.4.5 The Synonym Properties Dialog

The Synonym Properties Dialog is displayed when you create a synonym.

There are two ways of creating a synonym for a selected schema:

- In SQL, in the Navigator tab, right-click the object for which you want to create the synonym, and select **Add Synonym**. In this case, the only fields that you can edit in the Properties pane are Public and Synonym Name. The values of the remaining fields are predetermined by the object selected.
- In SQL, in the Navigator tab, select the object type as **Synonyms** or **Public Synonyms** from the drop-down list. Click Object submenu , and select **Create Object**. All the fields in the Properties dialog are available for edit.

The different panes in the dialog are described in the following sections:

Properties pane

- **Public**: If this option is checked, the synonym is accessible to all users. However, each user must have appropriate privileges on the underlying object to use the synonym. If this option is not checked, the synonym is a private synonym, and is accessible only within its schema.
- **Synonym Schema**: Database schema in which to create the synonym.
- **Synonym Name**: Name of the synonym. A private synonym must be unique within its schema; a public synonym must be unique within the database.
- **Object Type**: Specify the type of object to which this synonym refers.

- **Object Schema**: Schema containing the object or name to which this synonym refers.
- **DB Filter**: After selecting the Object Type and Object Schema, the list of objects of the selected type may be very long. To filter the object names, enter the search entry and click Refresh  . The Object Name field is auto-filled with appropriate object names in the drop-down list.
- **Object Name**: Select the name of the object to which this synonym refers.
- **DB Link**: Enter a complete or partial database link to create a synonym for a schema object on a remote database where the object is located. If you specify DB Link and omit schema, then the synonym refers to an object in the schema specified by the database link. Oracle recommends that you specify the schema containing the object in the remote database. If you omit DB Link, then Oracle Database assumes the object is located on the local database.

DDL pane

Based on the inputs provided, the DDL statements are generated. You can review and save the SQL statements. If you want to make any changes, go back to the relevant pane and make the changes there.

- For a new view, click **CREATE** to view the generated DDL statements.
- When you edit a view, click **UPDATE** to view the generated ALTER statements. For a new view, the UPDATE tab will not be available.

When you are finished, click **Apply**.

Output pane

Displays the results of the DDL commands. If there are any errors, go to the respective pane, fix the errors, and run the commands again. You can save to a text file or clear the output.

3.4.6 The Materialized View Log Properties Dialog

The Materialized View Log Properties dialog is displayed when you create or edit a materialized view log, which is a table associated with the master table of a materialized view.

To create a materialized view log for a selected schema, in SQL, in the Navigator tab, select **Materialized View Logs** from the object type drop-down list, click Object submenu  , and select **Create Object**.

To edit, right-click a materialized view log object in the Navigator pane and select **Edit**.

Schema: Database schema in which to create the materialized view log.

Table: Name of the master table of the materialized view to be associated with this materialized view log.

Properties tab

- **Row ID Logged**: **Yes** indicates that the rowid of all rows changed should be recorded in the materialized view log. **No** indicates that the rowid of all rows changed should not be recorded in the materialized view log.
- **PK Logged**: **Yes** indicates that the primary key of all rows changed should be recorded in the materialized view log; **No** indicates that the primary key of all rows changed should not be recorded in the materialized view log.
- **New values**: **Yes** saves both old and new values for update DML operations in the materialized view log; **No** disables the recording of new values in the materialized view log.

If this log is for a table on which you have a single-table materialized aggregate view, and if you want the materialized view to be eligible for fast refresh, you must specify Yes.

- **Object ID Logged**: For a log on an object table only: **Yes** indicates that the system-generated or user-defined object identifier of every modified row should be recorded in the materialized view log. **No** indicates that the system-generated or user-defined object identifier of every modified row should not be recorded in the materialized view log.
- **Cache**: For data that will be accessed frequently, **CACHE** specifies that the blocks retrieved for this log are placed at the most recently used end of the least recently used list in the buffer cache when a full table scan is performed. This attribute is useful for small lookup tables. **NOCACHE** specifies that the blocks are placed at the least recently used end of the LRU list.
- **Parallel**: If **YES**, parallel operations will be supported for the materialized view log.
- **Sequence Logged**: **Yes** indicates that a sequence value providing additional ordering information should be recorded in the materialized view log. **No** indicates that a sequence value providing additional ordering information should not be recorded in the materialized view log. Sequence numbers (that is, Yes for this option) are necessary to support fast refresh after some update scenarios.
- **Commit SCN**: If this option is enabled, the database is instructed to use commit SCN data rather than timestamps.
- **Available Columns** and **Selected Columns**: Additional columns, which are non-primary-key columns referenced by subquery materialized views, to be recorded in the materialized view log. To select one or more filter columns, use the arrow buttons to move columns from Available to Selected.

Storage tab

- **Tablespace**: Tablespace in which the materialized view log is to be created
- **Logging**: **YES** or **NO**, to establish the logging characteristics for the materialized view log.
- **Buffer Mode**: Select **KEEP** to put blocks from the segment into the KEEP buffer pool. Select **RECYCLE** to put blocks from the segment into the RECYCLE pool. Select **DEFAULT** to indicate the default buffer pool.
- **Percent Free**: Specify a whole number representing the percentage of space in each data block of the database object reserved for future updates to rows of the object. The value of PCTFREE must be a value from 0 to 99.
- **Percent Used**: Specify a whole number representing the minimum percentage of used space that Oracle maintains for each data block of the database object. PCTUSED is specified as a positive integer from 0 to 99 and defaults to 40.
- **Initrans**: Specify the initial number of concurrent transaction entries allocated within each data block allocated to the database object. This value can range from 1 to 255 and defaults to 1.
- **Freelists**: In tablespaces with manual segment-space management, for objects other than tablespaces and rollback segments, specify the number of free lists for each of the free list groups for the table, partition, cluster, or index. The default and minimum value for this parameter is 1, meaning that each free list group contains one free list.
- **Freelist Groups**: In tablespaces with manual segment-space management, specify the number of groups of free lists for the database object you are creating.
- **Initial Extent**: Specify the size of the first extent of the object.
- **Next Extent**: Specify in bytes the size of the next extent to be allocated to the object.

- **Percent Increase**: In locally managed tablespaces, Oracle Database uses the value of PCTINCREASE during segment creation to determine the initial segment size and ignores this parameter during subsequent space allocation.
- **Min Extent**: In locally managed tablespaces, Oracle Database uses the value of MINEXTENTS in conjunction with PCTINCREASE, INITIAL and NEXT to determine the initial segment size.
- **Max Extent**: This storage parameter is valid only for objects in dictionary-managed tablespaces. Specify the total number of extents, including the first, that Oracle can allocate for the object.
- **Unlimited**: Select this option if you want extents to be allocated automatically as needed. Oracle recommends this setting as a way to minimize fragmentation.

Purge tab

- **Type**: In **IMMEDIATE SYNCHRONOUS**, the materialized view log is purged immediately after refresh. This is the default. In **IMMEDIATE ASYNCHRONOUS**, the materialized view log is purged in a separate Oracle Scheduler job after the refresh operation.
- **Deferred, Start With, Next, Repeat Interval**: Sets up a scheduled purge that is independent of the materialized view refresh and is initiated during CREATE or ALTER MATERIALIZED VIEW LOG statement.

Refresh tab

- **Type**: **Synchronous Refresh** creates a staging log that can be used for synchronous refresh. Specify the name of the staging log to be created. The staging log will be created in the schema in which the master table resides. **Fast Refresh** creates a materialized view log that can be used for fast refresh. The materialized view log will be created in the same schema in which the master table resides. This is the default.

DDL pane

Based on the inputs provided, the DDL statements are generated. You can review and save the SQL statements. If you want to make any changes, go back to the relevant pane and make the changes there.

- For a new materialized view log, click **CREATE** to view the generated DDL statements.
- When you edit a materialized view log, click **UPDATE** to view the generated ALTER statements. For a new materialized view log, the UPDATE tab will not be available.

When you are finished, click **Apply**.

Output pane

Displays the results of the DDL commands. If there are any errors, go to the respective pane, fix the errors, and run the commands again. You can save to a text file or clear the output.

3.4.7 Implied Foreign Keys

Implied foreign keys are dependencies that exist between tables but are not defined in the database. In a data warehouse environment, it is a common practice not to create foreign keys. However, it becomes necessary to show these dependencies for presentation or reporting purposes.

You can display implied foreign keys for objects in a star schema by defining them or by discovering them in the data dictionary. It is possible to have more than one source for implied foreign keys.

Define Implied Foreign Keys

You can defined implied foreign keys in two ways:

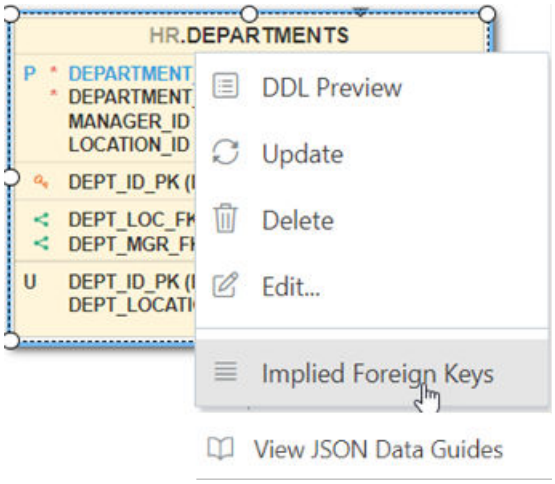
- Using the Implied Foreign Keys dialog
- By dragging the arrow to the referenced object in the diagram

Using the Implied Foreign Keys Dialog

You can define implied foreign keys using the Implied Foreign Keys dialog in Data Modeler.

1. In a Data Modeler diagram, right-click an object (table or view) and select **Implied Foreign Keys**.

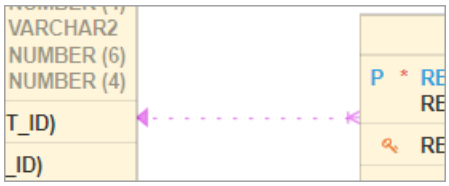
Figure 3-17 Select Implied Foreign Keys for an Object



The Implied Foreign Keys dialog is displayed.

2. In the Implied Foreign Keys dialog, click + to add an entry in the grid.
3. Select the entry in the grid to enable and enter values in the following fields:
   - **Referenced Object**: Object in the diagram that has a dependency to the source object.
   - **Local Column**: Name of the column in the source object.
   - **Referenced Column**: Name of the column in the targeted object.
   - **Discovery Sources**: Automatically prefilled, displays whether the implied foreign keys have been defined or were discovered in the data dictionary.
4. Click **OK**. The implied foreign key dependency is displayed with a dotted line on the diagram.

Figure 3-18 Dotted Line Between Two Objects

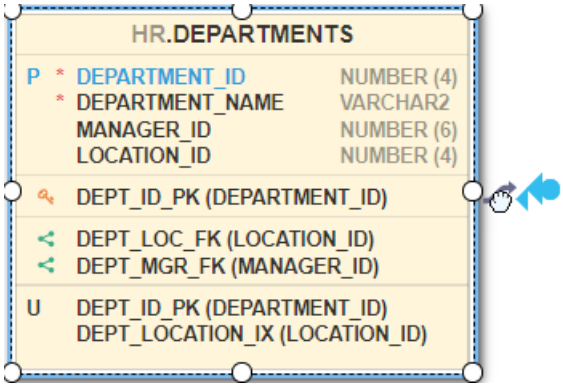


By Dragging the Arrow to the Referenced Object in the Diagram

You can also define an implied foreign key in the following way:

1. Select the source object on the diagram.
2. Click and drag the small curved arrow with a blue indicator to the referenced object. The dependency will be displayed with a dotted line on the diagram.

Figure 3-19 Drag Blue Indicator



3. Right-click the source object and enter the column names in the Implied Foreign Keys dialog.

Discover Implied Foreign Keys in the Data Dictionary

A star schema can be discovered by searching for several types of definitions in the data dictionary.

In the Navigator tab, right-click an object (table or view) and select **Add Object as Star Schema to Diagram**. The object must be a fact table. The data dictionary is then searched for joins and dependencies related to the object, such as:

- Foreign keys defined for the selected table to other tables. If implied foreign keys are later discovered for the same columns, they will not be displayed on the diagram.
- Joins used in the definitions of bitmap join indexes, materialized views with aggregates, and attribute clustering.
- Dependencies based on dimension definitions and column name matching in fact table.
- Fact and dimension definitions for Analytic views and OLAP cube and dimensions.

4

The REST Pages

The REST pages enable you to develop and protect ORDS (Oracle REST Data Services) based RESTful web services for your database.

To navigate to the REST pages, do either of the following:

- In the Launchpad page, select the **Development** tab and click **REST**.
- Click **Selector**  to display the navigation menu. Under Development, select **REST**.

Topics

4.1 About RESTful Web Services

Representational State Transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web. A service is described as RESTful when it conforms to the tenets of REST.

A RESTful service has the following characteristics:

- Data is modeled as a set of resources. Resources are identified by URIs.
- A small, uniform set of operations are used to manipulate resources (for example, GET, POST, PUT, DELETE).
- A resource can have multiple representations (for example, a blog might have an HTML representation and a RSS representation).
- Services are stateless and because it is likely that the client will want to access related resources, these should be identified in the representation returned, typically by providing hypertext links.

4.2 RESTful Services Terminology

The following are some major terms related to RESTful services.

- **RESTful service**: An HTTP web service that conforms to the tenets of the RESTful architectural style.

- **Resource module**: An organizational unit that is used to group related resource templates.
- **Resource template**: An individual RESTful service that is able to service requests for some set of URIs (Universal Resource Identifiers). The set of URIs is defined by the URI pattern of the Resource Template
- **URI pattern**: A pattern for the resource template. Can be either a route pattern or a URI template, although you are encouraged to use route patterns.
  - **Route pattern**: A pattern that focuses on decomposing the path portion of a URI into its component parts. For example, a pattern of `/objects/:object/:id?` will match `/objects/emp/101` (matches a request for the item in the `emp` resource with `id` of `101`) and will also match `/objects/emp/` (matches a request for the `emp` resource, because the `:id` parameter is annotated with the `?` modifier, which indicates that the `id` parameter is optional).
  - **URI template**: A simple grammar that defines the specific patterns of URIs that a given resource template can handle. For example, the pattern `employees/{id}` will match any URI whose path begins with `employees/`, such as `employees/2560`.
- **Resource handler**: Provides the logic required to service a specific HTTP method for a specific resource template. For example, the logic of the GET HTTP method for the preceding resource template might be:

  ```
  select empno, ename, dept from emp where empno = :id
  ```

- **HTTP operation**: HTTP (HyperText Transport Protocol) defines the standard methods that are supported in Oracle REST Data Services for building RESTful services: `GET` (retrieve the resource contents), `POST` (store a new resource), `PUT` (update an existing resource), and `DELETE` (remove a resource).

> **See Also:**
>
> - Structure of RESTful Web Services Video
> - Getting Started with RESTful Services

4.3 About the Overview Page

The Overview page provides an overview of the activity in the REST pages.

To navigate to the Overview page, do either of the following:

- In the Launchpad page, select the **Development** tab and click **REST**.
- Click **Selector** to display the navigation menu. Under Development, select **REST**.

You can navigate to pages within REST by using the menu in the header: Overview, Modules, AutoREST and Security.

Use **Export Schema** to copy or download all the roles, privileges, modules, REST-enabled objects, and OAuth clients defined in the schema.

The Overview page consists of three sections.

Objects

Displays the total number of resource modules, schema objects with REST access, roles, privileges, and OAuth clients created. Click the respective card to navigate to the corresponding pages for modules, roles, privileges and OAuth clients.

In the REST pages, the term "object" is used to refer to a module, template, handler, role, privilege, or OAuth Client.

Security

Metadata Catalog: Displays whether access to the ORDS metadata catalog of the schema is protected. The metadata catalog lists all the services in the schema. The user authorization for the metadata catalog is provided at the time of enabling user access.

Modules: Displays the number of modules that are published out of the total modules created.

Module Security: Displays the number of modules that are protected by a privilege out of the total modules created. A partially secured privilege is where the module is protected both by a privilege and a pattern.

Recent Objects

Displays recently updated or created objects. Each entry has a context menu available at the end of the row. Click **Actions** ⋮ to access options such as Edit and Delete.

> **See Also:**
>
> Accessing Database Actions

4.4 About the AutoREST Page

Use the Automatic Enabling of Schema Objects for REST Access (AutoREST) feature to enable REST access for database objects in the current schema.

Enabling REST access for a schema object such as table, view, function, procedure and package allows it to be accessed through RESTful services. See AutoREST in *Oracle REST Data Services Developer's Guide*

To navigate to the AutoREST page, in the REST Overview page, click **AutoREST** or select **AutoREST** from the menu in the header.

The AutoREST page consists of three parts:

- **Objects**: Displays the number of REST-enabled schema objects that are protected and require authorization out of the total number of schema objects enabled in the schema.
- **AutoREST**: Displays the number of tables and views, packages, procedures and functions that are REST enabled in the schema. Click a card to see the corresponding objects listed below.
- The bottom part of the page lists schema objects based on the selection made in AutoREST. You can view these objects in card view or grid view. The actions available in the context menu for an object are:

- **Edit**: Edit the Object Alias or Require Authentication fields for the REST enabled object. For a description of the fields, see Enabling REST Access for a Database Object
- **Disable**: Disable REST access for the object. See Disabling REST Access for a Database Object
- **Get Curl**: Generate cURL calls for a REST enabled object. See Generating cURL Requests for a REST-Enabled Database Object
- **Export OpenAPI**: Export the object as JSON Open API code.
- **OpenAPI View**: Display the object as a Swagger UI implementation. You can view and execute the handlers, pass parameters to the handlers, and copy or download the responses.

See Also:

- About the REST Search Toolbar
- About the Overview Page

4.4.1 Enabling REST Access for a Database Object

You can enable REST access for a table, view, materialized view, function, procedure or package.

To REST enable a schema object:

1. In the Navigator tab in the SQL page, right-click an object and select **REST**, then select **Enable**.
2. Enter the following fields:
   - **Object Alias**: Enter another name for the object. For security purposes, it is recommended to use a different name than the actual object name.
   - **Require Authentication**: Select this option to secure the REST endpoint.
   - **Authorization Role** and **Authorization Privilege**: These fields are automatically pre-filled.

   Show code: Select this option to view the PL/SQL code equivalent of the Enable REST Access slider. You can copy and execute this PL/SQL code in the worksheet to perform the same action that occurs when you click **Enable** in the Enable REST Access slider.
3. Click **Enable**.

   A REST-enabled object is indicated by a **REST Enabled** icon in the Navigator tab.

4.4.2 Disabling REST Access for a Database Object

This section describes how to disable REST access for a database object.

1. In the SQL page, in the Navigator tab, right-click the database object that is REST enabled and select **Disable**.
2. You see a prompt to confirm. Click **OK**.

4.4.3 Generating cURL Requests for a REST-Enabled Database Object

For a REST-enabled database object, you can generate sample cURL calls for GET, POST, PUT and DELETE requests for the selected object. Use the Copy to Clipboard icon to copy the cURL code.

To view the cURL call for a specific method for the selected object:

1. In the SQL page, in the Navigator tab, right-click the REST-enabled object, select **REST** and then select **cURL command**.
2. Select the HTTP method from the left pane, enter values if required and click **Next** to see the cURL command.

   For more information about each HTTP method, see Examples in Oracle REST Data Services.

4.5 About the REST Search Toolbar

This section describes the various options available on the search toolbar in the REST pages. The options selected are saved for your session.

Search: Performs a search based on the text entered in the field. This field is not case sensitive. When you start typing in the Search field, the icon appears. If you click the icon, the text that you typed is deleted and you can re-enter text again.

Filter by: Performs a search based on the filters selected. In the Filter by drop down, first select the column to apply the filter on, and then select the values to search for in that column.

Sort By: Sorts the data in ascending or descending based on the columns selected. The columns can be turned on or off. Reset restores the columns to the original state.The sort criteria selected is shown in the upper right corner as Sort By.

Page Size. First Page, Previous page, Next Page: Only displayed in the Card View. Enables you to select page size, and move between pages.

Reload: Refreshes the information on the page.

There are two views available:

Open Card View: This is the default display view. Information is displayed for each object in a card format. Each card displays associated attributes for the object along with the context menu, which list the actions available for the object.

Open Grid View: Displays the information for each object in a single row of a table. The last column in each row contains the context menu icon, which shows the actions available for the object.

4.6 Creating RESTful Web Services

Create RESTful web services using the Modules, Templates and Handlers pages.

To create a RESTful web service, you need to:

- Define a resource module
- Define a resource template
- Define one or more resource handlers such as GET, PUT, POST or DELETE. Optionally, define parameters that you need to pass to the resource handler.

The following sections provide information on how to create resource modules, resource templates and resource handlers.

- Managing Resource Modules
- Managing Resource Templates
- Managing Resource Handlers
- Example: Inserting a Record using a POST Handler
- Viewing Resource Handler Details and Managing Parameters

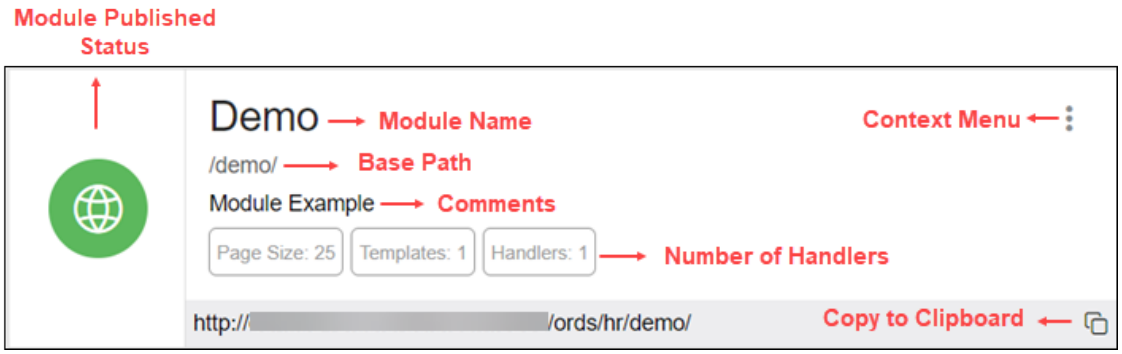
> **See Also:**
>
> - REST Workshop Overview Video
> - Creating a RESTful Service in *Oracle REST Data Services Quick Start Guide*

4.6.1 Managing Resource Modules

You can define, edit and delete resource modules in the Modules page.

To navigate to the Modules page, in the REST Overview page, click **Modules** in Objects, or from the menu in the header, select **Modules**.

The module attributes displayed by default in card view are shown in the following figure.



The module status can be published or unpublished. If the status is unpublished, the icon is displayed in a red color.

Click the module name in card view to go to the Templates page for that module. You can also navigate to the Templates page through the context menu.

The actions available in the context menu are:

- Creating a Resource Module

- Importing a Resource Module
- Editing a Resource Module
- Deleting a Resource Module
- Publishing/Unpublishing a Resource Module
- Exporting a Resource Module
- Viewing the Module in OpenAPI View
- Navigating to the Templates page for a module

> **See Also:**
>
> About the REST Search Toolbar

4.6.1.1 Creating a Resource Module

This section describes how to create a resource module.

1. In the Modules page, click **Create Module**.
2. Enter the following fields. The fields with an asterisk (*) are mandatory:

   Module Definition tab

   - **Module Name**: Enter the name of the module. This field is case sensitive.
   - **Base Path**: Enter the base of the URI for accessing the RESTful service. For example: hr/ means that all URIs starting with hr/ will be serviced by this resource module. Note the change in the **Preview URL** as you enter the base path.
   - **Is Published**: Enable to make the RESTful service available for use.
   - **Pagination Size**: Enter or select the number of results to return on each page based on a database query. The default value is 25.
   - **Protected By Privilege**: Select a privilege to protect the module from the drop-down list.
   - **Comments**: Enter descriptive comments.
   - **Go to Module after creation**: Select this option to go to the Templates page after the module is created.

   Origins Allowed tab

   - Enter origins that are allowed to access the resource templates. Click **Add** (+) to add each origin. For example:

     http://example1.org

     https://*.example2.com

   Show code: Select this option to view the PL/SQL code equivalent of the Create Module slider. You can copy and execute this PL/SQL code in the worksheet to perform the same action that occurs when you click **Create** in the Create Module slider.
3. Click **Create**.

   The new module is displayed on the Modules page. If you cannot see it, locate the module using the Search field.

4.6.1.2 Importing a Resource Module

This section describes how to import a resource module.

> **Note:**
>
> You can import an OpenAPI version 2.0 or 3.0 document in JSON format only.

1. In the Modules page, at the top right, click **Import Module**.
2. Browse and select the JSON file to import. Click **Open**.

   The Import Module slider is displayed with **Base Path** and **Preview URL** fields prefilled. Edit the Base Path as needed. Editing the base path field will also update the base path in the JSON format that is displayed in the slider.
3. Click **Import**.

   When the module is imported, a privilege configuration is created and set to protect the module. The name of the privilege is generated by using the base path and adding "_priv" at the end of the base path string and also replacing every slash "/" in the base path to underscore "_".
4. Your module is ready for editing.

   After your module is imported, the templates and handlers are created, but the module is not published. You need to supply SQL and PL/SQL blocks for each handler, and publish your module when it is ready for testing or production.

4.6.1.3 Editing a Resource Module

This section describes how to edit a resource module.

1. In the Modules page, for the specific module, click ⋮ and select **Edit**.
2. Edit the required fields and click **Save**. For a description of the fields, see Creating a Resource Module.

4.6.1.4 Deleting a Resource Module

This section describes how to delete a resource module.

1. In the Modules page, for the specific module, click ⋮ and select **Delete**.

   A prompt appears asking you to confirm.
2. Click **OK** to delete.

4.6.1.5 Publishing/Unpublishing a Resource Module

This section describes how to make a resource module available or not.

1. In the Modules page, for the specific module, click ⋮ and then select **Publish** or **Unpublish**.

   You see a prompt to confirm.

2. Click **OK**.

4.6.1.6 Exporting a Resource Module

You can export the PL/SQL source code for the API or Open API (Swagger) JSON code for a resource module or RESTful service. If the module has templates, handlers, parameters, roles, and so on, these objects are also displayed and exported.

1. In the Modules page, for a specific module, click ⋮ and select **Export Module**.
2. Select **PL/SQL** to copy or export the PL/SQL source for the API. Select **OpenAPI** to copy or export the Open API (Swagger) JSON code for the module.

> **Note:**
>
> A module exported using Open API is always imported with an unpublished status.

3. Click **Copy to Clipboard** to copy the code or click **Download** to download to your local computer.

4.6.1.7 Viewing the Module in OpenAPI View

This section describes how to open the module in Open API view.

The Open API view option is available for modules that are published and not protected.

1. In the Modules page, for the specific module, click ⋮ and select **OpenAPI View**.
2. The module is displayed as a Swagger UI implementation.

   You can view and execute the handlers, pass parameters to the handlers, and copy or download the responses.
   To learn more about Swagger UI implementation, refer to the official Swagger UI website.

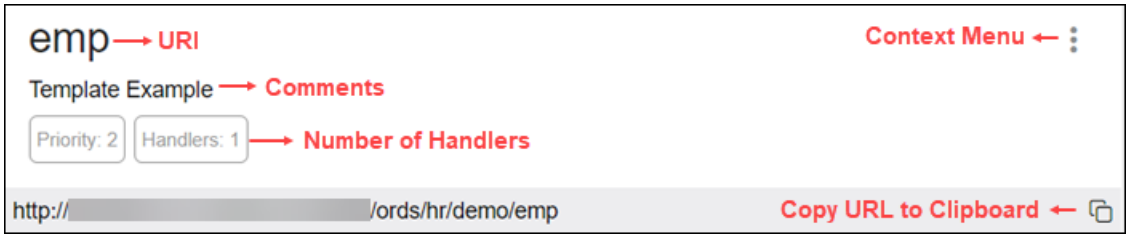
4.6.2 Managing Resource Templates

You can define, edit and delete resource templates for a module in the Templates page.

To navigate to this page, in the Modules page, click ⋮ for a module and select **Templates**, or click the name of the module in card view.

At the top of the page, the module card is available with context menu options such as Edit, Delete, Publish and Export.

The templates attributes displayed by default in card view are shown in the following figure.



Click the template name in card view to go to the Handlers page. You can also navigate to the Handlers page through the context menu.

The actions available in the context menu are:

- Creating a Resource Template
- Editing a Resource Template
- Deleting a Resource Template
- Navigating to the Handlers page for the template

4.6.2.1 Creating a Resource Template

This section describes how to create a resource template.

1. In the Templates page for a module, click **Create Template**.
2. Enter the following fields. The fields with an asterisk (*) indicate that they are mandatory:
   - **URI Template**: Enter the URI pattern for the resource template.

     For example, a pattern of /objects/:object/:id? will match /objects/emp/101 (matches a request for the item in the emp resource with id of 101) and will also match / objects/emp/ (matches a request for the emp resource, because the :id parameter is annotated with the ? modifier, which indicates that the id parameter is optional).

     Note that the **Preview URL** changes as you type the URI template.
   - **Priority**: Enter or choose the priority for how the resource template should be evaluated (1 is low priority and 9 is high priority).
   - **HTTP Entity Tag Type**: Select the type of entity tag to be used by the resource template. An entity tag is an HTTP Header that acts as a version identifier for a resource.

     Options include:

     – **Secure HASH (default)**: The contents of the returned resource representation are hashed using a secure digest function to provide a unique fingerprint for a given resource version.

     – **Query**: Define a query manually that uniquely identifies a resource version. A manually defined query can often generate an entity tag more efficiently than hashing the entire resource representation.

     – **None**: Do not generate an entity tag.
   - **Comments**: Enter descriptive comments if any.
   - **Go to Handlers after creation**: Select this option to go to the Handlers page after the template is created.

   Show code: Select this option to view the PL/SQL code equivalent of the Create Template slider. You can copy and execute this PL/SQL code in the worksheet to perform the same action that occurs when you click **Create** in the Create Template slider.
3. Click **Create**.

   The new template for the module is added to the Templates page.

   If "Go to Handlers after creation" is selected, you can use the module name in the breadcrumbs at the top of the page to see the list of templates for the module.

4.6.2.2 Editing a Resource Template

This section describes how to edit a resource template.

1. In the Templates page, for a specific template, click ⋮ and then select **Edit**.
2. Edit the required fields and click **Save**.

   For a description of the fields, see Creating a Resource Template.

4.6.2.3 Deleting a Resource Template

This section describes how to delete a resource template.

1. In the Templates page, for a specifc template, click ⋮ and then select **Delete**.

   You are prompted to confirm.
2. Click **OK** to delete.

4.6.3 Managing Resource Handlers

You can create, edit and delete resource handlers for a template in the Handlers page.

To navigate to this page, in the Templates page, click ⋮ for a template and select **Handlers**, or click the name of the template in card view.

At the top of the page, the template card is available with context menu options such as Edit and Delete.

Tthe handler attributes displayed by default in card view are shown in the following figure.

The HTTP Method can be one of the following: GET, PUT, POST and DELETE.

Click the handler name to go to the HTTP method page or you can also navigate to the HTTP method page using the context menu.

The actions available in the context menu are:

- Creating a Resource Handler

> **Note:**
>
> This option is not available if the template has all four handlers created (GET, POST, PUT, DELETE).

- Editing a Resource Handler
- Deleting a Resource Handler
- Navigating to the Handler *HTTP method* page

4.6.3.1 Creating a Resource Handler

This section describes how to create a resource handler.

1. In the Handlers page, click **Create Handler**.
2. Specify the properties of the resource handler. The specific options depend on the method type.

   Handler Definition Tab

   - **Method**: Enter the HTTP request method for this handler: GET (retrieves a representation of a resource), POST (creates a new resource or adds a resource to a collection), PUT (updates an existing resource), or DELETE (deletes an existing resource). Only one handler for each HTTP method is permitted.
   - **Items Per Page**: Enter or choose the default pagination size, or the number of rows to return for a database query. This option is only available for GET handlers. If this value is not defined, the number of items per page is the one defined in the module.
   - **Source Type**: Select the source implementation for the selected HTTP method. The default is collection query.
     - **Collection Query**: Executes a SQL query and transforms the result set into a **JSON** representation. Available when the HTTP method is GET.
     - **Collection Item**: Executes a SQL query returning one row of data into a **JSON** representation. Available when the HTTP method is GET.
     - **Media**: Executes a SQL Query conforming to a specific format and turns the result set into a **binary** representation with an accompanying HTTP Content-Type header identifying the Internet media type of the representation.
     - **PL/SQL**: Executes an anonymous PL/SQL block and transforms any OUT or IN/OUT parameters into a **JSON** representation.
   - **Source**: Enter the SQL query or the PL/SQL block for the HTTP method. In the handler details page, you can test the SQL or PL/SQL handler code in the handler editor.
   - **Go to Handler after creation**: Select this option to go to the details page after the handler is created.

   MIMEs Allowed tab

   Identifies the type of information included in the HTTP Request Body. For example, if the POST Handler is expecting JSON in the request, add "Application/JSON" as the MIME type.

   Show code: Select this option to view the PL/SQL code equivalent of the Create Handler slider. You can copy and execute this PL/SQL code in the worksheet to perform the same action that occurs when you click **Create** in the Create Handler slider.

3. Click **Create**.

   The handler is displayed on the Handlers page for the specific template.

   You can test the REST service endpoint using a REST client or a command-line tool such as cURL.

> **See Also:**
>
> About cURL and Testing RESTful Services in *Oracle REST Data Services Developer's Guide*.

4.6.3.2 Editing a Resource Handler

This section describes how to edit a resource handler.

1. In the Handlers page, for a specific handler, click ⋮ and then click **Details**.
2. Edit the required fields and click **Save**.

   For a description of the fields, see Creating a Resource Handler.

4.6.3.3 Deleting a Resource Handler

This section describes how to delete a resource handler.

1. In the Handlers page, for the specific handler, click ⋮ and then click **Delete**.

   You are prompted to confirm.
2. Click **OK** to delete.

4.6.4 Example: Inserting a Record using a POST Handler

The following example illustrates how to insert a record in the DEPT table.

Prerequisites

- Using the worksheet in the SQL page, create a **DEPT** table.

```
CREATE TABLE DEPT(
  DEPTNO     number(2,0),
  DNAME      varchar2(14),
  LOC        varchar2(13),
  CONSTRAINT PK_DEPT PRIMARY KEY (DEPTNO)
)
```

- Create a module named **example**. See Creating a Resource Module
- Create a template named **emp/** for the module. See Creating a Resource Template

To insert a record:

1. In the Handlers page, click **Create Handler**.
2. Enter the following details:
   - In the **Method** field, select **POST**.
   - In the **Source Type** field, select **PL/SQL**.

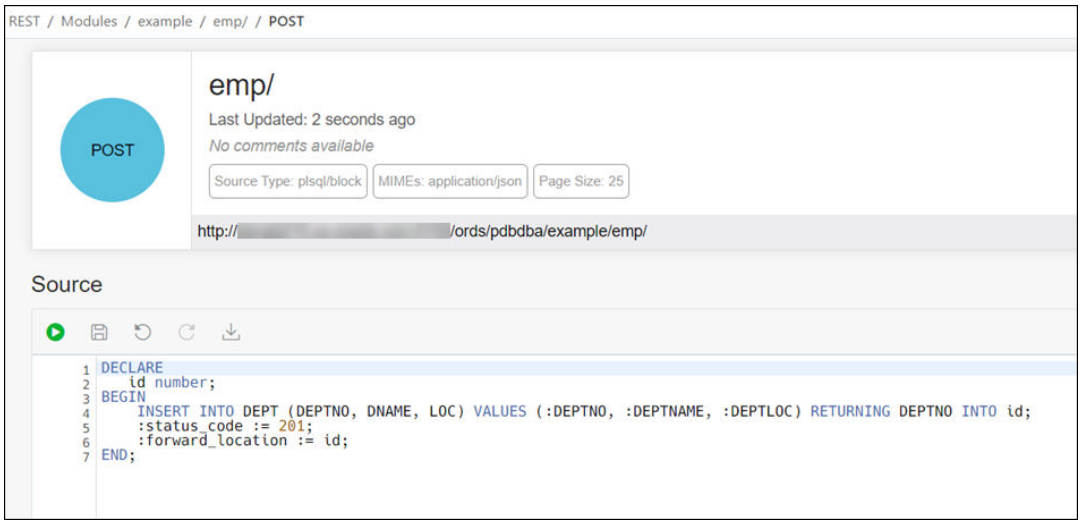
- In the **Source** field, enter the following PL/SQL block:

```
DECLARE
   id number;
BEGIN
    INSERT INTO DEPT (DEPTNO, DNAME, LOC) VALUES
(:DEPTNO, :DEPTNAME, :DEPTLOC) RETURNING DEPTNO INTO id;
    :status_code := 201;
    :forward_location := id;
END;
```

  where

  - `RETURNING DEPTNO INTO id` returns the value assigned to `DEPTNO` into the variable `id`.
  - `:status_code` is an implicit parameter that is assigned 201 to indicate that the resource is created.
  - `:forward_location` is an implicit parameter that is assigned the `id` that contains the value of `DEPTNO`. This parameter specifies the location to forward a GET request to produce the response for the POST request.
- Ensure **Go to Handler page after creation** is selected.
- In the **MIMEs Allowed** Tab, select **application/json** from the drop-down list and click **Add** +.
- Click **Create**.

The Handler POST page appears.



You can test the source code by clicking **Execute** in the Source editor.

3. Create a GET resource handler to produce a response for the POST request.

   In the Example module, create a template named **emp/:id**.

Create a GET resource handler where **Source Type** is **Collection Item** and **Source** contains the following query:

```
SELECT * FROM DEPT WHERE DEPTNO = :id
```

4. Test the RESTful service endpoint using the following command in cURL. You can copy the URL (http://xyz.us.comp.com:1234/ords/pdbdba/example/emp) using the Copy to Clipboard icon in the POST handler card at the top.

   ```
   curl --location --request POST --header "Content-Type: application/json"
   --data '{"DEPTNO": 54, "DEPTNAME": "HR", "DEPTLOC": "America" }' 'http://
   xyz.us.comp.com:1234/ords/pdbdba/example/emp/'
   ```

   The output is:

   ```
   {"deptno":54,"dname":"HR","loc":"America","links":[{"rel":"collection",
   "href":"xyz.us.comp.com:1234/ords/pdbdba/example/emp/"}]}
   ```

5. In the SQL page, check the DEPT table to see if the new record has been inserted by using the following statement:

   ```
   SELECT * FROM DEPT;
   ```

4.6.5 Viewing Resource Handler Details and Managing Parameters

You can view resource handler details and manage parameters in the *HTTP method* page.

To navigate to this page, for a specific handler, click ⋮ and then select **Details** or click the name of the handler in card view.

At the top of the page, the handler card is available with context menu options such as Edit and Delete.

The Handler *HTTP method* page has two major sections:

- A code editor for executing the SQL or PL/SQL source code that was defined in the resource handler. You can execute the code, save, undo, redo, download and clear output. For more information about the editor, see Executing SQL Statements in the Code Editor.

  Implicit parameters used in REST service handlers are displayed in a scrolling list towards the right side of the editor. See Implicit Parameters

- A section for managing parameters required for running the HTTP method.

By default, the parameters are displayed in grid view. The attributes displayed on a parameter in card view are shown in the following figure.

The actions available in the context menu are:

- Creating a Parameter
- Editing a Parameter
- Deleting a Parameter

> **See Also:**
>
> About the REST Search Toolbar

4.6.5.1 Creating a Parameter

This section describes how to create a parameter for a resource handler.

1. In the handler *HTTP method* page, click **Create Parameter**.
2. Enter the following fields. The fields with an asterisk (*) are mandatory:
   - **Parameter Name**: Enter the name of the parameter, as it is named in the URI Template or HTTP Header. The name defines how the parameter is identified in the incoming request or how the parameter is named in the Response.
   - **Bind Variable Name**: Enter the name of the parameter, as it will be referred to in SQL. If NULL is specified, then the parameter is unbound.
   - **Source Type**: Select the type that is identified if the parameter originates in the URI Template or HTTP Header.
   - **Parameter Type**: Select the native type of the parameter.
   - **Access Method**: Select the parameter access method. Indicates if the parameter is an input value, output value, or both.

   Show code: Select this option to view the PL/SQL code equivalent of the Create Parameter slider. You can copy and execute this PL/SQL code in the worksheet to perform the same action that occurs when you click **Create** in the Create Parameter slider.

3. Click **Create**.

   The parameter is added in the Parameters section.

4.6.5.2 Editing a Parameter

This section describes how to edit a parameter.

1. In the Parameters section, for the specific parameter, click ⋮ and then select **Edit**.
2. Edit the required fields and click **Save**.

   For a description of the fields, see Creating a Parameter.

4.6.5.3 Deleting a Parameter

This section describes how to delete a parameter.

1. In the Parameters section, for the specific parameter, click ⋮ and then select **Delete**.

   You are prompted to confirm.
2. Click **Yes** to delete.

4.6.5.4 Implicit Parameters

This section describes the implicit parameters in the Resource Handler Details page. These parameters are automatically added to the resource handlers.

Table 4-1 Implicit Parameters

Name	Type	Access Mode	HTTP Header	Description
`:body`	BLOB	IN	N/A	Specifies the body of the request as a temporary BLOB.
`:body_text`	CLOB	IN	N/A	Specifies the body of the request as a temporary CLOB.

Table 4-1 (Cont.) Implicit Parameters

Name	Type	Access Mode	HTTP Header	Description
:content_type	VARCHAR	IN	Content-Type	Specifies the MIME type of the request body, as indicated by the Content-Type request header.
:current_user	VARCHAR	IN	N/A	Specifies the authenticated user for the request. If no user is authenticated, then the value is set to null.
:forward_location	VARCHAR	OUT	X-ORDS-FORWARD-LOCATION	Specifies the location where Oracle REST Data Services must forward a GET request to produce the response for this request.
:fetch_offset	NUMBER	IN	N/A	Specifies the zero-based offset of the first row to be displayed on a page.

Table 4-1 (Cont.) Implicit Parameters

Name	Type	Access Mode	HTTP Header	Description
`:fetch_size`	NUMBER	IN	N/A	Specifies the maximum number of rows to be retrieved on a page.
`:page_offset`	NUMBER	IN	N/A	Specifies the zero based page offset in a paginated request. **Note:** The `:page_offset` parameter is deprecated. Use `:row_offset` parameter instead.
`:page_size`	NUMBER	IN	N/A	Specifies the maximum number of rows to be retrieved on a page. The `:page_size` parameter is deprecated. Use `:fetch_size parameter` instead.

Table 4-1 (Cont.) Implicit Parameters

Name	Type	Access Mode	HTTP Header	Description
`:row_offset`	NUMBER	IN	N/A	Specifies the one-based index of the first row to be displayed in a paginated request.
`:row_count`	NUMBER	IN	N/A	Specifies the one-based index of the last row to be displayed in a paginated request.
`:status_code`	NUMBER	OUT	`X-ORDS-STATUS-CODE`	Specifies the HTTP status code for the request.

For more information about the implicit parameters, see *Oracle REST Data Services Developer's Guide*.

4.7 Securing RESTful Web Services

Define roles, privileges and OAuth Clients to ensure authentication and authorization are required for accessing RESTful web services.

To protect a RESTful web service, you need to:

- Create a role
- Create a privilege selecting the role and modules or resources to protect

To enable access to a protected RESTful service using the OAUTH2 Workflow, create an OAuth client using the role and privilege created for protecting the RESTful service.

The following sections provide information on how to create roles, privileges and OAuth clients:

- Managing Roles
- Managing Privileges
- Managing OAuth Clients
- Examples

> **See Also:**
>
> - REST Workshop Overview Video
> - Tutorial for Protecting and Accessing Resources

4.7.1 Managing Roles

You can create, edit and delete roles for RESTful services in the Roles page.

To navigate to the Roles page, from the REST Overview page, click **Roles** in Objects, or from the menu in the header, select **Security** and then select **Roles**.

The actions available in the context menu are:

- Creating a Role
- Editing a Role
- Deleting a Role
- Viewing Assigned Privileges

> **See Also:**
>
> About Users and Roles in *Oracle REST Data Services Developer's Guide*.

4.7.1.1 Creating a Role

Create a role with a specific name. After the role is created, you can associate it with a privilege.

1. In the Roles page, click **Create Role**.
2. In the Role Name field, enter the name of the role to create.

   Show code: Select this option to view the PL/SQL code equivalent of the Create Role slider. You can copy and execute this PL/SQL code in the worksheet to perform the same action that occurs when you click **Create** in the Create Role slider.
3. Click **Create**.

   The new assigned role is displayed on the Roles page.

4.7.1.2 Editing a Role

This section describes how to edit a role.

1. In the Roles page, for the specific role, click **Actions** ⋮ and select **Edit**.
2. Enter the changes required and click **Save**.

   For a description of the fields, see Creating a Role.

4.7.1.3 Deleting a Role

This section describes how to delete a role.

1. In the Roles page, for the specific role, click **Actions** ⋮ and select **Delete**.

   You are prompted to confirm.
2. Click **Yes** to delete.

4.7.1.4 Viewing Assigned Privileges

This section describes how to view the privileges associated with a role.

In the Roles page, for the specific role, click **Actions** ⋮ and select **Details**. The privilges assigned to the role are displayed.

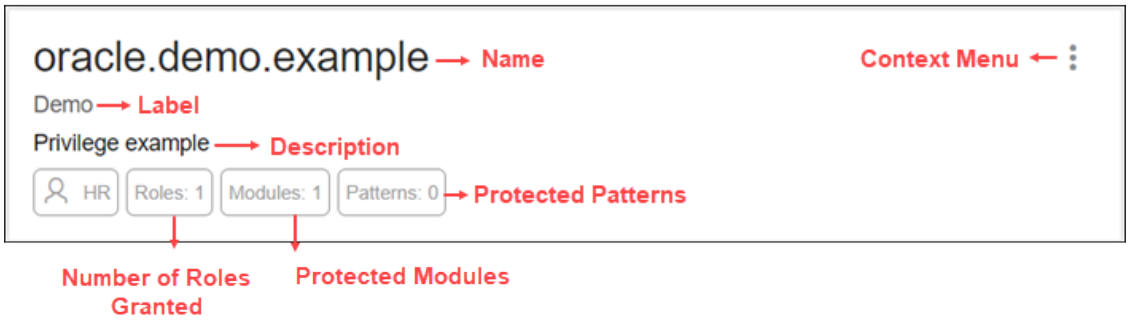
4.7.2 Managing Privileges

You can create, edit and delete privileges for RESTful services in the Privileges page.

A privilege defines the set of roles, at least one of which an authenticated user must possess to access a RESTful service protected by a privilege.

To navigate to the Privileges page, from the REST Overview page, click **Privileges** in Objects, or from the menu in the header, select **Security** and then select **Privileges**.

The privilege attributes displayed by default in card view are shown in the following figure.



The actions available in the context menu are:

- Creating a Privilege
- Editing a Privilege
- Deleting a Privilege

4.7.2.1 Creating a Privilege

This section describes how to create a privilege.

1. In the Privileges page, click **Create Privilege**.
2. Enter the following fields. The fields with an asterisk (*) are mandatory:

- **Label**: Enter the name of the privilege in a way that it is easy to understand.
- **Name**: Enter a unique name for the privilege.
- **Description**: Enter a brief description of the purpose of the privilege.
- **Comments**: Enter comments.
- **Roles**: Enter one or more roles assigned to the privilege.
- **Protected Modules**: Select the modules to protect.
- **Protected Resources**: Instead of Protected Modules, use the Protect Resources tab to apply security based on a URI pattern.

Show code: Select this option to view the PL/SQL code equivalent of the Create Privilege slider. You can copy and execute this PL/SQL code in the worksheet to perform the same action that occurs when you click **Create** in the Create Privilege slider.

3. Click **Create**.

   The new privilege is displayed on the Privileges page.

4.7.2.2 Editing a Privilege

This section describes how to edit a privilege.

1. In the Privileges page, for the specific privilege, click **Actions** ⋮ and select **Edit**.
2. Make the required changes and click **Save**.

   For a description of the fields, see Creating a Privilege.

4.7.2.3 Deleting a Privilege

This section describes how to delete a privilege.

1. In the Privileges page, for the specific privilege, click **Actions** ⋮ and select **Delete**.
2. You are prompted to confirm. Click **Yes**.

4.7.3 Managing OAuth Clients

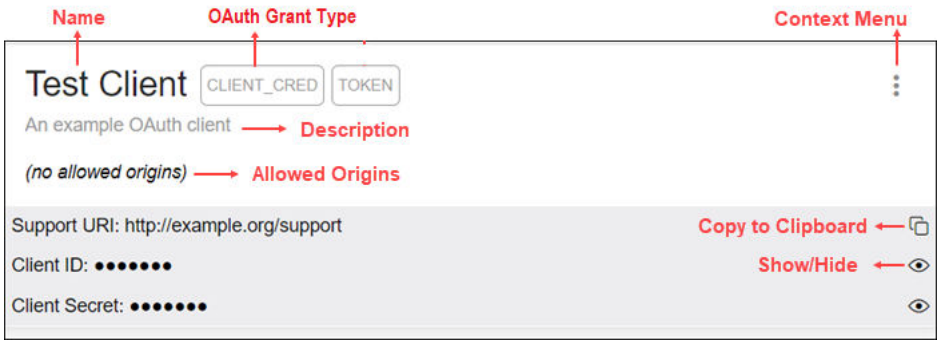
Using OAuth 2.0-based authentication, you can ensure that your RESTful web services are accessed only by specific users or clients.

OAuth 2.0 is a standard Internet protocol that defines flows to provide conditional and limited access to a RESTful API. For more information, see OAuth-Based Authentication .

You can create, edit and delete OAuth Clients in the OAuth Clients page.

To navigate to the OAuth Clients page, from the REST Overview page, click **Clients** in Objects, or from the menu in the header, select **Security** and then select **OAuth Client**.

The OAuth Client attributes displayed by default in card view are shown in the following figure.



The actions available in the context menu are:

- Creating an OAuth Client
- Editing an OAuth Client
- Deleting an OAuth Client
- Exporting an OAuth Client
- **Get Bearer Token**: Provides the access token to call the RESTful service for Client Credentials and Implicit OAuth grant types. See Creating an OAuth Client Using the Client Credentials Grant Type
- **Auth Details**: Displays the Unique Value and Authorization URI for the Auth Code OAuth grant type. See Creating an OAuth Client Using the Auth Code Grant Type

> **See Also:**
>
> Tutorial for Protectinng and Accessing Resources

4.7.3.1 Creating an OAuth Client

Creates the OAuth Client and grants the required roles and privileges.

1. In the OAuth Clients page, select **Create OAuth Client**.
2. Enter the following fields. The fields with an asterisk (*) are mandatory:

   Client Definition tab

   - **Grant type**: Select the authorization grant type. The options are Client_Cred, Auth Code, or Implicit.
     For more information about these grant types, see OAuth Flows in *Oracle REST Data Services Developer's Guide*.
   - **Name**: Name of the client.
   - **Description**: Description of the purpose of the client.
   - **Redirect URI**: Enter the client-controlled URI to which redirect containing an OAuth access token or error will be sent.
   - **Support URI**: Enter the URI where end users can contact the client for support. Example: http:// www.myclientdomain.com/support/
   - **Support Email**: Enter the email where end users can contact the client for support.

- **Logo**: Upload your logo in JPG, BMP or SVG file format. Ensure that the logo is less than 100 KB in size.
- **Roles**: Select roles to be granted.
- **Allowed Origins**: Add the list of URL prefixes.
- **Privileges**: Select privileges that the client wants to access.

Show code: Select this option to view the PL/SQL code equivalent of the Create OAuth Client panel. You can copy and execute this PL/SQL code in the worksheet to perform the same action that occurs when you click **Create** in the Create OAuth Client panel.

3. Click **Create**.

   The OAuth Client registered is displayed on the OAuth Clients page.

   The Client ID and Client Secret values represent the secret credentials for the OAuth client. Click **Show/Hide** to see the values.

   Test the secured REST service endpoint using a REST client or the cURL command line tool.

> **See Also:**
>
> Tutorial for Protecting and Accessing Resources in *Oracle REST Data Services Developer's Guide*

4.7.3.2 Editing an OAuth Client

This section describes how to edit an OAuth Client.

1. In the OAuth Clients page, for the specific client, click **Actions** and select **Edit**.
2. Edit the required fields and click **Save**.

   For a description of the fields, see Creating an OAuth Client.

4.7.3.3 Deleting an OAuth Client

This section describes how to delete an OAuth Client.

1. In the OAuth Clients page, for the specific client, click **Actions** and select **Delete**.
2. You are prompted to confirm. Click **Yes**.

4.7.3.4 Exporting an OAuth Client

This section describes how to export an OAuth Client.

1. In the OAuth Clients page, for the specific client, click **Actions** and select **Export**.
2. In the OAuth Client panel, click the **Copy** icon or **Download** to copy or download the OAuth Client information.

4.7.4 Examples

This section provides a few examples on creating an OAuth client with different grant types.

Topics:

- Creating an OAuth Client Using the Client Credentials Grant Type
- Creating an OAuth Client Using the Auth Code Grant Type

4.7.4.1 Creating an OAuth Client Using the Client Credentials Grant Type

This section describes how to create an OAuth Client using the Client Credential grant type.

Create an OAuth Client for the created module "example" in Example: Inserting a Record using a POST Handler. The endpoint for the RESTful service is http://xyz.us.comp.com:1234/ords/pdbdba/example/emp/.

Prerequisites

Create a role named HR Admin. See Creating a Role

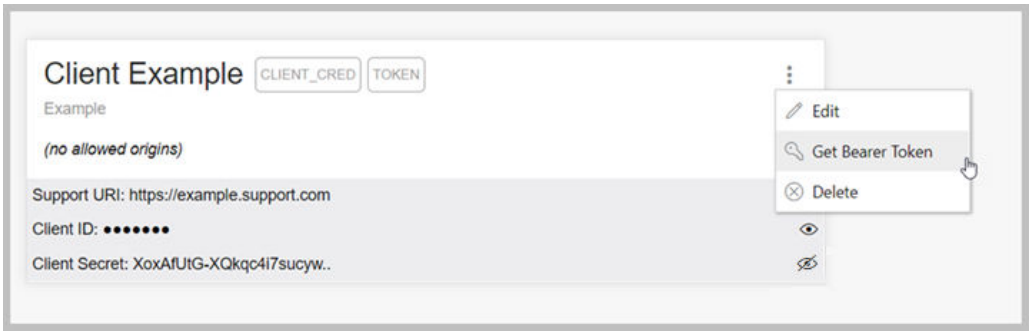
Create a privilege named Example.HR. See Creating a Privilege

1. In the OAuth Clients page, click **Create OAuth Client**.
2. Enter the following fields:
   - In the Grant type field, select **CLIENT_CRED**.
   - Enter Name, Description, Redirect URI, Support URI and Support Email for your OAuth Client.
   - In the Roles tab, add the created role (HR Admin).
   - In the Privileges tab, add the created privilege (Example.HR).
   - Click **Create**.

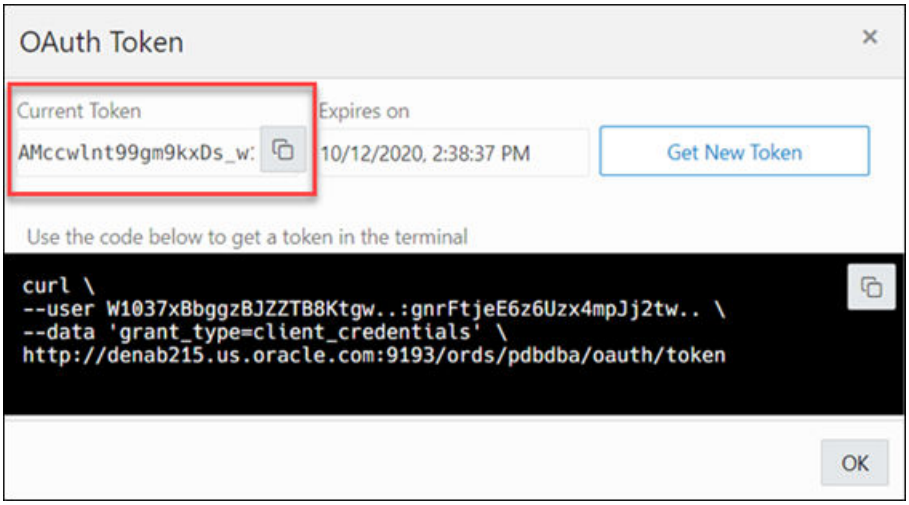
   The new OAuth Client card appears on the OAuth Clients page.
3. Using cURL, test the service endpoint without OAuth credentials.

   ```
   curl --location --request POST --header "Content-Type: application/json"
   --data '{"DEPTNO": 55, "DEPTNAME": "Sales", "DEPTLOC": "Australia" }'
   'http://xyz.us.comp.com:1234/ords/pdbdba/example/emp/'
   ```

   You see an error.
4. Test the endpoint with OAuth credentials:
   a. To get an access token, select **Get Bearer Token** from the context menu in the OAuth Client card.



The OAuth Token dialog appears. Use **Copy to Clipboard** to copy the token.



b. In cURL, use the bearer access token in the following statement to request the resource:

```
curl -H "Content-Type: application/json" -H "Authorization: Bearer
AMccwlnt99gm9kxDs_w1DA" --location --request POST --data
'{"DEPTNO": 55, "DEPTNAME": "Sales", "DEPTLOC": "Australia"}' 'http://
xyz.us.comp.com:1234/ords/pdbdba/example/emp/'
```

The output shows:

```
{"deptno":55,"dname":"Sales","loc":"Australia","links":
[{"rel":"collection",
"href":"http://xyz.us.comp.com:1234/ords/pdbdba/example/emp/"}]}
```

5. In the SQL page, you can verify the new record that has been inserted in the DEPT table by using the following statement:

```
SELECT * FROM DEPT;
```

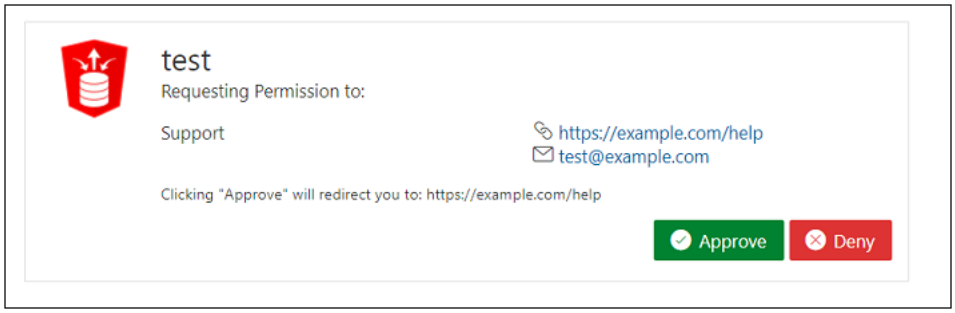
4.7.4.2 Creating an OAuth Client Using the Auth Code Grant Type

This section describes how to create an OAuth Client using the Auth Code grant type.

1. In the OAuth Clients page, click **Create OAuth Client**.
2. Enter the following fields:
   - In the **Grant type** field, select **Auth Code**.
   - Enter **Name**, **Description**, **Redirect URI**, **Support URI** and **Support Email** for your OAuth Client.
   - In the **Roles** tab, add the relevant roles.
   - In the **Privileges** tab, add the relevant privileges.
   - Click **Create**.

   The new OAuth Client card appears on the OAuth Clients page.
3. In the OAuth Client card, click the **Actions** icon and select **Auth Details**.

   The Unique Value and Authorization URI appears.
4. In the Authorization URI field, click the **Open in New Tab** icon. A new window is displayed with an `Unauthorized` error message along with a Sign In link.
5. Click **Sign in** and enter your login credentials again.
6. You are prompted to approve the request to access the protected URI. Click **Approve**.



5
The Liquibase Page

The Liquibase page displays information of all database deployments made in the current schema. A deployment consists of changesets, which is a list of sequential changes to be applied to the database.

To navigate to the Liquibase page, do either of the following:

- In the Launchpad page, select the **Development** tab and click **Liquibase**.
- Click **Selector** to display the navigation menu. Under **Development**, select **Liquibase**.

If you have not made any deployments into the schema and you access the Liquibase page, no data appears.

Topics:

5.1 Generate a Deployment

First, you have to make changes in the current schema such as creating a table, procedure or any database object.

Then, create a folder that will contain the changesets for the deployment. It is a good practice to create a different folder for each deployment. This ensures that you have information of all the deployments, in case you want to roll back to a previous version.

In the following example, Oracle SQLcl is used to deploy the changes.

1. Open the terminal and navigate to the path of the folder that you will use to save the changes made to the database (changelog files).
2. After you navigate to the correct path, log in to the schema you want to capture with the following command:

   ```
   sqlcl <username>/<password>@<host>:<port>/<servicename>
   ```

3. You can do one of the following:
   - Deploy the entire schema with the following command:

     ```
     lb genschema
     ```

     This command creates a controller.xml file that includes all the changesets for the schema.

- Deploy a specific object using the following command:

  ```
  lb genobject -type <object_type> -name <object_name>
  ```

  This command generates the changelogs (which are XML files) that contain the changes made to the database in the current folder.

4. Log in to the schema where you want to add these changes (in this example, XYZ) with the following command:

   ```
   sqlcl xyz/<password>@<host>:<port>/<servicename>
   ```

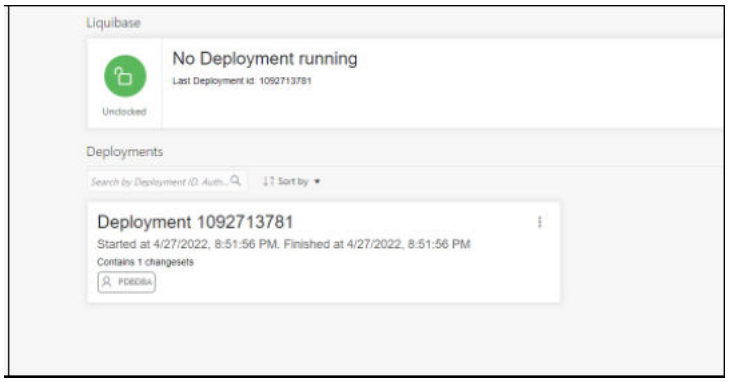
5. Run the following command:

   ```
   lb update -changelog controller.xml (or the file name of your changelog)
   ```

6. Log in to Database Actions on the schema where you deployed the changes and the Liquibase page displays information about the deployment.

5.2 About the Liquibase User Interface

In the Liquibase page, the card at the top displays the Liquibase status. The deployments are displayed as cards below, as shown in the following figure.



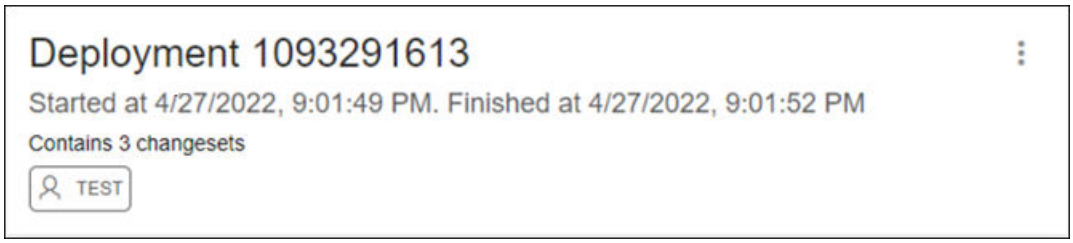
Liquibase Status

Liquibase runs only one deployment at a time. When there is no deployment running, the top Liquibase card indicates that Liquibase is currently unlocked and displays the ID of the last deployment.

If there is a deployment running at the time, the card indicates that Liquibase is locked, and displays the name of the resource blocking it and the time at which the deployment started.

Deployments

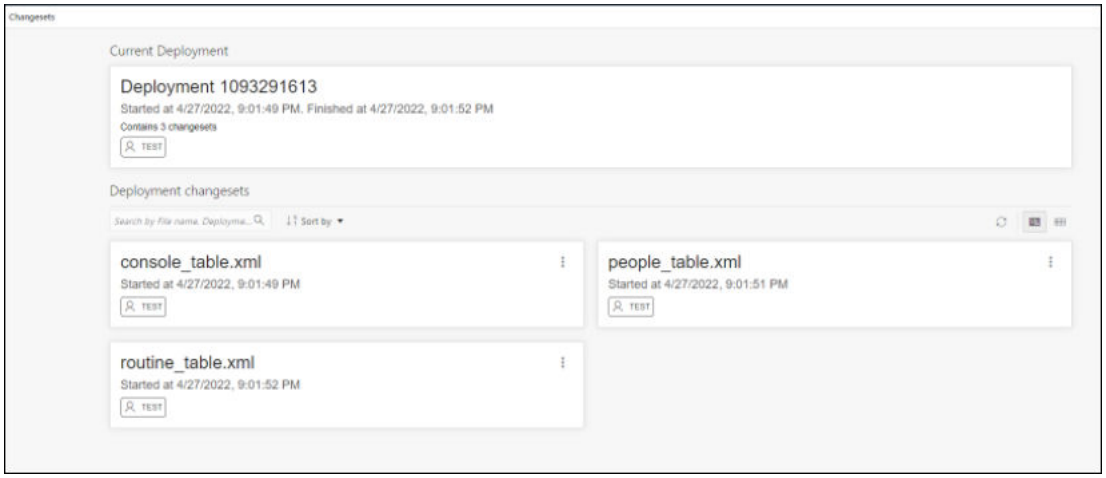
The card for each deployment provides details such as the schema, deployment id, start and complete deployment date and time, and number of changesets run in the deployment. To open a specific deployment page, click the title of the card, or click **Open Deployment Changes** in the context menu.



Changesets

In a deployment page, the changesets that have been run for that specific deployment is displayed below. The Current Deployment card at the top shows the deployment to which the changesets belong.

Each changeset card displays the changeset name, the date and time when the changeset started deploying, and the schema where the change was deployed.



The context menu for a changeset consists of the following options:

- **Previous Object State XML**: Displays an XML showing the state of the object (table, procedure) before the changeset was applied. If the object did not exist previously on the database (example, a table that was created in the deployment), the following message is displayed: `xml code not available....`
- **Show Executed SQL**: Displays the SQL code that was executed when the changeset was deployed.

6
The Charts and Dashboards Page

Charts enable you to create charts from the database. The chart is constructed using the input SQL command. Dashboards enable you to group charts together to create reports.

To navigate to the Charts and Dashboards page, do either of the following:

- In the Launchpad page, select the **Development** tab and click **Charts**.
- Click **Selector** to display the navigation menu. Under Development, select **Charts**.

Topics

6.1 Overview

This section describes the layout of the main Charts and Dashboards page.

To navigate to the Charts and Dashboards page, do either of the following:

The upper right corner has two icons:

- **Tour**: Provides a guided tour of charts and dashboards highlighting salient features and providing information that is useful if you are new to the interface.
- **Create**: Creates a new chart or dashboard. See Creating or Editing a Chart and Creating or Editing a Dashboard

This page consists of two sections:

- **Charts and Dashboards**: Displays the total number of charts and dashboards created. Click the respective card to view the charts or dashboards created in the section below.
- Displays the charts or dashboards created in the default card format.

The actions available for a chart or dashboard are:

- **View Chart/Dashboard**: To view the chart or dashboard on a different page.
- **Edit**: To make changes to the chart or dashboard.
- **Unpublish**: To make the chart or dashboard unavailable for use.
- **Go to module definition**: To show the module definition in the REST Modules page.
- **Delete**: To delete the chart or dashboard.

6.2 Creating or Editing a Chart

You can create or edit a chart to visualize data using SQL commands.

To create a new chart, click **Create Chart**.
To edit a chart, click the context menu icon and select **Edit**.

Enter the following details:

1. **Name:** Enter a name for the chart.
2. **Description:** Enter a description for the chart.
3. **Protected by Privilege:** Select an available privilege from the drop-down list to enable only those with this privilege to securely access the chart.
4. **Published:** When enabled, you can share and access the chart externally.

> **Note:**
>
> You can also protect a chart by protecting the corresponding module using the REST pages. For more information, see Editing a Resource Module. When a chart is protected, a user must authenticate before viewing the chart.

5. Click **Next**.
6. Enter the SQL statement in the command editor on the left. Ensure that you do not end the SQL statement with a semicolon.

   The chart settings are automatically updated in the **Chart Definition** section on the right.
7. Edit/Update the chart settings in the **Chart Definition** section. The fields in the mapping section varies according to the **Chart Type** selected.

   When the **Chart Type** selected is one of the following: *Area Chart*, *Bar Chart*, *Line Chart*, or *Line with Area Chart*, the mappings displayed are as follows:

   - **Orientation**: Choose between horizontal and vertical orientation types from the drop-down list.
   - **X axis label and Y axis label**: Optionally enter labels for X axis and Y axis.
   - **Label**: The label is auto-generated from the SQL statement which represents the X axis.
   - **Value**: The Value is auto-generated from the SQL statement which represents the Y axis (the values to be plotted on the chart).
   - **Refresh Columns**: Refresh the data of all the columns selected in the SQL statement.
   - **Coordinate System**: Choose between Cartesian and Polar coordinate systems from the drop-down list.
   - **Sorting**: Choose a sorting type (ascending, descending, off) for plotting the values. If selected, the Sorting value overrides the order specified by the `ORDER BY` clause in the SQL statement. For a large amount of data, Oracle recommends using the `ORDER BY` clause to sort data.

   When the **Chart Type** selected is one of the following: *Pie Chart*, *Pyramid Chart*, or *Funnel Chart*, the mappings displayed are as follows:

   - **Style**: Select from either 2D or 3D styles.

- **Series ID**: The Series ID is auto-generated from the SQL statement which represents the category to be plotted.
- **Value**: The Value is auto-generated from the SQL statement which represents the values to be plotted for each category.

Preview/Auto-preview: Click preview to view the changes made to the chart. If Auto-preview is checked, the changes to the chart are refreshed automatically.

8. The **Chart** tab at the bottom displays the chart.
9. The **Data** tab at the bottom displays the data resulting from the SQL command.
10. Click **Create** to create the chart, or click **Save** when editing the chart.

> **Note:**
>
> Due to browser resource restrictions, the preview of charts is limited to 3000 rows from your SQL query results. However, once you create a chart, all the data is rendered in the standalone view.

6.3 Example: Creating and Editing a Chart

This section provides an example of creating and editing a chart.

Creating a Chart

The following example shows how to create a chart to understand the cumulative salary earned by each job category.

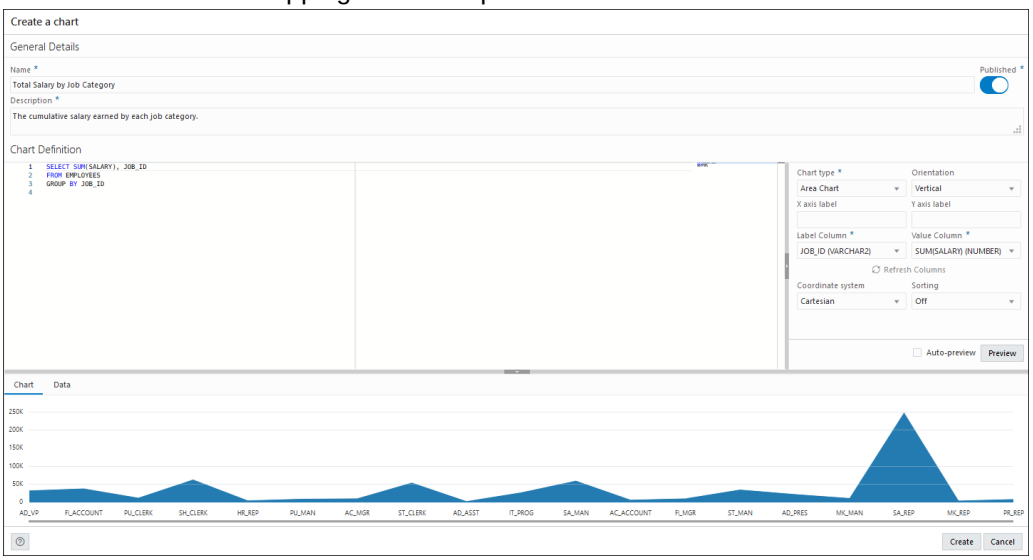
1. Navigate to the **Charts** page from Database Actions homepage.
2. Click **Create Chart**.
3. Enter the details of the chart as follows:
   - **Name:** Total Salary by Job Category
   - **Description**: The cumulative salary earned by each job category.
   - **Chart Definition**:
     - **SQL Statement**:

```
SELECT SUM(SALARY), JOB_ID
FROM EMPLOYEES
GROUP BY JOB_ID
```

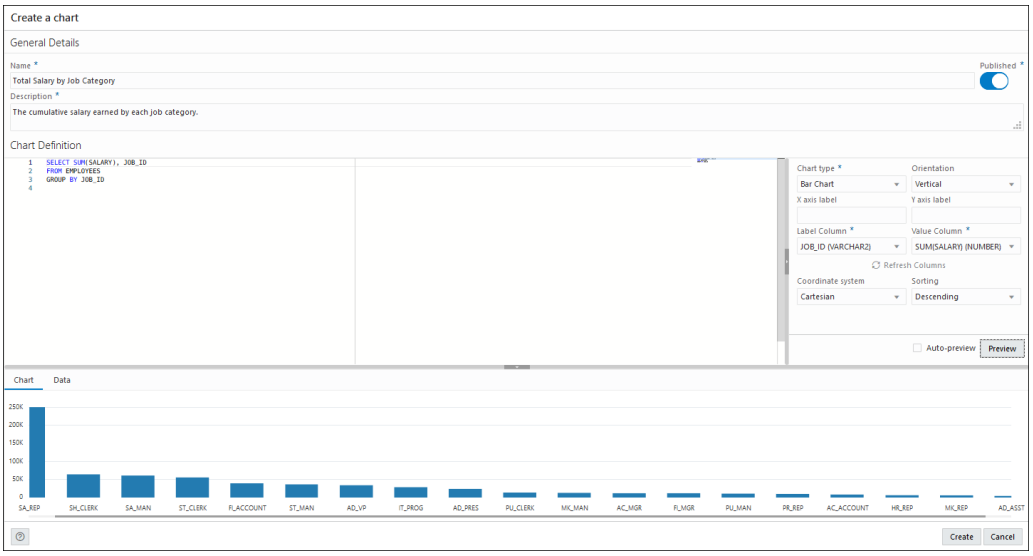
     - **Chart Type**: Area Chart
     - **Orientation**: Vertical
     - **X axis label**: Job Category
     - **Y axis label**: Total Salary
     - **Label**: `JOB_ID (VARCHAR2)`
     - **Value**: `SUM(SALARY) (NUMBER)`
     - **Coordinate system**: Cartesian

- **Sorting**: Off

- Click **Preview** in the Mapping section to preview the chart below.

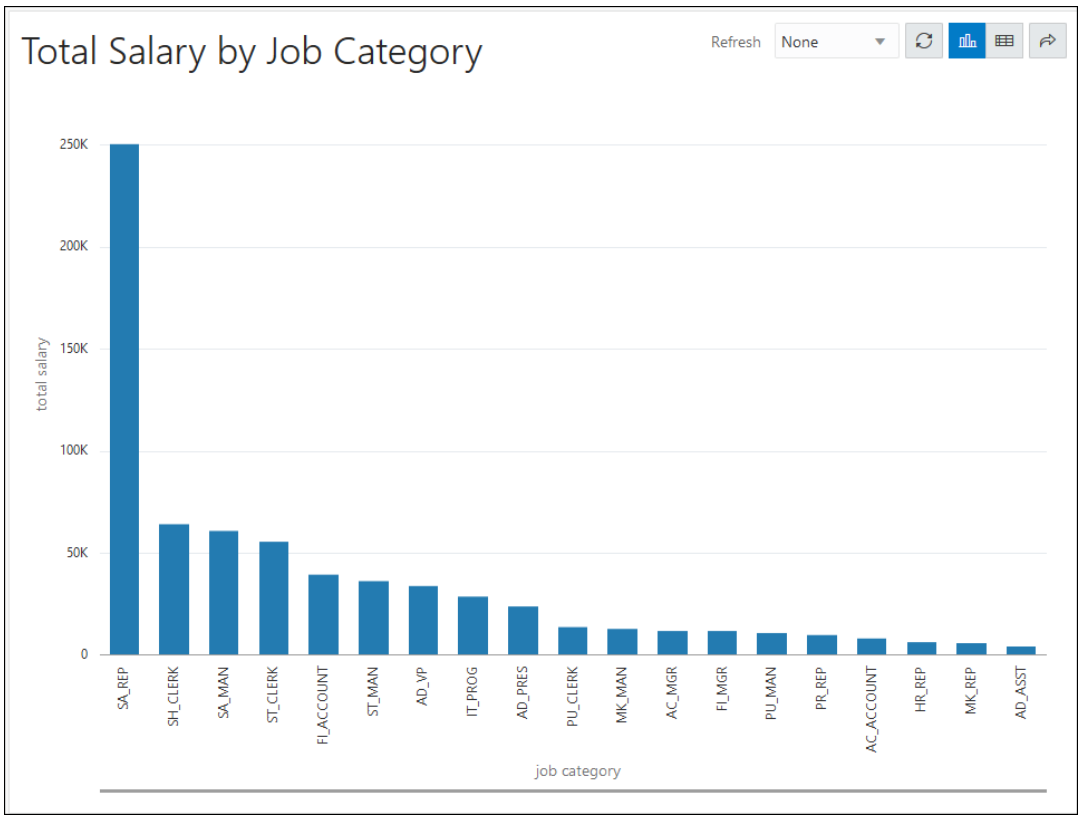


- Change the **Chart Type** to *Bar Chart*, **Sorting** to *Descending* and click **Preview** to view the updated chart.



4. Click **Create** to create the chart. A confirmation notification is displayed and the chart card is visible in the Charts page.

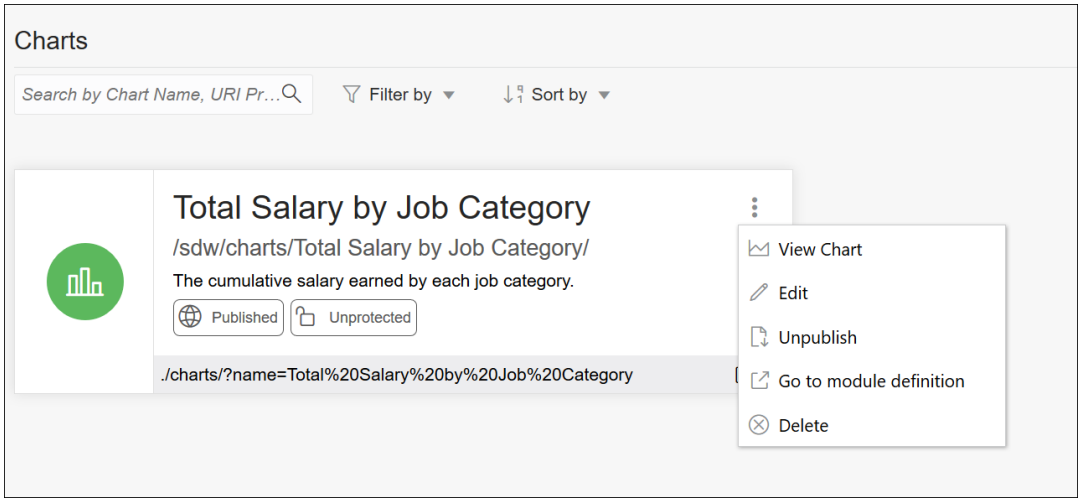
5. Click the Selector and select **View Chart** to view the chart in a new tab.



Editing a Chart

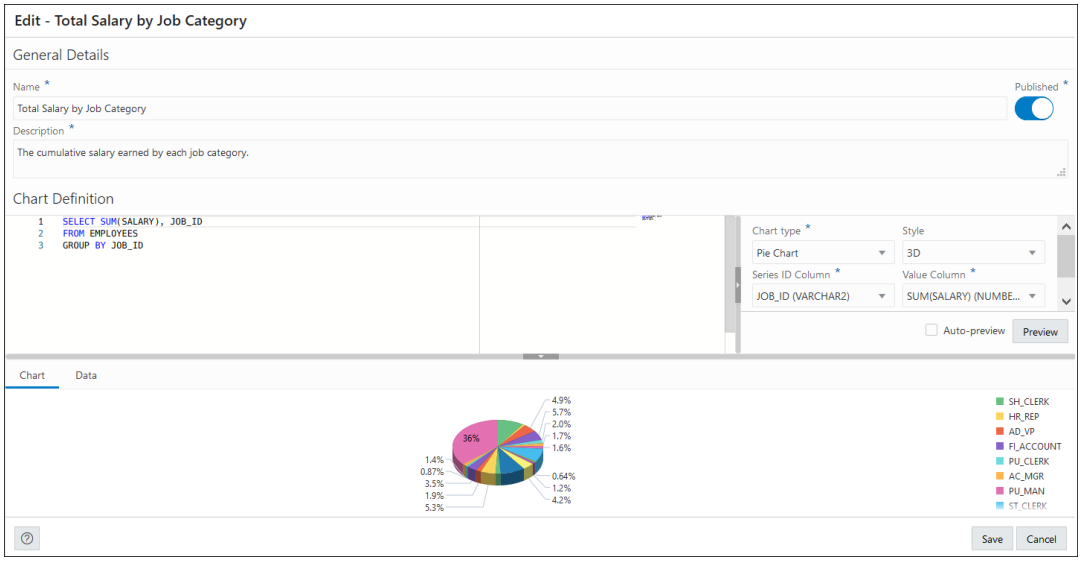
The following example edits the chart created in the previous section.

1. Click the Selector and select **Edit** to edit the chart.



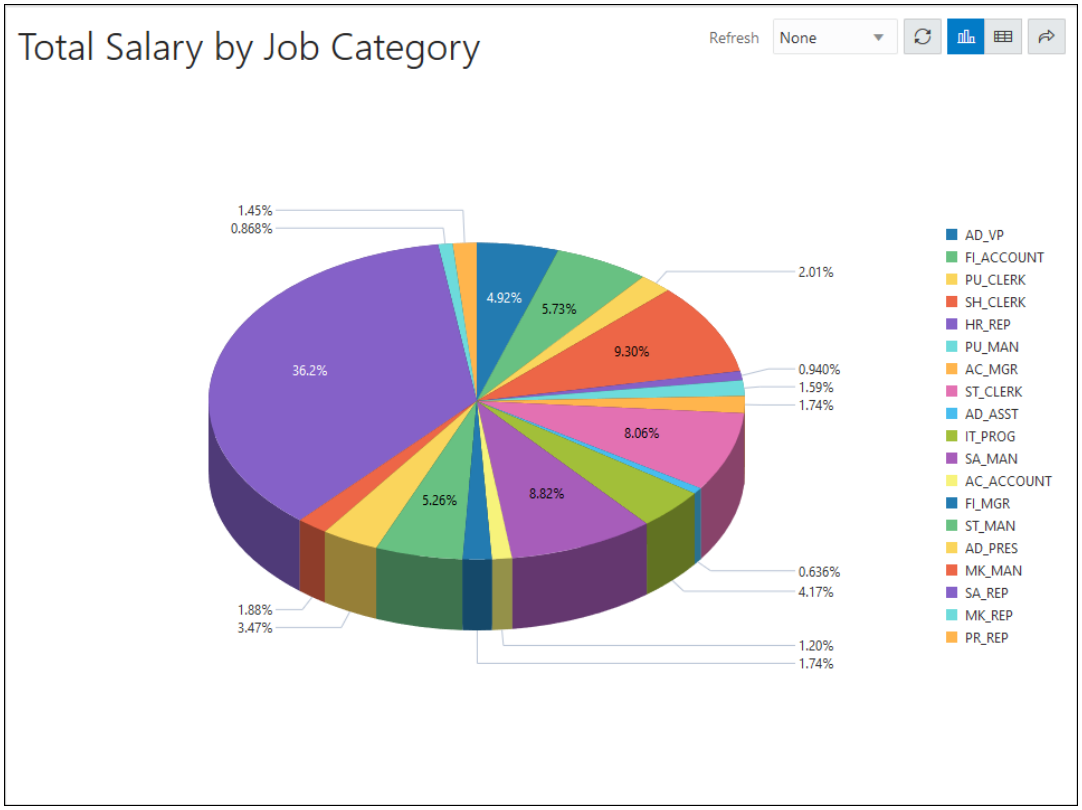
2. In the Mapping section, change the **Chart Type** to *Pie Chart* and **Style** to *3D*.

3. Click **Preview** to view the updated chart.



4. Click **Save**.

5. Click the Selector ⋮ and select **View Chart** to view the updated chart in a new tab.



6.4 Creating or Editing a Dashboard

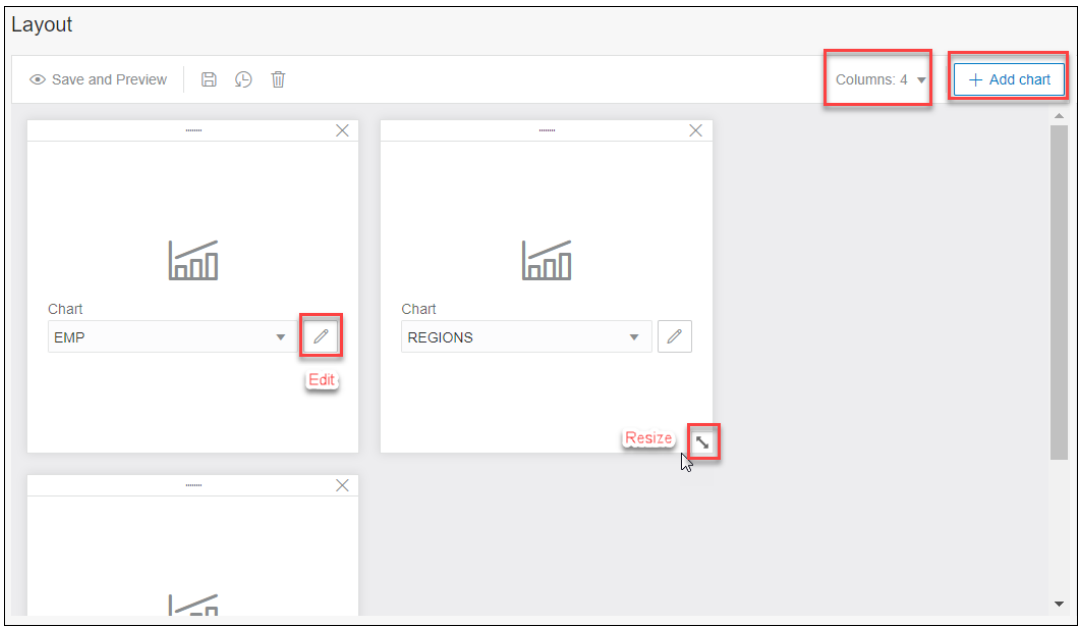
This section describes the steps to create or edit a dashboard.

To create a dashboard, in the Charts and Dashboard page, at the top right, click **Create** and then select **New Dashboard**.
To edit a dashboard, click the Actions icon and select **Edit**.

1. In the Create a Dashboard pane, enter or edit the following fields:
   - **Name**: Enter a name for the dashboard.
   - **Published**: Select this option to make the dashboard available for use.
   - **Description**: Enter a description for the dashboard.
   - **Protected by Privilege**: Select an available privilege from the drop-down list to enable only those with this privilege to securely access the dashboard.
   - **Go to Dashboard details after creation**: Select this option to go to the main Charts and Dashboards page after creating the dashboard.
2. Click **Create** or click **Save**, if editing. The Layout page is displayed
3. In the Layout page, you can add charts and design the layout for the dashboard. Click **Add Chart**.
   - In the **Select Chart** pane, for the **Selection** tab, select the chart to add from the drop-down list.
     You can select the **Custom** option and enter a URL for the chart from your schema or from a different schema. The URL must point to the REST Module root, for example, `http://xxxx.oracle.com:1234/ords/hr/sdw/charts/Chart01/`. Also, you need to have the associated privileges to view the chart.

     A preview of the selected chart is displayed. You can change the refresh rate of the chart, display the chart in Chart or tabular format, or share the chart by copying the URL.
   - In the **Settings** tab, select **Chart** and **Data Grid** to enable both views for the chart. By default, the Chart view is displayed in Preview.
4. Click **Create**.

   The chart is added to the Layout section.



5. Add more charts as needed to the dashboard. The maximum is 3 or 4 based on the number of columns that you select from the Columns drop-down list.
   - You can reorder the charts by dragging them horizontally or vertically to move them to the required position.

- Use the Edit icon on a chart to edit the chart if needed. You can also change the chart by selecting from the drop-down list.
- Drag the double-sided arrow at the right corner to resize the chart.

6. Click **Save and Preview** to save the layout and display a preview of the dashboard.

   The dashboard is displayed on a new page. For each chart, you can change or customize the refresh rate, display the chart in Chart or tabular format, or share the chart by copying the URL.

7

The JSON Page

> **Note:**
>
> Available for Oracle Database release 19c and later releases and only if you signed in as a database user with the **SODA_APP** role.

Use the JSON page in Database Actions to view and manage JSON collections, search for collection items using Query-by-Example (QBE), create JSON search indexes and display data guide diagrams for collections.

To navigate to the JSON page, do either of the following:

- In the Launchpad page, select the **Development** tab and click **JSON**.
- Click **Selector** to display the navigation menu. Under **Development**, select **JSON**.

Topics

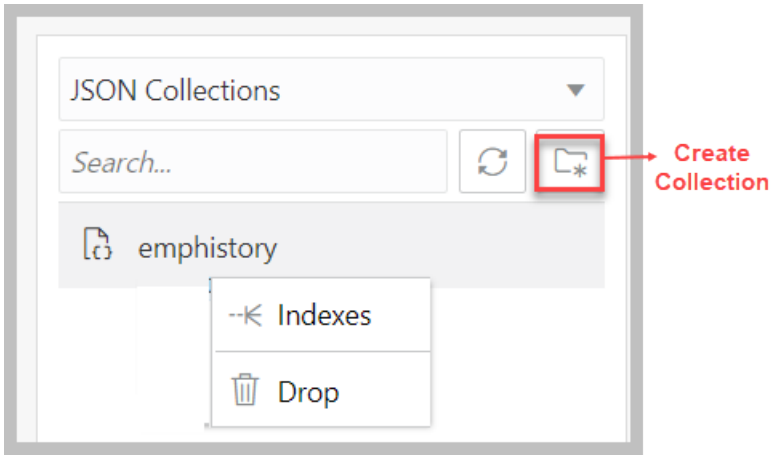
> **See Also:**
>
> SODA Overview in *Oracle Database Introduction to Simple Oracle Document Access (SODA)*

7.1 About the JSON User Interface

The JSON user interface consists of the left pane for listing and searching saved collections and the right pane for viewing and managing documents in a collection.

Listing and Searching JSON Collections

The following figure shows the main items in the left pane of the JSON page.



Select the appropriate option in the drop-down list to display saved JSON collections or recently accessed JSON collections.

The search functionality is not case-sensitive, retrieves all matching entries, and does not require the use of wildcard characters.

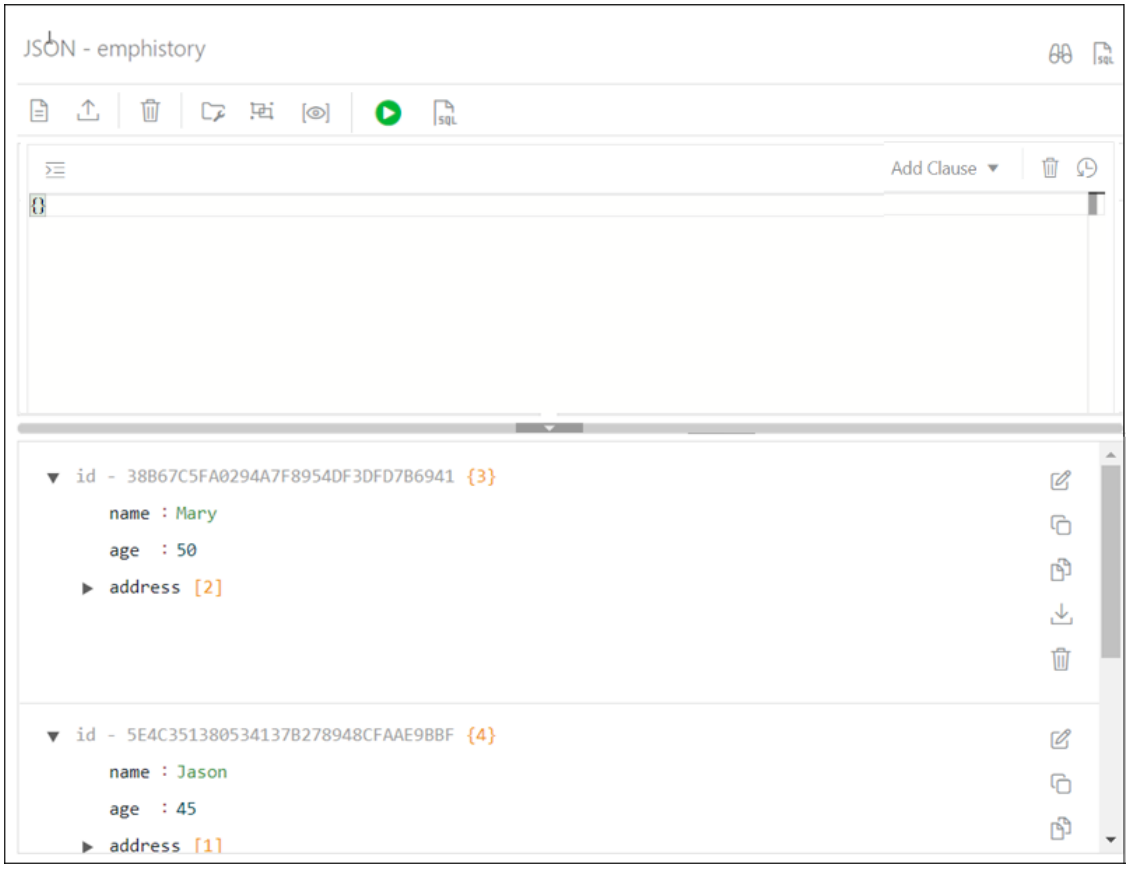
Click the Create Collection icon to create a new collection. See Creating a Collection

Right-click a collection name to open the context menu. The available options are:

- **Indexes**: Enables you to view existing JSON indexes and create search, functional or spatial indexes.
- **Drop**: Removes the collection from the database completely.

Viewing the Contents of a Collection

When a specific collection is selected in the left pane, the JSON documents that belong to the collection are displayed in the lower part of the right pane. The top part of the right pane contains the JSON editor where you can run queries for filtering or sorting the documents.



The two icons in the right corner are:

- **Tour** : Starts the JSON tour, which provides information about the features available.
- **Go to SQL** : Navigates to the SQL page.

The icons in the toolbar are:

- **New JSON Document** : Adds a new document to the collection. See About Adding or Editing a JSON Document
- **Import Document(s)** : Imports one or more existing JSON files from your local computer into the collection.
- **Delete All Documents in the list** : Deletes all JSON documents in the collection that match the current QBE search string. If the current string is {}, then all the documents in the collection are deleted.
- **Collection Details** : Enables you to view collection properties, JSON data guide (if created) and related statistics if they are gathered, size of search index, and page for managing JSON indexes.
- **Diagram** : Displays the the JSON data guide as a diagram in a hierarchical format. See Viewing the JSON Data Guide Diagram for a Collection
- **New Collection View** : Creates relational views of documents in a collection.

- **Run Query** : Filters documents using the QBE condition entered in the JSON editor. See About Querying Documents in a Collection
- **View Collection SQL**: Displays the JSON collection in SQL format. You can execute the SQL code, download it or copy to clipboard.

Each JSON document has the following icons:

- **Edit Document** : Edits the JSON document. See About Adding or Editing a JSON Document
- **Clone Document** : Creates a clone of the document.
- **Copy Document** : Copies the document to the clipboard.
- **Export Document** : Downloads the document as a .JSON file.
- **Delete Document** : Deletes the document.

7.2 Managing JSON Collections

You can add, view or drop collections, or browse, add, edit and delete JSON documents in a collection.

- **Create a Collection**: See Creating a Collection
- **Add or Edit Documents in a Collection**: See About Adding or Editing a JSON Document
- **View Documents in a Collection**: Select the collection name in the left pane. The documents in the collection are displayed in the right pane.
- **View Collection Details**: Select the specific collection in the left pane, and then click **Collection Details** in the right pane toolbar to view collection properties, JSON data guide (if created) and related statistics if gathered, the size of search index, and the page for managing JSON indexes.
- **Drop a Collection**: Right-click a collection name in the left pane to open the context menu. Select **Drop** to remove the collection from the database completely.

7.2.1 Creating a Collection

This section describes how to create a collection.

1. When you navigate to the JSON page:
   - If there are no collections created, you see the JSON home page. Click **Create Collection**.
   - If there are existing collections available, then in the left pane, click **Create Collection** .
2. Enter the collection name, which is case sensitive.
3. Click **Create** to create the collection.

   Click **Cancel** or press the **Esc** key to cancel the collection.

> **See Also:**
>
> SODA Collection Metadata Components in *Oracle Database Introduction to Simple Oracle Document Access (SODA)*

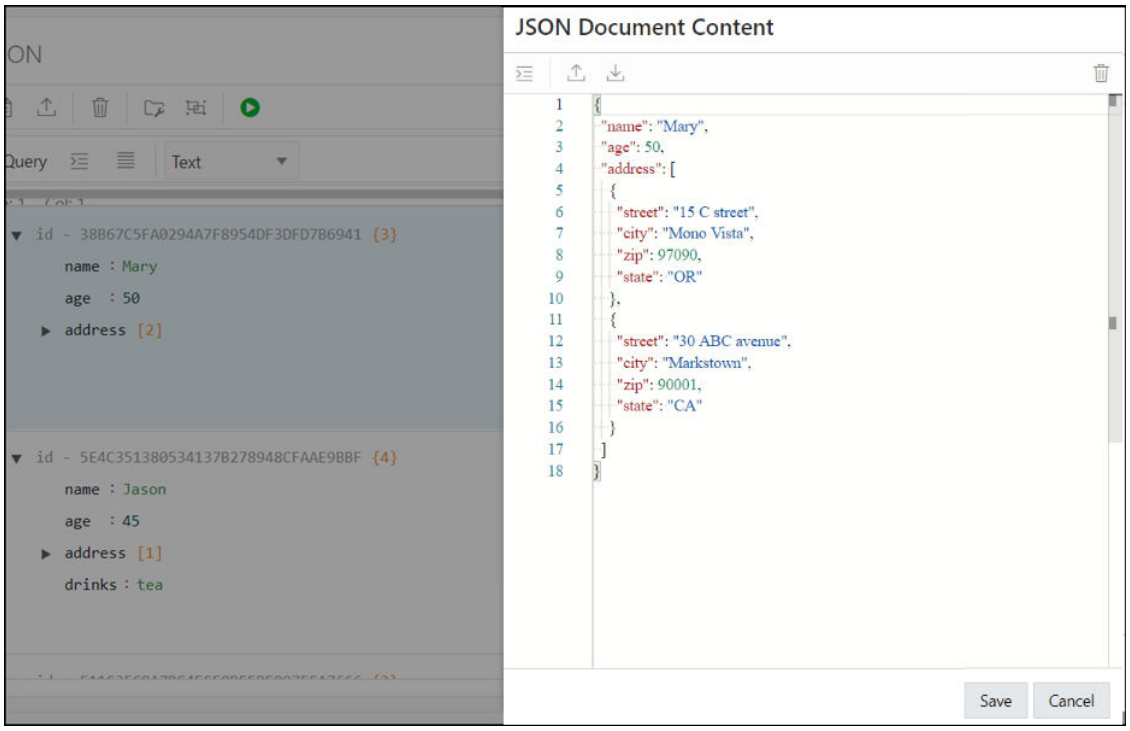
7.2.2 About Adding or Editing a JSON Document

You can add and edit JSON documents using the JSON editor.

To add a document, click **New JSON Document** in the right pane toolbar for a specific collection. In the JSON editor, you can copy and paste the JSON document or use the Import icon to import the JSON document.

To edit a document, click **Edit Document** in the document card.

The following figure shows an open document in the JSON editor.



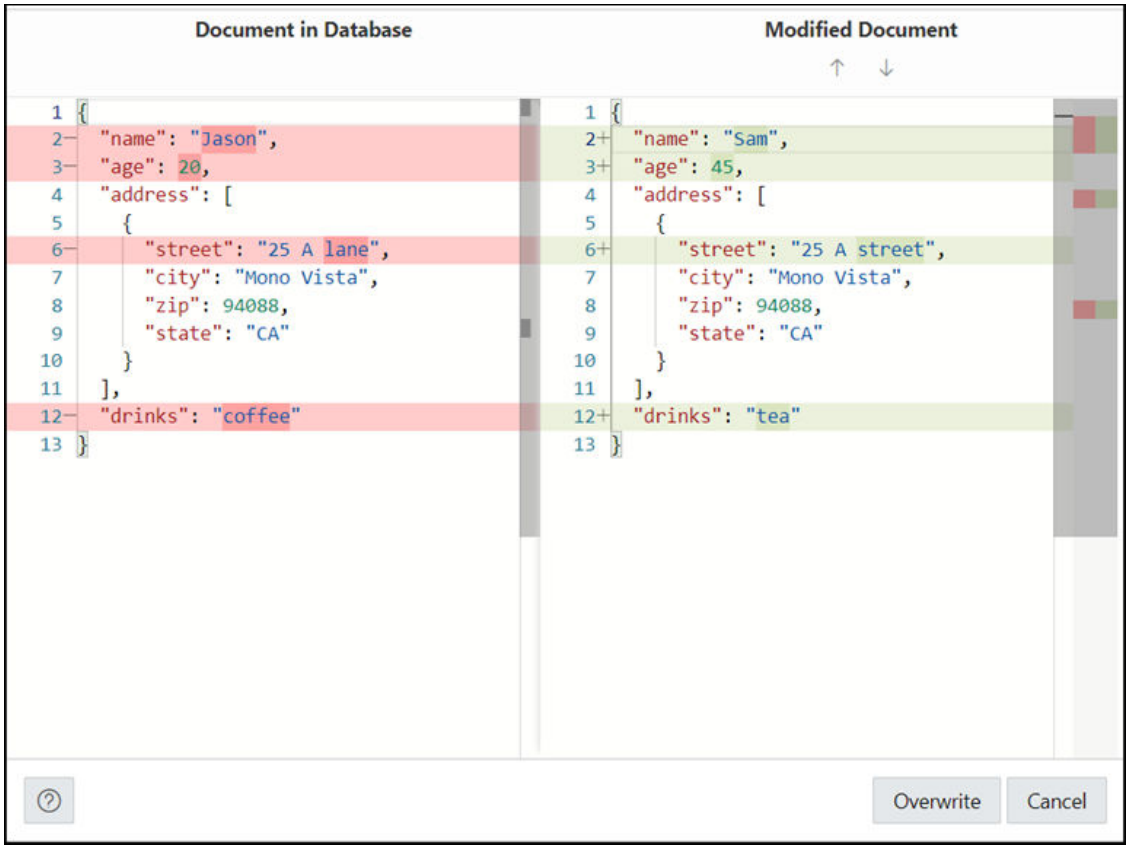
7.2.2.1 About Database Differences in JSON Documents

You can find differences in a JSON document that has been edited from two different connections.

At the time of saving an edited document, you are notified that there are differences between your edited document and the instance of the document in the database. The differences are also highlighted enabling a quick review.

Click **Overwrite** if you prefer to overwrite the changes to the document instance in the database.

Figure 7-1 Differences in the Document Displayed



7.3 About Querying Documents in a Collection

You can search for one or more documents in a collection by using a filter specification or Query-by-Example (QBE).

A QBE is a pattern expressed in JSON. You use it to select, from a collection, the JSON documents whose content matches it, meaning that the condition expressed by the pattern evaluates to true for the content of only those documents. For more information about QBEs and how to use them, see Overview of SODA QBEs in *Oracle Database Introduction to Simple Oracle Document Access (SODA)*.

For a specific collection, enter the QBE string in the JSON editor. For example, to select documents where the name is Mary, enter {"name":"Mary"} and then click **Run Query**, as shown in Figure 7-2. The results of the query are displayed in the lower right pane.

The editor offers a comprehensive list of commands available through the Command Palette. To open the Command Palette, press **Ctrl+Shift+P**. For a list of keyboard shortcut keys, see Keyboard Shortcuts.

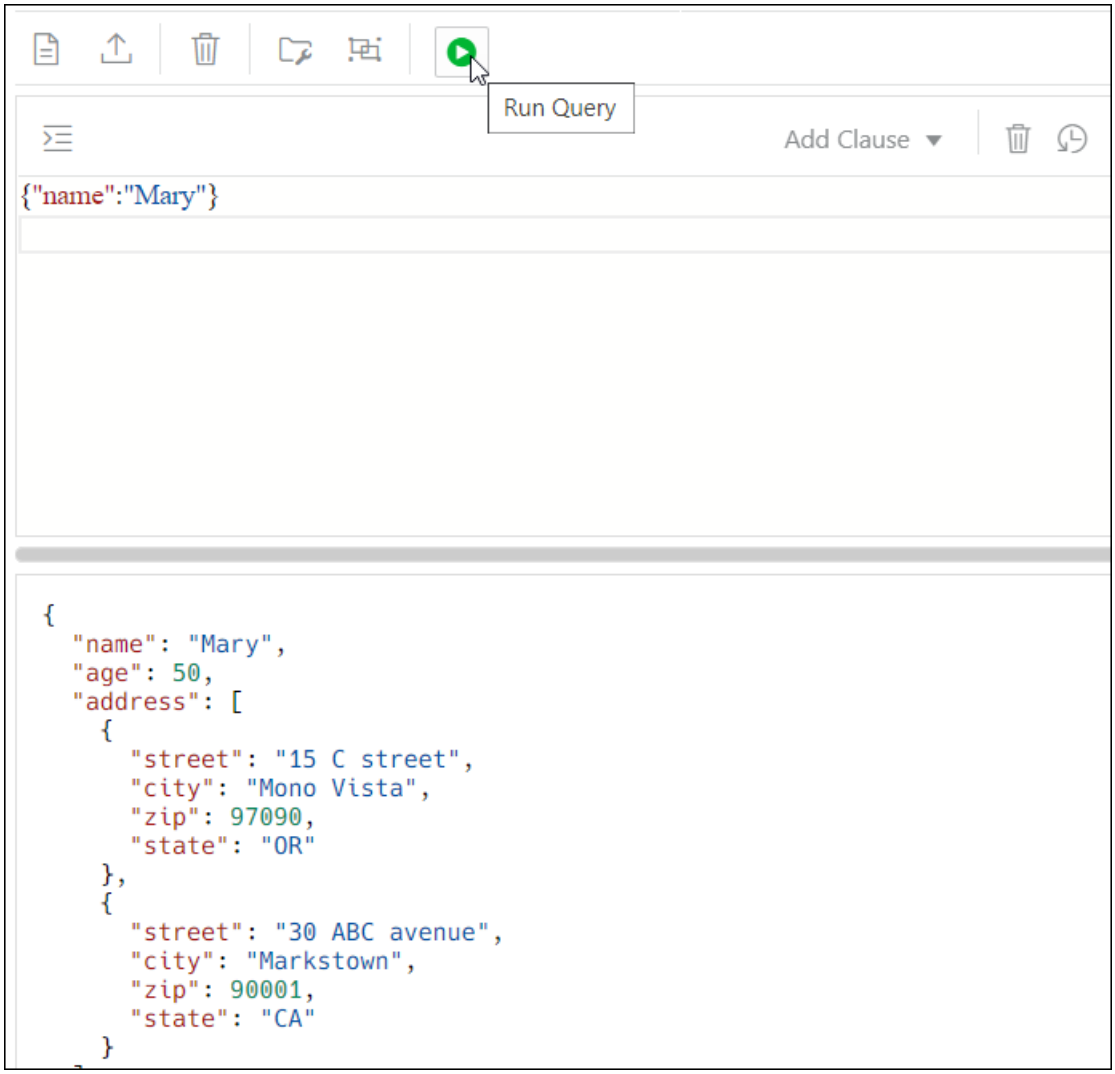
An error in the query is signified by a red dot in the left gutter and a squiggle line beneath the specific text. When you hover over it, you see a pop-up displaying possible fixes for resolving the error.

You can set editor preferences using the **Preferences** option in the top-right user drop-down list on the header. Some of the available options are Theme (Light, Dark and High contrast dark), Font size and family, Tab size, Word wrap, Ruler, Line numbers and so on.

The icons above the editor are:

- **Format JSON data** : Enables indentation and line feeds for the QBE string.
- **Add Clause**: Adds a formatted template of the $orderby or $patch clause to the QBE string.
- **Clear** : Clears the current QBE string.
- **History** : Retrieves previous QBE search strings.

Figure 7-2 QBE String in JSON Editor

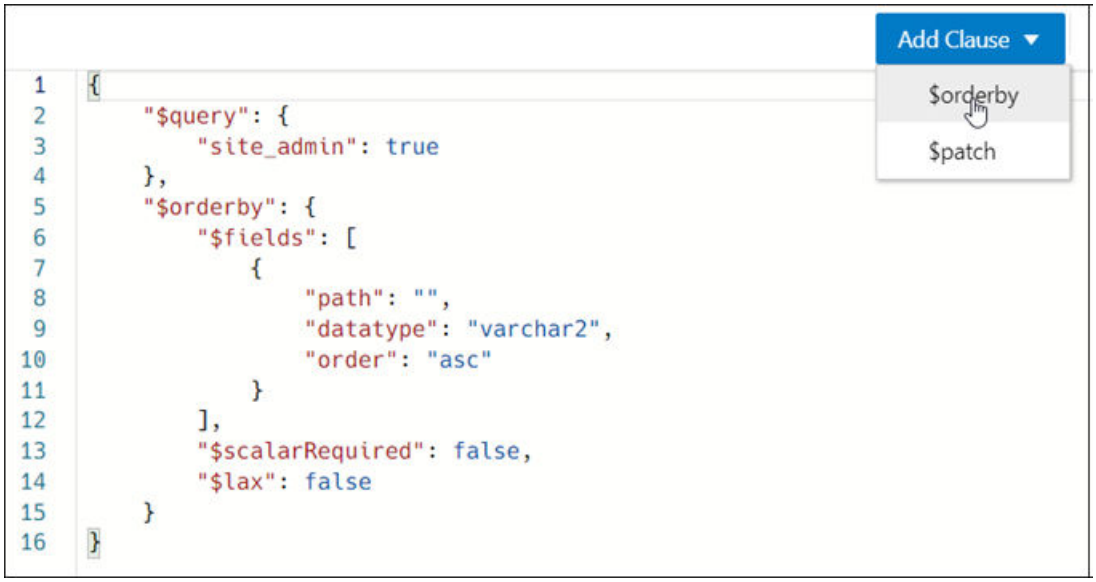


The QBE expression must be a valid JSON object and can contain $query and $orderby or $patch clauses. The QBE expression is treated as a $query clause if there are no clauses defined. Starting with the $query clause, add the $orderby or $patch clause later using the **Add Clause** list. The content is transformed and a template for the $orderby and $patch clause is provided. You need to set correct values in the templates. See Using the In-Context Autocomplete Feature in the JSON Editor

For example, this is a simple filtering query:

```
{
    "site_admin": true
}
```

After the $orderby clause is added using Add Clause, you see the following entry:



After the $patch clause is added, the query changes to:

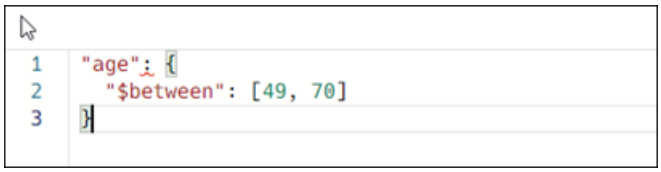
7.3.1 Using the In-Context Autocomplete Feature in the JSON Editor

If you press **Ctrl+Space**, the editor provides you with a context-aware list of options from which you can select and autocomplete at the insertion point.

The following types of information appear in the list:

- **Filter comparison clauses**

  A template is available for each clause and it is inserted at the cursor position. For example, when you select the `$between` clause from the autocomplete list, you see the following entry in the editor:



  where `age` is the property name and `49` and `70` are values. These are parameters that you can edit and they appear highlighted. You can type or use the autocomplete help to edit the property name. Press the **Tab** key to move to the next parameter.

- **Property names from JSON documents in the collection**

  There are two sources for property names from JSON documents:

  – If a search index is created with support for data guide, then the JSON data guide from the database dictionary is scanned for property names.

  – Property names that are collected from viewed or edited documents are presented. It is possible that the property names are a subset from the whole namespace.

  Based on the type of property, the related template is inserted and the cursor is positioned at the expected place for insertion. The following is a list of property types, their templates and the corresponding cursor position for each:

  – Object

```
"ShippingInstructions.Address": {
<cursor_here>
}
```

- Array

  ```
  "LineItems": [<cursor_here>]
  ```

- String

  ```
  "LineItems[*].Part.Description": "<cursor_here>]"
  ```

- Number

  ```
  "PONumber": 0
  ```

- Boolean

  ```
  "site_admin": true
  ```

When the cursor is between double quotes (“”) and autocompletion is activated, then only the property name is inserted without templates or additional double quotes.

7.4 Creating Indexes for JSON Collections

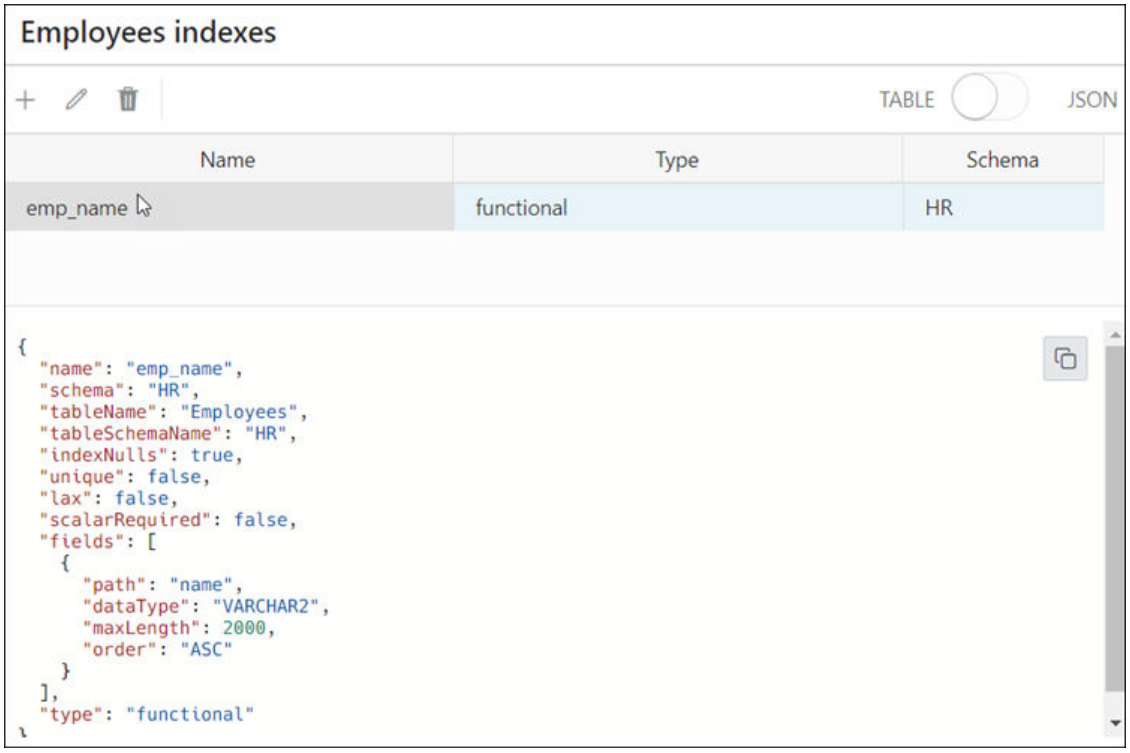
You can create indexes for JSON collections in the JSON page.

> **See Also:**
>
> Indexes for JSON Data in *Oracle Database JSON Developer's Guide*

Open the Indexes Pane

In the JSON left pane, right-click the collection, and select **Indexes**.

The Indexes pane lists the existing indexes for the collection. Select the index row to display more information.

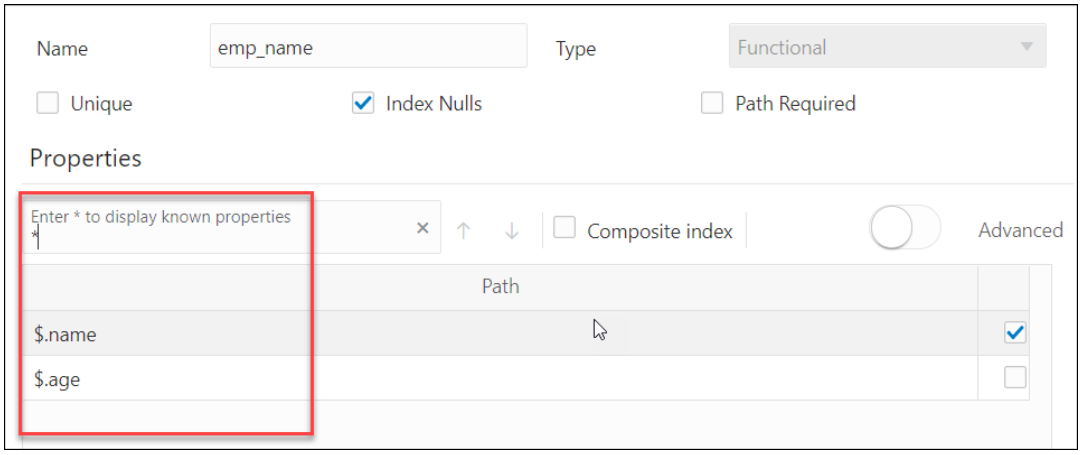


The icons at the top are **Add JSON Index**, **Edit JSON Index**, and **Delete JSON Index**.

The properties of the selected index appear in JSON format below the listed indexes. Select **JSON** from the TABLE - JSON option to view all indexes in JSON presentation.

Create an Index

1. Click the + **New JSON Index** icon. The New Index pane appears.
2. Enter the following fields to create an index:



- **Name**: Enter a name for the index.
- **Type**: Select the index type from the drop-down list. The different options are Functional, Spatial and Search. Based on the index type selected, the corresponding options appear.

- For a functional type index, the fields to enter are:
  - **Unique**: Select this option to make all indexed values unique.
  - **Index Nulls**: Select this option to use the index in Order By queries.
  - **Path Required**: Select this option if the path must select a scalar value, even a JSON null value.
  - **Properties**: Type the property that you want to index on, or Type ***** to display all available document properties in the collection. To select a property, select the checkbox in the respective row.

    > **Note:**
    >
    > You cannot index properties in arrays.

  - **Composite Index**: Select this option if you want to use more than one property.
  - **Advanced**: Select this option to change the storage properties of the indexed property. For each property, you can change the type (varchar2, number, date or timestamp), maximum length for indexing (for character properties), and sort order.
- For search index, the options are:
  - **Dataguide off-on**: Select **on** to create JSON data guide for collection.
  - **Text Search off-on**: Select **on** to index all properties in documents to support full-text search based on string equality (every property is treated as string)
  - **Range Search off-on**: Select **on** to support range search when string-range search or temporal search (equality or range) is required.
- Spatial index is used to index GeoJSON geographic data. The selected property should be of GeoJSON type. See Using GeoJSON Geographic Data

  For spatial index, the options are:
  - **Path Required**: Select this option if the path must select a value, even if it is a JSON null value.
  - **Lax**: Select this option if the targeted field does not need to be present or does not have a GeoJSON geometry object as its value.

    > **Note:**
    >
    > You cannot enable **Path Required** and **Lax** at the same time.

3. Click **Create**. A notification is displayed indicating that the index is created and the Indexes pane is populated.

7.5 Creating Relational Views of JSON Documents

You can create relational views of JSON documents.

1. In the JSON page, click **New Collection View** in the right pane.
2. Enter the following fields:
   - **View Name**: Enter a name for the view.

- **Select columns to add**: Click in this field to view the list of available properties and to select the required properties, adding them one at a time. Alternatively, select **Add All** to add all objects and arrays with all their properties. Select one or several properties and move them up or down to position them in the resulting view. Objects and arrays are moved or deleted together with all their properties.

  You can edit the Column Name attribute for each column, and for string types, you can also edit the Length attribute. Precision and scale can be set for numeric properties, for example, (10,2).

3. Click **Test Query** to review the DDL statements before creating the view.

   Click **Create** to create the view, or click **Definition** to return to the previous screen and make changes to the corresponding attributes.

   The created view is now available in the SQL page.

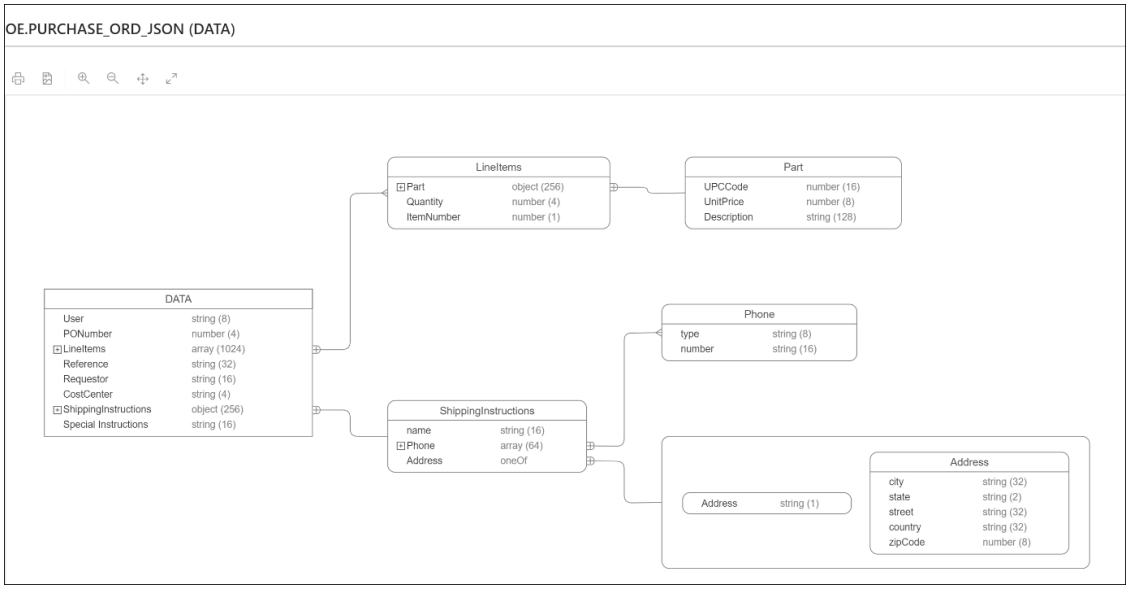
7.6 Viewing the JSON Data Guide Diagram for a Collection

The diagram view displays the JSON data guide for a collection as a hierarchical structure using a format similar to entity-relationship diagrams.

The JSON data guide represents the JSON schema for documents that have a column with JSON content.

In the diagram, arrays are presented as one-to-many relationships, contained objects as one-to-one relationships, and "oneOf" constructs as a box that surrounds possible choices.

Click **Diagram** in the right pane toolbar to display the JSON data guide diagram for a specific collection.



The icons in the toolbar are Print Diagram, Save to SVG format. Zoom In, Zoom Out, Fit Screen and Actual Size.

> **See Also:**
>
> JSON Data Guide in *Oracle Database JSON Developer's Guide*

8
The MLE JS Page

> **Note:**
>
> A database user requires the following privileges to use the MLE JS feature:
>
> - `CREATE MLE`
> - `EXECUTE DYNAMIC MLE`
> - `EXECUTE ON JAVASCRIPT`

With the introduction of Oracle Database Multilingual Engine (MLE), it is possible to run JavaScript directly in Oracle Database. For more information about MLE, see Introduction to Oracle Database Multilingual Engine in the *Oracle Database JavaScript Developer's Guide*.

The MLE JS feature in Database Actions supports JavaScript code development in Oracle Database 23c and later versions. To use this feature in Database Actions, you must first understand the concepts about MLE, which is explained in detail in the *Oracle Database JavaScript Developer's Guide*.

In the MLE JS interface, you can:

- Managing MLE Modules
- Managing MLE Environments
- Managing MLE Call Specifications
- Executing JavaScript Code Snippets and Debugging MLE Modules

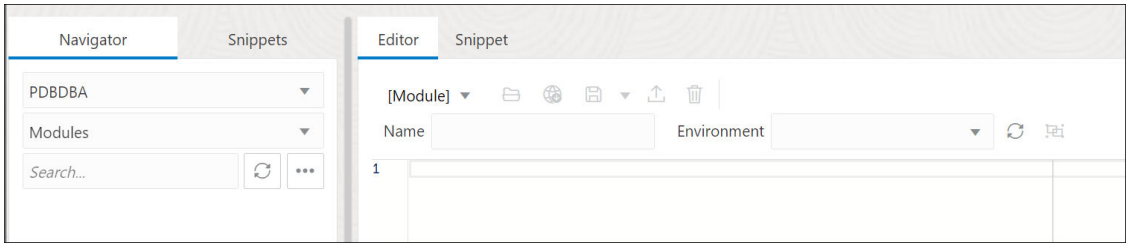
To navigate to the MLE JS page:

- In the Launchpad page, select the **Development** tab and click **MLE JS**.
- Click **Selector** to display the navigation menu. Under **Development**, select **MLE JS**.

> **See Also:**
>
> MLE LiveLab (Includes using Database Actions to create an MLE module, environment and call specification, and debug an MLE module)

8.1 About the MLE JS Page

The MLE JS page is divided into two parts. The left pane consists of the Navigator and Snippet tabs. The right pane includes the JavaScript code editor at the top and results of the code output at the bottom (this applies only for the Snippet tab in the right pane).



Left Pane

The left pane has the following two tabs:

- **Navigator** to access saved MLE modules, environments and call specifications.
- **Snippets** to access JavaScript code snippets saved in the browser or file system. See Executing JavaScript Code Snippets and Debugging MLE Modules

In the Navigator tab, you have:

- **Schema and Object type selectors**: Use the drop-down lists to select the schema and filter the results by object type (whether modules, environments or call specifications).
- **Search**: Search for objects in the Navigator tab.
- **Object submenu** - **Create Object**: Create a new object for the object type.
- **Object submenu** - **Compile All** (for Environments and Call Specifications): Compile all invalid environment objects or call specification objects.

Code Editor

The JavaScript code editor in the right pane supports autocompletion of text preceding the cursor when using the **Ctrl+Space** key. The autocomplete functionality is triggered for:

- built-in JavaScript modules such as `mle-js-oracledb`.
- export statements of modules in the environment object.
- variable declaration statements exported by the module.

The code editor has two tabs: Editor and Snippet.

Editor Tab

The icons in the Editor tab are:

- **New**, **Save As**, **Code Dependencies Diagram**: Create a module, save a module, and view the code dependencies diagram.
- **Open File**: Open a JavaScript file from your file system.

> **Note:**
>
> The Open File icon is available only when using a Chromium-based browser in a secure context (HTTPS).

- **Save Module**: Save the module in the database, file system, or in both locations.
- **Import JS Files**: Load multiple JavaScript files from your file system to create them as MLE modules. See Loading Multiple JavaScript Files to Create MLE Modules

> **Note:**
>
> The Import JS Files icon is available only when using a Chromium-based browser in a secure context (HTTPS).

- **Clear**: Remove the contents of the editor.
- **Name**: Enter a name for the MLE module.
- **Environment**: Select the environment for the module.
- **Environment Dependencies Diagram**: Show imported modules and exported functions for each module in the environment in a diagram format.

Snippet Tab

Additionally, the Snippet tab has the following icons:

- **Save**: Save the code snippet that you entered in the editor to the browser or device.
- **Debug Snippet**: Run the debug operation to collect debug information for the selected code snippet.
- **Debug Specification**: Specify the debug specification to use to debug the MLE module.
- **Debug Specification Menu** (**New**, **Edit**, **Delete**, **Import**, **Export**): Use the options in this menu to create, edit, or delete a debug specification, and also import or export a debug specification to the file system.

> **Note:**
>
> The Import and Export Debug Specification options are available only when using a Chromium-based browser in a secure context (HTTPS).

8.2 Managing MLE Modules

A JavaScript module is a unit of MLE's language code stored in the database as a schema object. You can create, edit and delete JavaScript modules.

Topics:

8.2.1 Creating an MLE Module

A JavaScript module can be created in two ways:

- By entering Javascript code into the editor
- By opening the file already available on your local computer

Creating a JavaScript Module by Entering Code in the Editor

To create an MLE module:

1. In the right editor pane, select the **Editor** tab (if not already selected) and enter the JavaScript code.
2. In the **Name** field, enter a name for the module.
3. Select the appropriate environment from the drop-down list.

   This is needed when importing or exporting modules already saved to the database and imported in the selected environment.
4. Click **Save** to save in database, file system or both database and file system.

Creating a JavaScript Module by Opening the File

To create an MLE module:

1. In the right pane, select the **Editor** tab, click **Open File** and browse for the file you want. The file contents are displayed in the editor.
2. In the **Name** field, enter a name for the module.
3. Click **Save**.

8.2.2 About Context Menu Options for MLE Modules

In the left Navigator pane, right-click the MLE module name to access the following options:

- **Edit**: Select to edit the attributes of the MLE module.
- **Drop**: Select to remove the MLE module from the database.
- **Dependencies Diagram**: Shows module dependencies in the form of a diagram. The diagram also includes the interlinking of all modules dependent on the specified module. These dependencies are based on the imports and exports in each used module. The details of the exported and imported functions can be seen on the diagram.
- **Create Call Specification**: Opens the Create MLE Call Spec panel. Enables you to create a call specification for the MLE module.
- **Compare (File to Module, Module to File)**: Compares the module version saved in the database (or file system) to the version saved in the file system (or database). When the module is saved in both database and file system (indicated by a round blue overlay icon), use this option to compare the two versions. The differences are highlighted enabling you to quickly identify the changes. Click **Overwrite** to implement the changes needed to synchronize both the database and file system versions of the module.
- **Show Code**: Displays the JavaScript code for the module in a separate panel.
- **Show Metadata**
  - **User Data**: Creates and edits user metadata that can be stored along with the module. The metadata must be in valid JSON syntax.
  - **Code Metadata**: When a module is saved, the metadata related to imports and exports is stored in the database. This metadata is used:
    * in tracking dependencies on dependency diagrams.
    * for code completion functionality such as code peek.
    * for creating an MLE environment.
    * for creating a call specification based on the module.

Only ES6 modules are supported. There is no support for CommonJS, AMD and UMD modules.

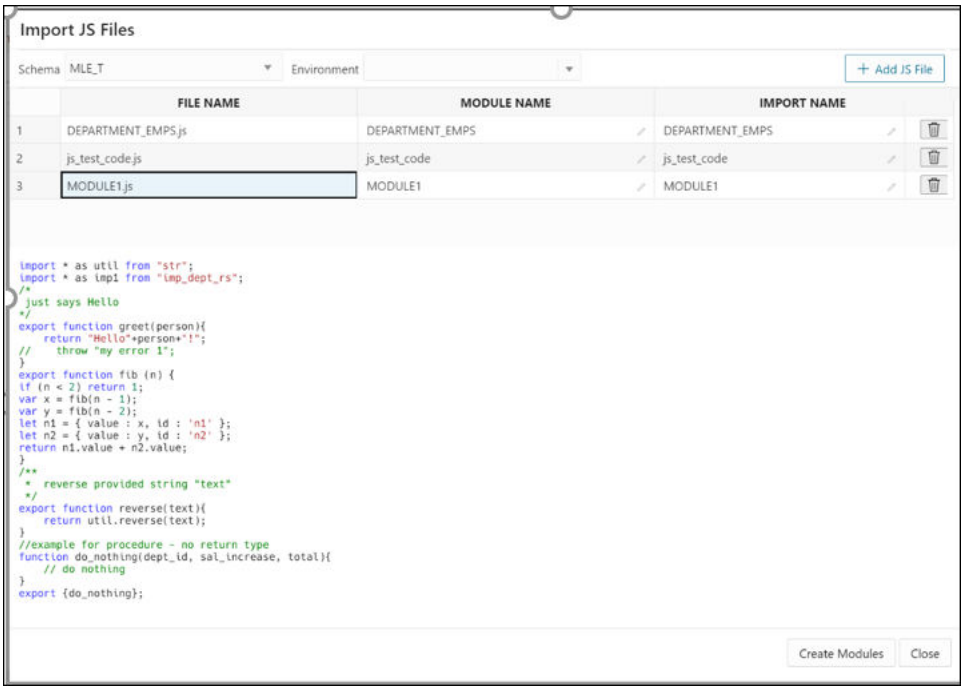
  - **Code**: Shows the module code.
- **Generate JSDoc**: Generates the Markdown file based on exports from the module. Data types for function parameters are included if the following exist:
  - JSDoc about function parameters in the code.
  - Call specification defined for the exported function.

8.2.3 Loading Multiple JavaScript Files to Create MLE Modules

You can load multiple JavaScript files to the database as MLE modules. This feature is supported only in Chrome and Chromium-based browsers. A maximum of 150 files can be uploaded in one session.

1. In the MLE JS page, in the right pane, select **Editor** and click **Import JS Files** from the toolbar.
2. In the Import JS Files panel:
   a. Select the target schema to create the modules.
   b. Select the environment from the target schema. You can enter a new name to create a new environment.
   c. Click **Add JS File** and select the files to add.

   The files appear in a tabular format in the panel. You can view the module code by clicking the file name. The module name and import name (for the environment) are generated based on the file name but they can be edited.
3. After all the changes are done, click **Create Modules**.



8.2.4 Example: Creating an MLE Module

This example demonstrates the creation of an MLE module.

1. In the MLE JS page, in the left pane, select **Modules** from the object type selector and then click **Object submenu** - **Create Object**.
2. In the right pane, select the **Editor** tab, and enter the following JavaScript function:

```
export function factorial(num) {
    if(num < 0) {
        return -1;
    } else if (num == 0) {
        return 1;
    } else {
        return(num * factorial(num-1));
    }
}
```

3. In the **Name** field, enter **factorial_mod**.

4. Click **Save Module**.

   A notification that the module is saved in the database is displayed.

8.3 Managing MLE Environments

MLE environments are schema objects that can be managed in the database. You can create, edit and delete MLE environments using Database Actions.

Creating an MLE Environment

To create an MLE environment:

1. In the left pane, select the **Navigator** tab, select **Object submenu** and click **Create Object**.

   The Create MLE Environment panel appears.
2. Select the schema from the list. By default, the current schema is used.
3. In the **Name** field, enter a name for the environment.

4. Select the appropriate **Language Options**. For more information, see JavaScript Language Options in *JavaScript Developer's Guide*.
5. The Available Modules field automatically populates the modules available for the selected schema. Select one or more modules. Click the **Add Selected Modules** icon or the **Add All Modules** icon to move the modules to the Imported Modules field. You can edit the Import Name field for the module if needed.

   When a single module is selected, the **Add module with dependencies** icon becomes available. Click this icon to find environments where the module is used and also to add other modules that the selected module depends on.

   To view the code for a module, in the Available Modules field, right-click the module and select **Details**.
6. The DDL pane displays the code for the new MLE environment. Click **Create**.

   The MLE environment is saved and appears in the left pane under Environments.

Context Menu Options

In the left pane, right-click the MLE environment to view the following options:

- **Edit**: Select to edit the MLE environment. You can modify the language options, add more modules to import or change the import names for the modules.
- **Drop**: Select to drop the environment from the database.
- **Compile**: Select to compile or recompile the environment.
- **Dependencies Diagram**: Shows the imported modules and exported functions for each module in the environment using a diagram.

See Example: Creating an MLE Environment for an example of how to create an MLE environment.

8.3.1 Example: Creating an MLE Environment

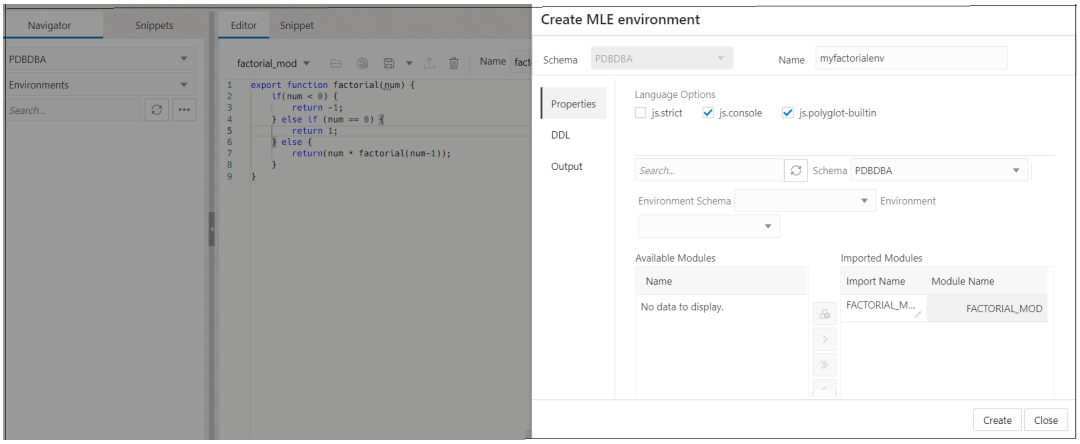
This example demonstrates the creation of an MLE environment and demonstrates how a mapping is created between the identifier FACTORIAL_MOD and the JavaScript module FACTORIAL_MOD created in Example: Creating an MLE Module.

Create the MLE Environment

1. In the MLE JS page, in the left pane, select **Environments** from the object type selector and then click **Object submenu** - **Create Object**.

   The Create MLE Environment panel appears.
2. In the **Name** field, enter **myfactorialenv**.
3. In the Available Modules field, select **FACTORIAL_MOD**. Click **Add Selected Module** to move the module to the Imported Modules field.

> **Note:**
>
> By default, the **Import Name** matches the Module name. You can edit the import name by clicking in the associated column cell and typing in the new name.



4. Click **Create**.

   The Output pane displays the underlying source code along with a CREATED notification.

5. Click **Close**.

   The new environment is now available in the left Navigator pane. (If you do not see it, click **Refresh** in the left pane.)

Create the MLE module that Imports FACTORIAL_MOD

6. In the MLE JS page, in the left pane, select **Modules** from the object type selector and then click **Object submenu** - **Create Object**.

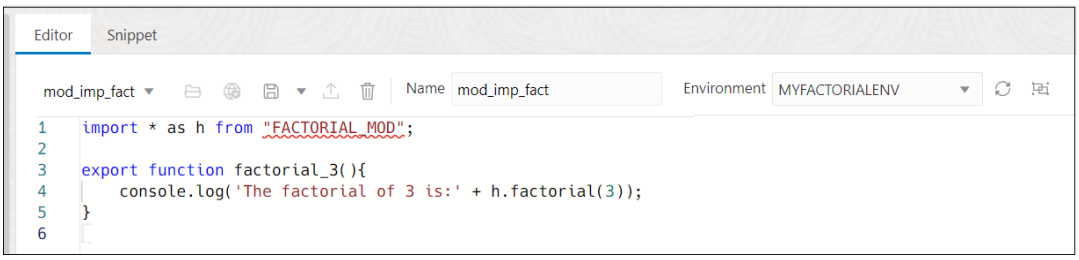
7. In the right pane, select the **Editor** tab, and enter the following JavaScript function:

```
import * as h from "FACTORIAL_MOD";

export function factorial_3(){
    console.log('The factorial of 3 is:' + h.factorial(3));
}
```

8. In the **Name** field, enter **mod_imp_fact**.

9. In the **Environment** field, select **MYFACTORIALENV**.



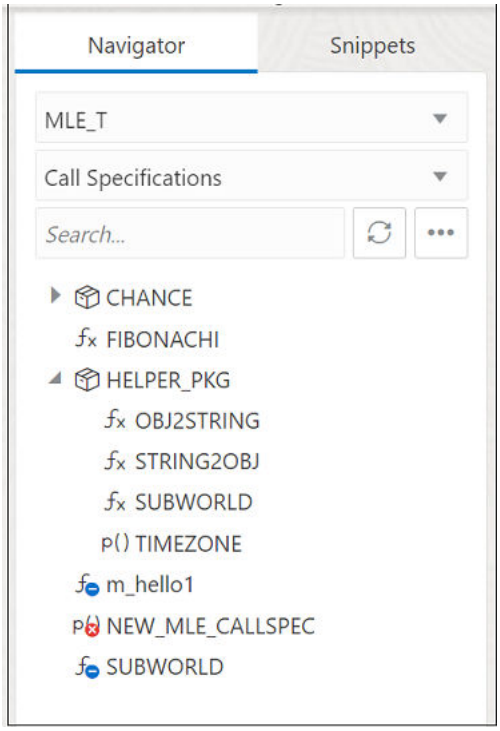
10. Click **Save Module**.

    A notification that the module is saved in the database is displayed.

8.4 Managing MLE Call Specifications

Functions exported by an MLE JavaScript module can be published by creating a call specification.

There are two types of call specifications: Module and Inlined (that is JavaScript code embedded in the DDL). Also, call specifications can be created as schema-level functions and procedures or they can be part of package declarations. Accordingly, the Navigator tab on the MLE JS page displays different icons to indicate a call specification in a package, module-based call specification and inlined call specification (with the blue overlay icon).



To create an inlined call specification, in the the **Navigator** tab, select **Object submenu** and click **Inline Call Spec**.

Creating a Call Specification for an MLE Module

To create a call specification for an MLE module:

1. In the left pane, in the **Navigator** tab, select **Object submenu** and click **Create Object**.

   The Create MLE Call Spec panel appears.
2. In the **Name** field, enter a name for the call specification.
3. Select the module and environment in the **Module Name** and **Environment Name** fields.

   When the module is selected, the functions are automatically populated in the Available Functions box.
4. Select **Procedure** or **Function**. If Function, select the return data type from the drop-down list.
5. Select the function from the list of available functions. When you select the function, the parameters are automatically prefilled in the **Parameters** field. If needed, you can edit the Direction and Type fields. When you click the parameter row, the **Parameter declaration** and **Signature** fields are automatically prefilled.
6. Click **Create**.

See Creating a Call Specification in the *Oracle Database JavaScript Developer's Guide* to know more about creating a call specification.

Context Menu Options

In the left pane, right-click the call specification object to view the following options:

- **Edit**: Select to edit the call specification. You can only edit the parameter-related fields.
- **Drop**: Select to remove the call specification from the database.
- **Compile**: Select to compile or recompile the call specification.

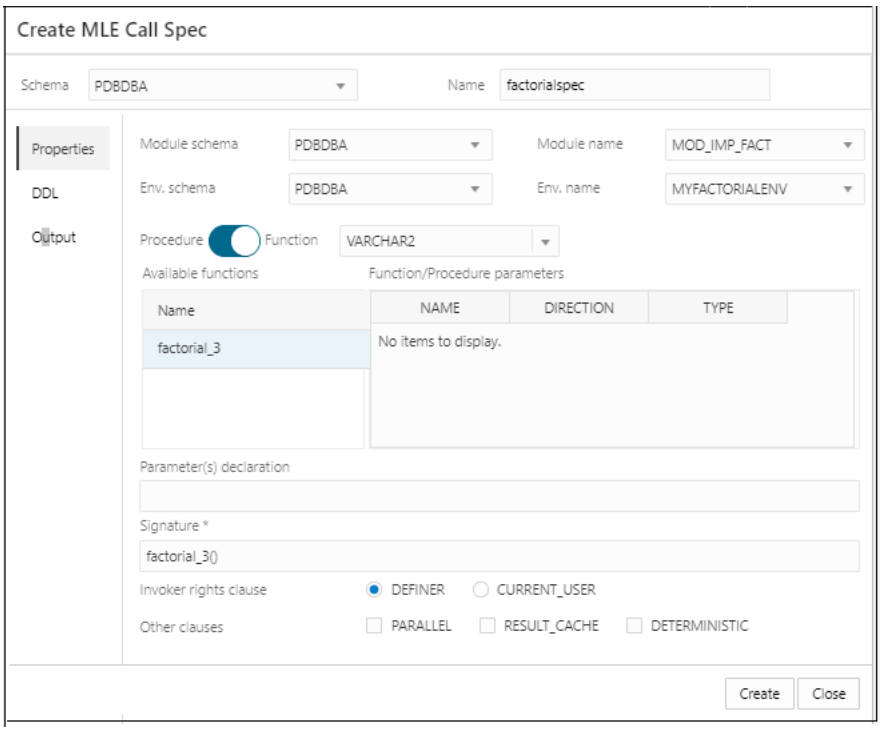
See Example: Creating an MLE Call Specification for an example of how to create a call specification.

8.4.1 Example: Creating an MLE Call Specification

This example demonstrates how to create a call specification for the `mod_imp_fact` module created in Example: Creating an MLE Environment.

1. In the MLE JS page, in the left pane, select **Modules** from the object type selector. From the list of modules displayed, select `mod_imp_fact`.
2. Right-click and select **Create**, and then select **Call Specification**.

   The Create MLE Call Spec panel appears.
3. In the **Name** field, enter `factorialspec`.
4. In the **Environment Schema** and **Enviornment Name** fields, enter your currrent schema name and select `myfactorialenv`.
5. Select Function, and then select **VARCHAR2** from the dropdown list.
6. In the **Available Functions** field, select the `greetings` function and the **Signature** field is automatically prefilled.



7. Click **Create**.

The Output pane displays the code for the call specification along with a function compiled notification.

8. Run the function from the SQL Worksheet page and you see the following output:

```
The factorial of 3 is: 6
```

8.5 Executing JavaScript Code Snippets and Debugging MLE Modules

You can add, edit, save and execute JavaScript code snippets using the Snippets tab in the right pane of the MLE JS page. Code snippets saved in the browser and file system are listed under the Snippet tab in the left pane.

JavaScript snippets can also be used to test and debug MLE modules using a debug specification. For more information about debug specifications, see Post-Execution Debugging in the *JavaScript Developer's Guide*.

> **Note:**
>
> SQL worksheet supports the execution of JavaScript code in Oracle Database release 21c. These saved JavaScript worksheets will also appear under the Snippets tab in the left pane.

Executing a JavaScript Code Snippet

To execute a code snippet:

1. In the right pane, select the **Snippet** tab and enter the JavaScript snippet.
   You can do this in one of two ways:
   - Type and enter the code, or copy and paste the code.
   - Click **Open File** in the toolbar to browse and open the file that contains the code.
2. Click the down arrow next to **Snippet** in the toolbar and select **Save As**. Enter a name for the code snippet and save.
3. If required (such as when testing MLE modules), select the preferred environment.
4. Click **Run Snippet**. The Output tab in the lower pane displays the results of the code execution.

Debugging a JavaScript MLE Module

To debug an MLE module:

1. Enter the code snippet, save it and select the environment. See steps 1-3 in **Executing a JavaScript Code Snippet**.
2. In the Snippet toolbar, from the Debug Specification menu, click **New Debug Specification**.

   The Create Debug Specification panel appears.
3. In the Create Debug Specification panel, do the following:
   a. In the right side of the panel, select the module to debug.

b. In the left side of the panel, select the JSON template you want: **Snapshot** or **Watch**. For more information about these templates, see Debugpoint Actions in the*JavaScript Developer's Guide*.

c. In the JSON template, the name of the MLE module to debug is specified using the `name` field and the location within the module where the debug information is to be collected is specified using the `line` field. To specify the condition that will trigger the debug action, use the `condition` field. See Example: Debugging a JavaScript Module.

d. Click **Create**,

4. In the MLS JS page, select the **Snippet** tab and in the **Debug Specification** field, select the created debug specification.

5. From the Snippet toolbar, click **Debug Snippet**.

   The debug information appears under the **Debug Console** tab in the lower pane.

See Example: Debugging a JavaScript Module for an example of how to debug a JavaScript MLE module.

8.5.1 Example: Debugging a JavaScript Module

This example demonstrates how to debug a JavaScript module.

Create the JavaScript Module

1. In the MLE JS page, in the right pane, select the **Editor** tab and enter the following JavaScript function:

```
export function fib( n ) {
    if ( n < 2 ) return 1;
    let n1 = { value : fib( n - 1 ), id : 'n1' };
    let n2 = { value : fib( n - 2 ), id : 'n2' };
    return n1.value + n2.value;
}
```

2. In the **Name** field, enter **fib_mod** and click **Save**.

   The created module appears in the left Navigator pane under Modules.

Create the MLE Environment

3. In the Navigator tab, from the object type selector, select **Environments**. Click **Object submenu**.

4. In the Create MLE Environment panel, do the following:

   a. In the **Name** field, enter **fib_env**.

   b. From the **Available Modules** field, select the module **FIB_MOD** and click **Add Selected Modules** to move the module to the **Imported Modules** field.

   c. Click **Create**.

   The new environment appears in the left pane under Environments.

Enter the Code Snippet

5. In the right pane, select the **Snippet** tab and enter the code snippet.

   Select the environment **fib_mod** to set the execution context. If the environment does not display, click **Reload Environments** and then expand the dropdown again.

Enter the following code:

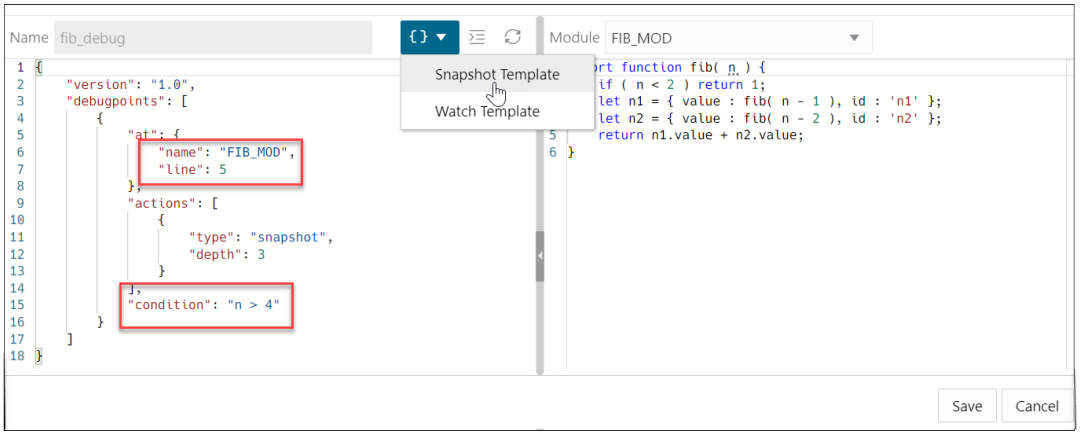
```
(async ()=>{
const aa = await import('FIB_MOD') ;
console.log(aa.fib(6));
})()
```

Click the dropdown in the toolbar, select **Save As** and enter a name for the snippet.

6. To execute the snippet, click **Run Snippet** and the output is displayed in the lower right pane under the Output tab.

Create the Debug Specification

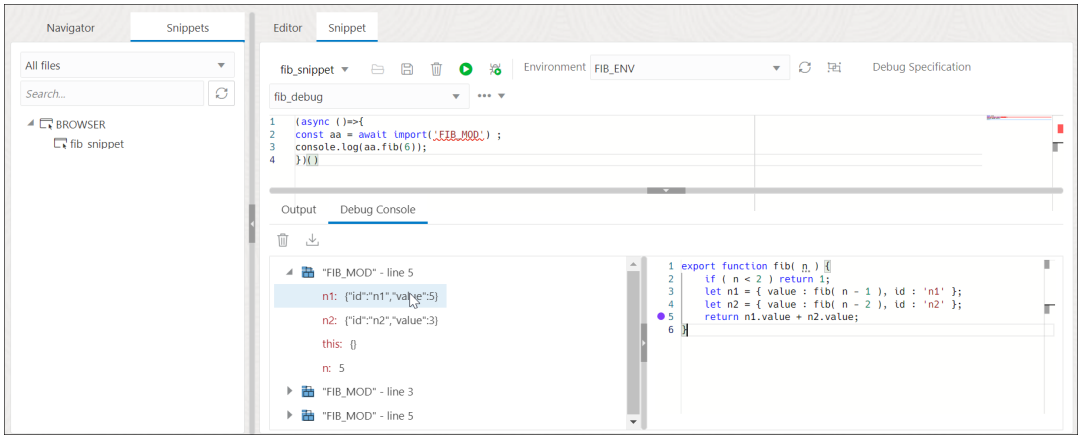
7. To create a new debug specification, in the toolbar, click the **Debug Specification Menu** and select **New Debug Specification**.
8. In the Create Debug Specification panel, do the following:
   a. Select **FIB_MOD** from the **Module** drop-down list.
   b. In the left side of the panel, select **Snapshot Template**.
   c. In the JSON document, edit the following:
      - **name**: Enter the module name.
      - **line**: Enter **5** to identify the position in the module code.
      - **condition**: Enter **n > 4**.
   d. Enter the name **fib_debug** for the debug specification.
   e. Click **Create**.



9. From the Snippet toolbar, click **Debug Snippet**.

   The output appears under the **Debug Console** tab in the lower pane.

   If you position the cursor over the debug points, the corresponding position in the module code is displayed.


Navigator
Snippets
All files
Search...
BROWSER
fib snippet
Editor
Snippet
fib_snippet
Environment
FIB_ENV
Debug Specification
fib_debug
(async ()=>{
const aa = await import('FIB_MOD') ;
console.log(aa.fib(6));
})()
Output
Debug Console
"FIB_MOD" - line 5
n1: {"id":"n1","value":5}
n2: {"id":"n2","value":3}
this: {}
n: 5
"FIB_MOD" - line 3
"FIB_MOD" - line 5
export function fib(n) {
if (n < 2) return 1;
let n1 = { value : fib(n - 1), id : 'n1' };
let n2 = { value : fib(n - 2), id : 'n2' };
return n1.value + n2.value;
}

9
The Scheduling Pages

> **Note:**
>
> Available for Oracle Database 12c release 2 and later releases.

Database Actions provides a graphical interface for using the DBMS_SCHEDULER PL/SQL package to work with Oracle Scheduler objects such as Jobs, Chains, Programs and Schedules. To use the scheduling features, you must first understand the concepts and essential tasks for job scheduling, which are explained in the chapters about Oracle Scheduler concepts and scheduling jobs in the *Oracle Database Administrator's Guide*.

The Scheduling functionality is focused on exploring Oracle Database Scheduler artifacts and provides details for jobs, chains, chain steps and rules, job execution history, programs, schedules, job classes, file watchers, windows and window groups. Depending on the access rights of the logged-in user, information is taken from `ALL_*` or `DBA_*` views.

> **Note:**
>
> SQL and PL/SQL editors are used in the user interfaces for Jobs and Programs. Some shortcut keys used in these editors are:
>
> - **Alt+F1**: Invokes accessibility options.
> - **Ctrl+M**: Alters the Tab key behavior, from inserting tab to going to the next control on the form.
>
> For more keyboard shortcuts, see Keyboard Shortcuts.

To navigate to the Scheduling pages, do either of the following:

- In the Launchpad page, select the **Development** tab and click **Scheduling**.
- Click **Selector** to display the navigation menu. Under Development, select **Scheduling**.

Topics

9.1 Overview

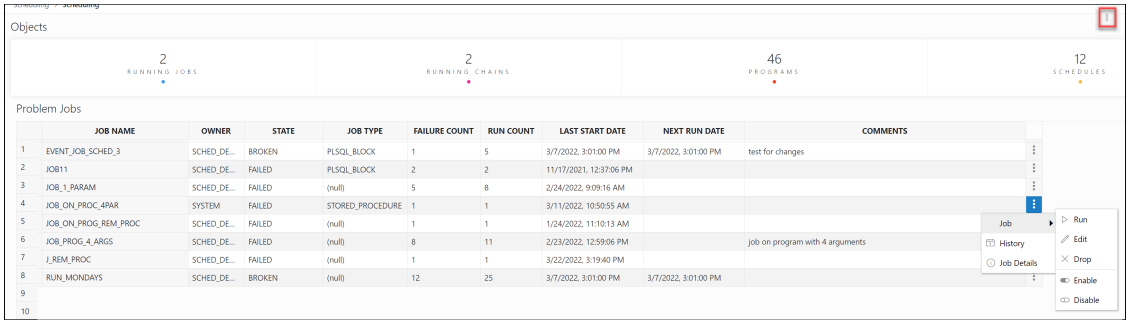
The Overview page provides an overview of the activity in the Scheduling pages. It consists of two parts: Objects and Problem Jobs

The Objects section provides an overview of the total number of running jobs, running chains, programs and schedules. Click an object tile to go to the corresponding page.

Click **Scheduler Summary** at the top right to view details such as Running Jobs, Current open window, Log history retain, Default timezone, Event expiry time, File watcher count, repeat interval, and mail server settings.

This Problem Jobs section provides information about jobs with problems in their execution. Listed in tabular format are jobs having one of the following statuses: Failed, Broken, Chain_Stalled, Blocked, Retry Scheduled, Resource_Unavailable. The **Actions** context menu provides links to the history and job details for each problem job. Other actions that you can perform are run, edit, drop, enable and disable job.

> **Note:**
>
> If there are no jobs with problems, the Jobs Summary page appears as the initial page for the Scheduling functionality.



9.2 Jobs

There are five options available in the Jobs menu: Summary, Running, Forecast, History, Notifications. Each option is described below.

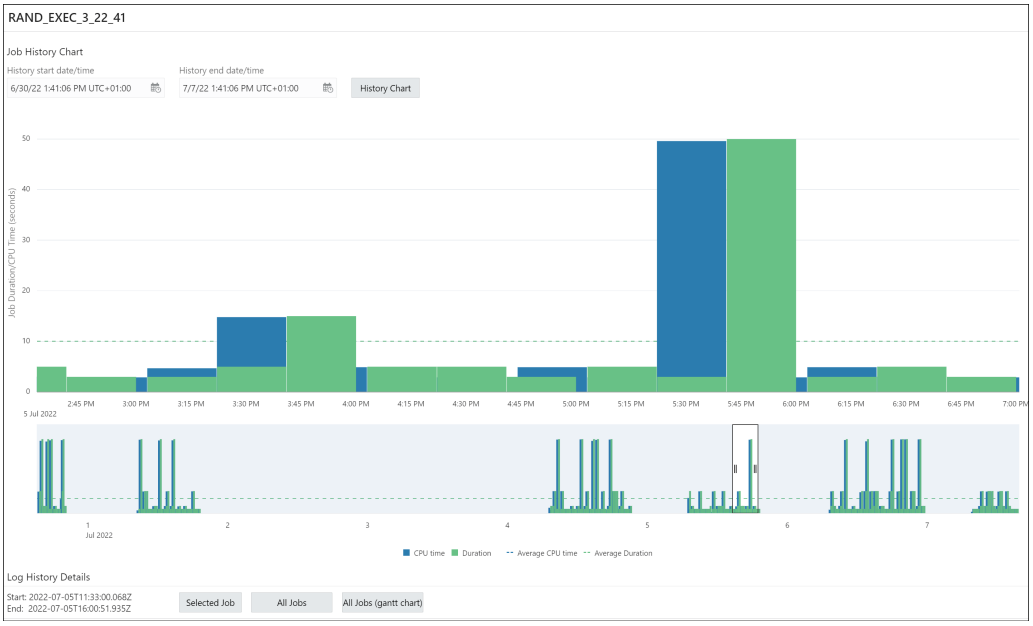
9.2.1 Summary

At the top, the Summary page provides an overview of the total number of Failed, Blocked, Chain stalled, Resource unavailable, Broken, and Retry Scheduled jobs. Click a job tile and the corresponding list of jobs are displayed in tabular format below. Remove the filters to display all scheduled jobs.

To create a job, see Create or Edit Job.

The Actions icon ⋮ is available at the end of each job row. Click **Actions** to view the following list of options:

- **Job**
  - **Run**: Runs the specific job.
  - **Edit**: See Create or Edit Job.
  - **Drop**: Drops the specific job.
  - **Job-Enable/Disable**: If enabled is selected, the job is picked up by the Scheduler for processing. The status of the job (enabled or not) is seen in the Job Details page, where the property “enabled” displays TRUE or FALSE.
- **History**
  - **Report**: Provides a history of the job runs in a report format, including log details, status of the run, run duration, errors if any, and so on.
  - **Chart**: Provides a history of the job runs in a visual bar chart representation. Run duration and used CPU time are presented for each execution. Jobs in chains are presented with aggregated information for the whole job. There is an overview scrollbar that enables zoom and scroll by changing the time frame. You can display all details from logs for that time frame for a selected job or for all jobs. Also, a Gantt chart can be shown for all jobs for a selected time frame.

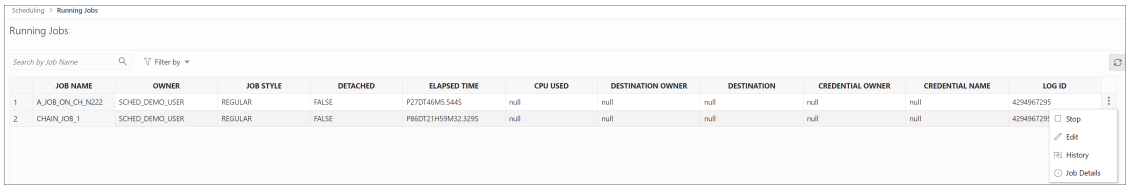


- **Job Forecast**: You can execute the job forecast functionality for a single job (available in the Action menu) or for a set of jobs (available on the toolbar for the table). For a set of jobs, filtering can be used to narrow the set of jobs. Use **Ctrl key+click** to select some jobs and then Job Forecast will run for the selected jobs only. If there are no selected jobs, then forecast will run for all listed jobs. Not every job included in the set is included in the final forecast. Only jobs with a defined calendar (repeat interval) are included. You can define the calendar inline or using schedule, window or window group. After the forecast is done, you can select different zoom levels and filter the results by schema.
- **Job Details**: Displays job attributes, such as action, job class, type, schedule, and so on. Select **JSON** to view the attributes in JSON format. Depending on the job’s definition, details about used objects are provided such as used program, PL/SQL code, procedure, procedure dependencies, program arguments, job arguments, job class, schedule,

window, window group together with windows in the group, file watcher, history chart presenting one week of history from last start date.

9.2.2 Running

The Running page displays the list of currently running jobs. The following commands are available: Stop, Edit, History, Job Details.



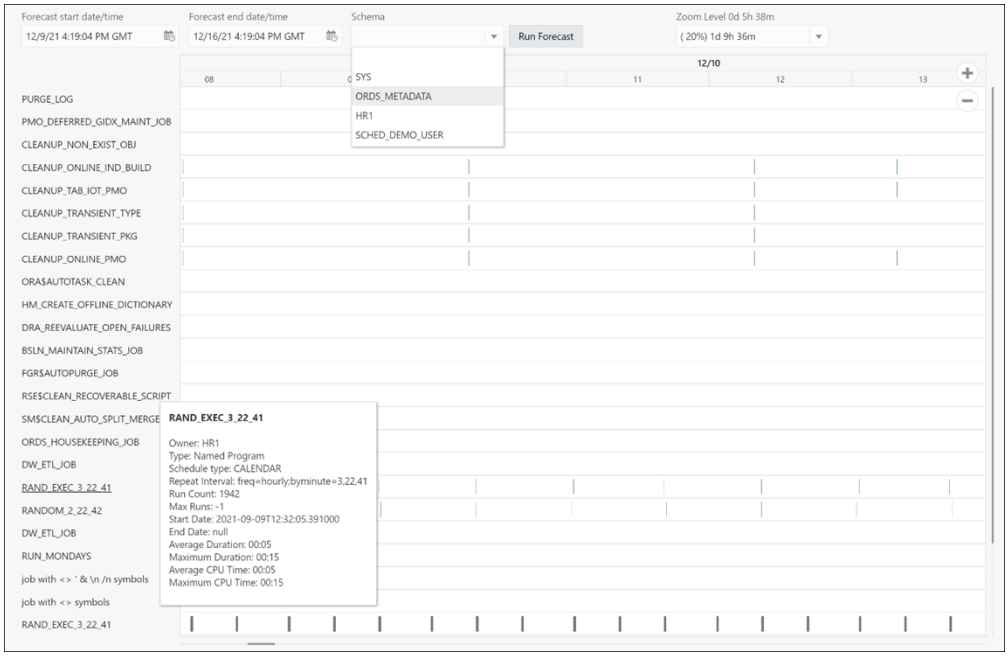
9.2.3 History

The History page displays log run details for all Scheduler jobs available to the user. You can use the History window and set filters to limit the amount of data. You can filter using delayed jobs by providing a delay interval. Ordering is supported on the grid by clicking the column header.

9.2.4 Forecast

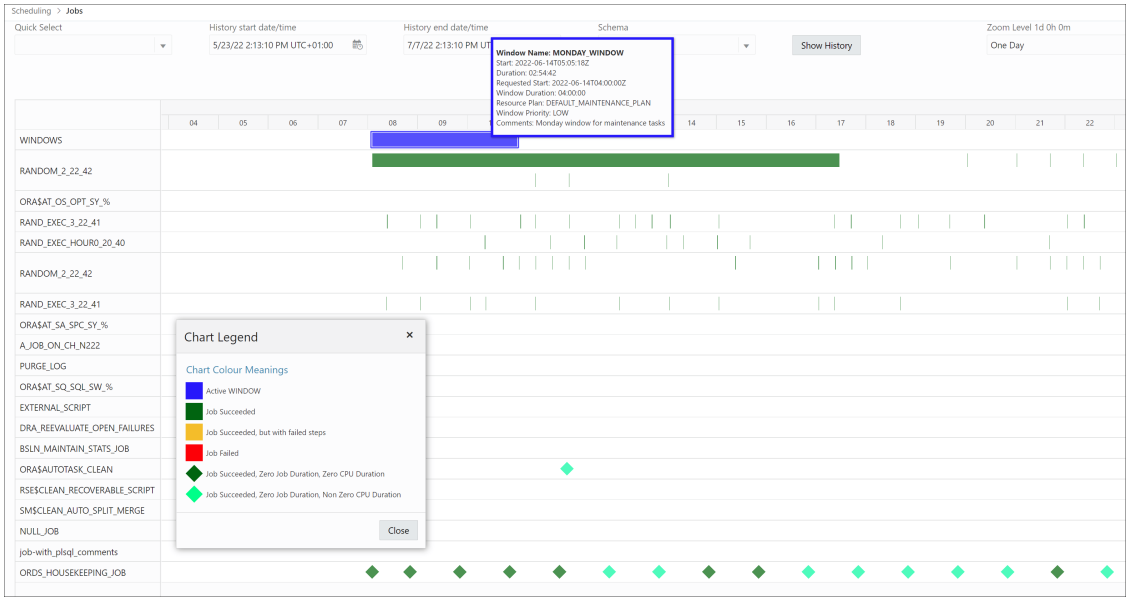
The Forecast page provides the job execution forecast for all available jobs. This operation takes time to finish depending on the number of jobs and forecast interval. Forecast for all available jobs depends on the rights of the connecting user. After execution, you can filter the results based on the schema of the jobs.

The Forecase functionality is also available on the Jobs - Summary page. In that case, it works on the list of available jobs. You can filter the jobs before the forecast functionality is used. If there is a selection of jobs, then the functionality uses only the selected jobs.



9.2.5 History (Gantt Chart)

History represents windows and job execution history in the form of a Gantt chart. The windows activation history is shown on the first row. Jobs are ordered in a descending order based on the maximum used CPU time. Details are shown for each window activation and job execution. The job summary is available when hovering over the label of a job's row.



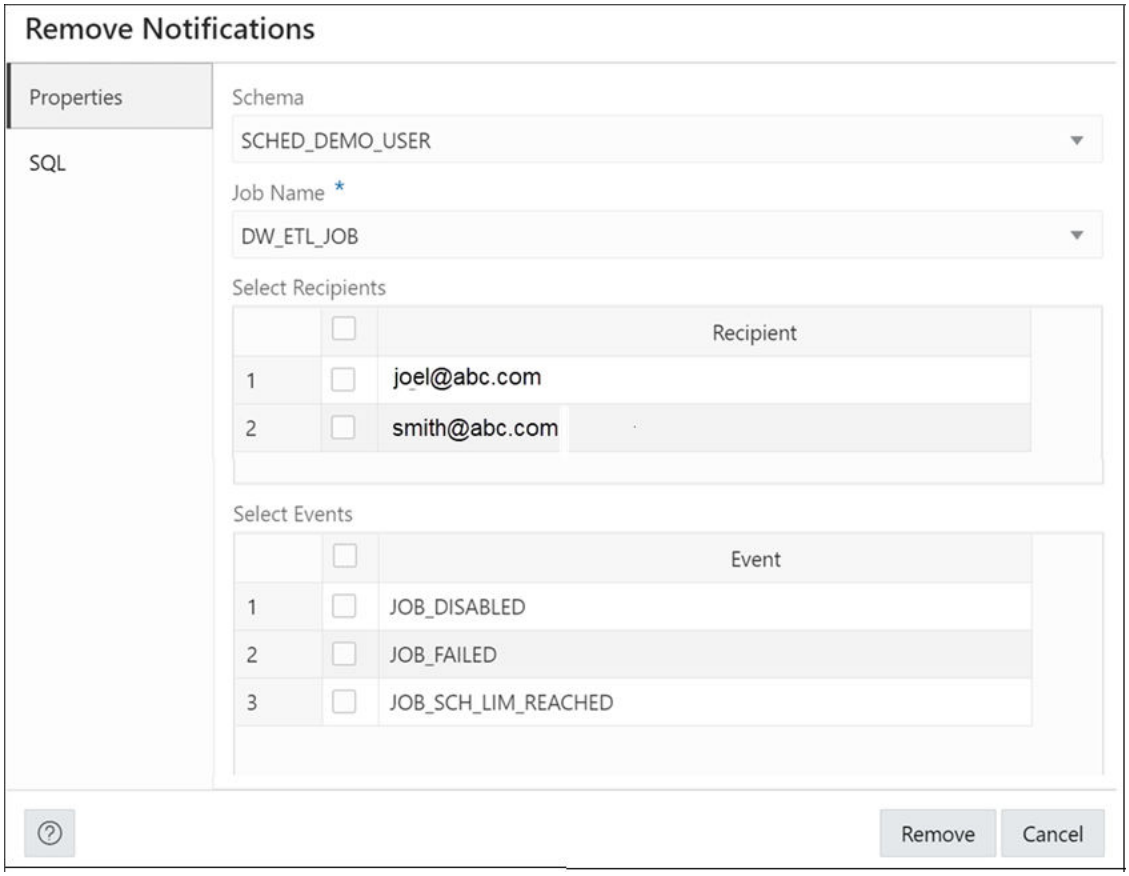
9.2.6 Notifications

You can create notifications only if an email server is set for the scheduler.

The Notifications page enables you to view, create, edit and delete notifications related to job events. Also, you can see email server details by clicking the **Notifications Email Server** icon at the top right of the page.

For each message, you can specify job, recipient email addresses and sender (or no sender), and you can modify the subject and body of the message and set the filter condition. If multiple recipients and events are provided, then Oracle Scheduler creates a separate notification record for each combination <recipient,event>. You can edit the content of a set of notifications created in this way by using **Edit Aggregated**, which appears in the context menu of each notification. **Edit** edits the content of a single notification.

Notifications can be filtered by job name, job owners, recipients and events. You can select and remove some notifications by clicking the **Remove Notifications** icon. If there are no selected notifications (use **Ctrl+click** to deselect a single notification), then the Remove Notifications dialog appears enabling removal of all notifications for the selected job, or job and recipients, or job and events, or job and events and recipients.



9.2.7 Create or Edit Job

This section describes how to create a new Oracle Scheduler job or edit an existing job.

To create a job, Database Actions internally uses the DBMS_SCHEDULER.CREATE_JOB procedure, which is documented in *Oracle Database PL/SQL Packages and Types Reference*.

1. In the Jobs page, at the top right, click **Create Job**.

2. In Job Properties, enter the following fields:

Details

- **Enabled**: If this option is specified, validity checks are made and the job is created enabled if all the checks are successful. If this option is not specified, the job is not created enabled.
- **Name**: Name of the job.
- **Description**: Optional text string that can be used to describe the job.
- **Type**: Type of object to be executed by the job: PL/SQL Block, Chain, Stored Procedure, Named Program, or Script. Additional controls appear for Chain, Stored Procedure and Named Program enabling you to select related objects.

  For Stored Procedure, only procedures with IN parameters are listed, procedures with IN OUT or OUT parameters are not permitted.

  Schema level and package level procedures are listed for the selected schema but you can directly type the procedure name (or package_name.procedure_name) in the field.
- **Class**: Name of the job class to which this job belongs.

Execution Mode

- **Mode**: When to execute the job: **Immediate** (immediately on creation, and once only), **Once** (once, at a specified time), **Repeating**, **Queue**, **File Watcher**, **Schedule** (using a named schedule object), **Window** and **Window group**. If you specify anything other than Immediate, you are prompted for additional information.
  For Repeating, you can manually define the repeat interval or click the pencil icon to select the date, frequency, weekday, and interval values. Some clauses of Oracle calendar syntax (include, exclude, intersect, periods and by period) are not supported and a warning is displayed when the edit icon is clicked.

Destination

- **Local** (local system), **Remote** (the database destination for a remote database job, or external destination for a remote external job), or **Multiple** (the job runs on all destinations associated with the provided destination group). Depending on what destination you selected for the job, select the local credential, the remote credential and destination, or the destination group.

Properties

- **Auto Drop**: Determines whether the job is to be automatically dropped after it has completed or has been automatically disabled.
- **Restart on failure**: Determines whether the job can be restarted in case of failure.
- **Restart on recovery**: Determines whether to restart the job in case of database failure.
- **Store Output**: If enabled, then for job runs that are logged, all job output and error messages are stored in the *_JOB_RUN_DETAILS views. If disabled, then the output and messages are not stored.
- **Follow Default Time Zone**: Determines whether if the job start date is null, then when the default time zone scheduler attribute is changed, the Scheduler recomputes the next run date and time for this job so that it is in accordance with the new time zone.
- **Allow Runs in Restricted Mode**: If enabled, the job is permitted to run when the database is in restricted mode, provided that the job owner is permitted to log in during this mode.

- **Stop on Window Close**: If the schedule of a job is a window or a window group, enabling this option causes the job to stop once the associated window is closed, and disabling causes the job to continue after the window closes. (Note that if the job is allowed to continue, its resource allocation will probably change because closing a window generally also implies a change in resource plans.)
- **Instance Stickiness**: This attribute should only be used for a database running in an Oracle Real Application Clusters (Oracle RAC) environment. By default, it is enabled. Jobs start running on the instance with the lightest load and the Scheduler thereafter attempts to run on the instance that it last ran on. If that instance is either down or so overloaded that it does not start new jobs for a significant period of time, another instance runs the job. If the interval between runs is large, instance_stickiness is ignored and the job is handled as if it were a non-sticky job. If instance_stickiness is disabled, each instance of the job runs on the first instance available.
- **Parallel Instances**: For an event-based job, determines what happens if an event is raised and the event-based job that processes that event is already running. If disabled, it causes the new event to be ignored. If enabled, it causes an instance of the job to be started for every instance of the event, and each job instance is a lightweight job so multiple instances of the same event-based job can run in parallel.
- **Job Style**: Style of the job being created: **REGULAR** (regular job) or **LIGHTWEIGHT** (lightweight job). A lightweight must reference a program object. Use lightweight jobs when you have many short-duration jobs that run frequently. Under certain circumstances, using lightweight jobs can deliver a small performance gain.
- **Job Priority**: The priority of this job relative to other jobs in the same class as this job. If multiple jobs within a class are scheduled to be executed at the same time, the job priority determines the order in which jobs from that class are picked up for execution by the job coordinator. It can be a value from 1 through 5, with 1 being the first to be picked up for job execution.
- **Logging Level**: Determines how much information is logged: DBMS_SCHEDULER.LOGGING_OFF (no logging), DBMS_SCHEDULER.LOGGING_FAILED_RUNS (only jobs that failed, with the reason for failure), DBMS_SCHEDULER.LOGGING_RUNS (all runs of each job in this class), or DBMS_SCHEDULER.LOGGING_FULL (all operations performed on all jobs).

  However, if the job class has a higher (more detailed) logging level than the level specified for the job, the job class logging level is used.
- **Instance ID**: In an Oracle Real Application Clusters environment., the instance ID of the instance that the job must run on.
- **Max Runs**: The maximum number of consecutive scheduled runs of the job.
- **Max Failures**: The number of times a job can fail on consecutive scheduled runs before it is automatically disabled.
- **Raise Events**: Determines at what stages of the job execution to raise events.
- **Max Run Duration**: Maximum amount of time that the job should be allowed to run. Its data type is INTERVAL DAY TO SECOND. If this attribute is set to a nonzero and non-null value, and job duration exceeds this value, the Scheduler raises an event of type JOB_OVER_MAX_DUR. It is then up to your event handler to decide whether or not to allow the job to continue.
- **Schedule Limit**: Maximum delay time between scheduled and actual job start before a program run is canceled.
- **Reset to Defaults**: Resets all properties to their default values.

NLS

Enables you to set the NLS-related property required for this job. To enter a value, type in to the related field for the Value column.

For a new job, parameters are taken from the database session.

3. In the **DDL** pane, you can review and save the SQL statements that are generated when creating or editing the job.
   - For a new job, click **CREATE** to view the generated DDL statements.
   - When you edit a job, click **UPDATE** to view the generated ALTER statements.

   When you are finished, click **Apply**.

   The **Output** pane displays the results of the DDL commands. If there are any errors, go to the corresponding pane, fix the errors, and run the commands again. You can save the content to a file.

9.3 Chains

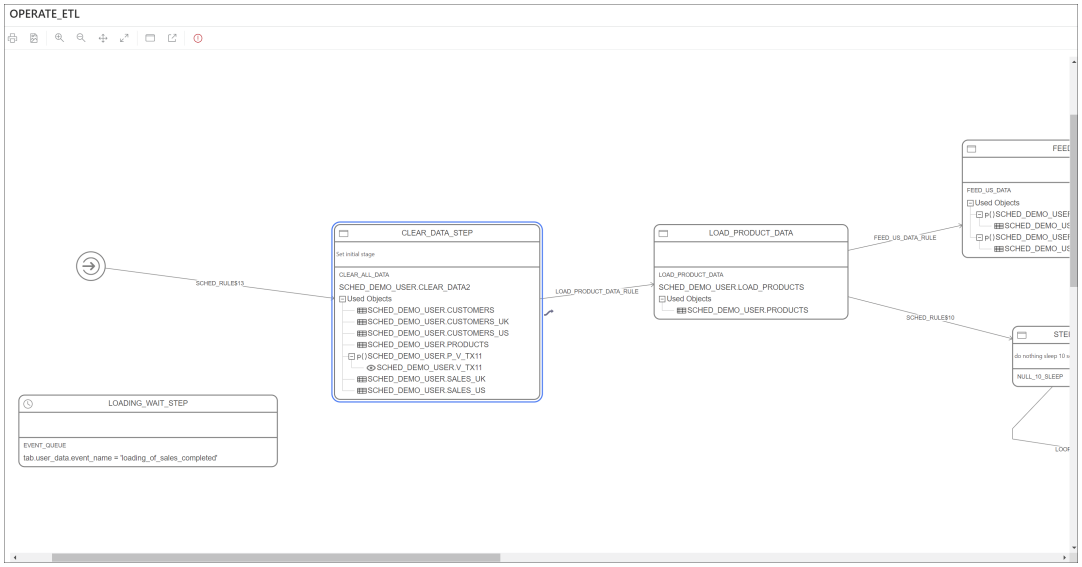
There are four options available in the Chains menu: Summary, Running Chains, Steps, Rules. Each option is described below.

Summary

The Summary page displays all the chains owned by the current user. The details are presented in a tabular format. At the end of each row is the Actions icon . Click **Actions** to view the list of options for the chain. Click a chain in the table and details about steps and rules appear below the Chains table. There is a **Drop Step** or **Drop Rule** option available in the context menu for a step or rule.

The actions available for a chain are:

- **Run**: Runs the specific chain.
- **Add Step**: Creates a new step for the selected chain.
- **Add Rule**: Creates a new rule for the selected chain.
- **Edit**: Edits the chain properties.
- **Drop**: Drops the chain
- **Chain-Enable/Disable**: Enables or disables the chain.
- **Show Diagram**: Displays the job steps of a chain in a visual diagram format.



Double-click a step or rule in the diagram to see more details. You can directly create steps and rules on the diagram using the **Add Step** and **Add Rule** icons.

The following dependencies are shown on each step:

– Event schedule: The name of the schedule.

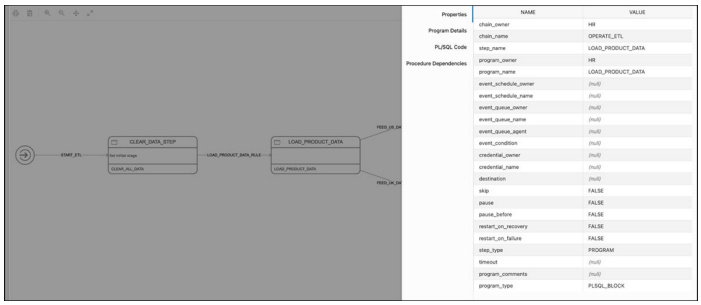
– Event queue: The name of the queue and condition.

– Program using procedure or PL/SQL block: Tables, views, procedures functions and packages used - two levels of dependencies.

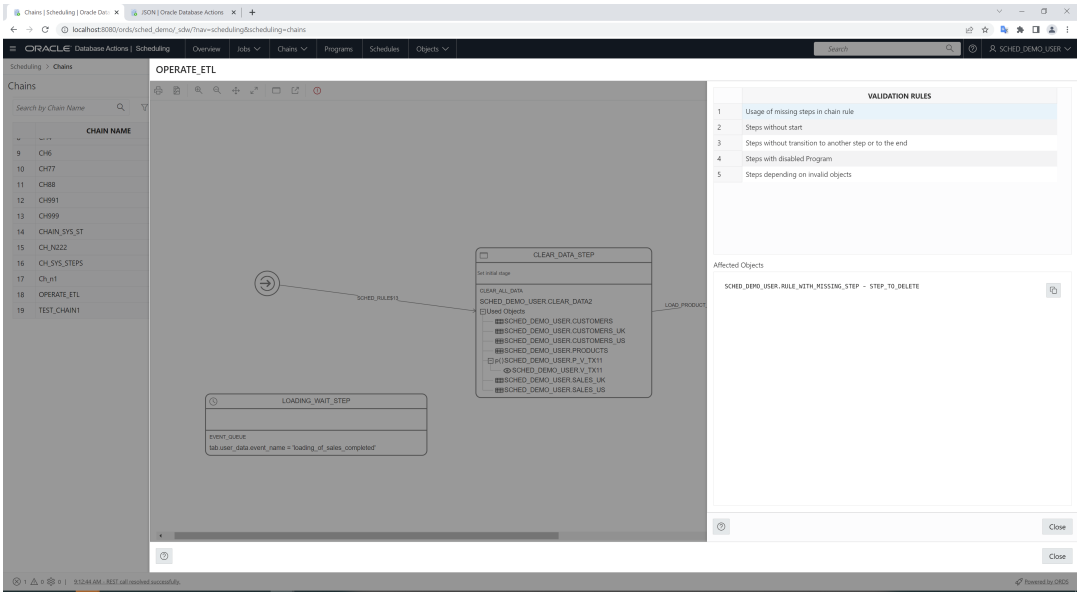
You can also create a rule by connecting two steps:

– Click on a step and the link icon appears.

– Click the link icon and drag to the next step. The Rule dialog appears where you can provide details.



There are validation rules that are applied to chain details and if problems are found, an icon appears on the menu bar that enables you to get details about the rules violated.



The following rules are applied:

- Usage of missing steps in chain rule
- Steps without start: There is no rule that starts the step.
- Steps without transition to another step or to the end: The step does not appear in the condition part of any rule.
- Steps with disabled Program: of type “Warning” – show steps using disabled programs
- Steps depending on invalid objects: of type “Warning” – show steps that use program dependent on invalid objects.

- **Analyze Chain**: Displays output from the `DBMS_SCHEDULER.ANALYZE_CHAIN` procedure in text format. Lists the type of steps, dependencies and transitions between them based on defined rules.

When a chain row is selected, the lists with steps and rules appear below. The Edit and Drop commands are available in the Actions menu for Steps and Rules.

Running Chains

The Running Chains page displays the execution of the steps of the currently running chains.

The following commands are available for each row in the report:

- For Chain, the commands are Edit, Drop, Enable, Disable, and Evaluate Chain.

  Evaluate Chain invokes the DBMS_SCHEDULER.EVALUATE_RUNNING_CHAIN procedure. It forces the reevaluation of the rules of a running chain to trigger any rules for which the conditions have been satisfied.

- Other chain-related command are Show Diagram, Analyze Chain, Step Details, and Edit Step.

Steps

The Steps page provides details of each step in all chains.

Rules

The Rules page provides details of each rule in all chains.

9.3.1 Create or Edit Chain

This section describes how to create or edit a chain.

To create a chain, Database Actions internally uses the DBMS_SCHEDULER.CREATE_CHAIN procedure.

1. In the Chains page, at the top right, click **Create Chain**.
2. In Properties, enter the following fields:
   - **Name**: Name to assign to the chain.
   - **Evaluation Interval**: **NULL** reevaluates of the rules of a running chain only when the job starts and when a step completes. A non-NULL value causes rule evaluations to also occur periodically at the specified interval.
   - **Enabled**: Enables the chain. (Causes the DBMS_SCHEDULER.ENABLE procedure to be called after the chain is created.)
3. Click **Create** to create the chain.

   Click **Create and Open** to create the chain and open in Diagram mode.

9.3.2 Add or Edit Step

This section describes how to add a step for a chain.

1. In the context menu for a chain, select **Add Step**.
2. In the Create Step slider, enter the following fields:
   - **Step Name**: Name of the step.
   - **Type**: Select **Program**, **Chain**, **Event** or **Queue** to run during the step. Additionally, select the name of the program, chain, event or queue in the corresponding field.
   - **Pause**: Select this option to pause the step.
   - **Skip**: Select this option to skip the step.
   - **Restart on Failure**: Select this option to restart the step after database failure.
   - **Restart on Recovery**: Select this option to restart the step after database recovery.
3. Click **Create**.

   The new step is displayed for the chain in the main Chains page.

9.3.3 Add or Edit Rule

This section decribes how to add a rule for a chain.

1. In the context menu for a chain, select **Add Rule**.
2. In the Create Rule slider, enter the following fields:
   - **Rule Name**: Name of the rule being created.

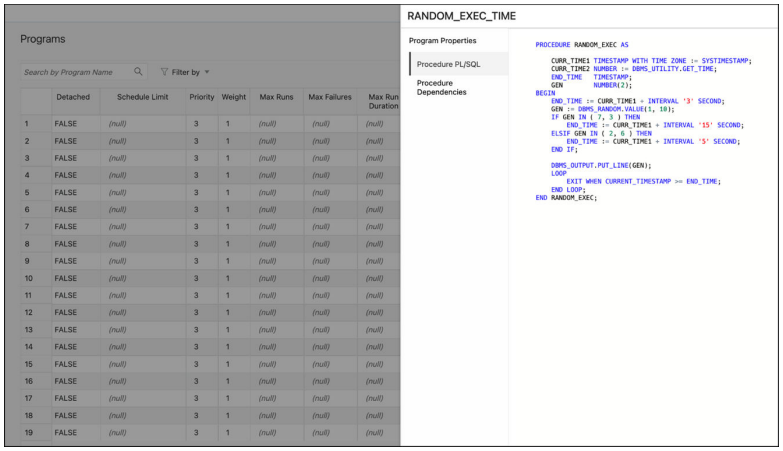
- **Condition**: Enter the condition to evaluate.
- **Action**: Action to perform if condition is met.
- **Comments**: Optional comments describing the rule.

> **See Also:**
>
> DEFINE_CHAIN_RULE Procedure in *Oracle Database PL/SQL Packages and Types Reference*

9.4 Programs

The Programs page displays the list of created programs.

The Actions menu for a program consists of the following options: Edit, Drop, Enable or Disable the program and View Program Details, including used procedure and procedure dependencies.



9.4.1 Create or Edit Program

This section describes how to create or edit an Oracle Scheduler program.

To create a program, Database Actions internally uses the DBMS_SCHEDULER.CREATE_PROGRAM procedure.

1. In the Programs page, at the top right, click **Create Program**.
2. In Program Properties, enter the following fields:

   Details tab

   - **Name**: Name of the program. The name has to be unique in the SQL namespace. For example, a program cannot have the same name as a table in a schema.
   - **Enabled**: If this option is specified, validity checks will be made and the program will be created enabled if all the checks are successful. If this option is not specified, the program is not created enabled.

- **Description**: Optional text string that can be used to describe the program.
- **Type**
  - **PL/SQL Block**: The program is a PL/SQL block. Job or program arguments are not supported when the job or program type is PLSQL_BLOCK. In this case, the number of arguments must be 0. Enter or paste in the complete PL/SQL code, or edit the existing code.
  - **Stored Procedure**: The program is a PL/SQL or Java stored procedure, or an external C subprogram. Only procedures, not functions with return values, are supported. PL/SQL procedures with IN OUT or OUT arguments are not supported.

    Schema: Schema of the stored procedure. If not specified, the schema of the job is assumed.

    Procedure: Name of the stored procedure.

    Arguments: For each argument, name, data type, default value, and whether it is an input, output, or input/output argument.
  - **Script**: The program is a SQL Script (SQL*Plus statements), Backup Script (RMAN commands), or External Script (operating system commands). Enter or paste the script text in the box.

Properties tab

Enables you to set program properties. For most properties the default is null, but you can check the box to specify a value.

- **Detached**: Enabled if the program is a detached job. Use a detached job to start a script or application that runs in a separate process, independently and asynchronously to the Scheduler. A detached job typically starts another process and then exits. Upon exit (when the job action is completed) a detached job remains in the running state. The running state indicates that the asynchronous process that the job started is still active. When the asynchronous process finishes its work, it must connect to the database and call DBMS_SCHEDULER.END_DETACHED_JOB_RUN, which ends the job.
- **Max Runs**: Maximum number of runs before the program is marked as completed.
- **Max Failures**: Maximum number of failures tolerated before the program is marked as broken.
- **Max Run Duration**: Maximum run duration of the program.
- **Schedule Limit**: Maximum delay time between scheduled and actual job start before a program run is canceled.

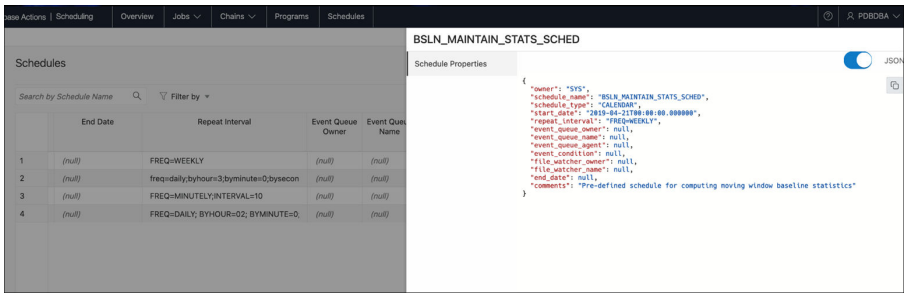
3. In the **DDL** pane, you can review and save the SQL statements that are generated.
   - For a new program, click **CREATE** to view the generated DDL statements.
   - When you edit a program, click **UPDATE** to view the generated ALTER statements.

   When you are finished, click **Apply**.
4. The **Output** pane displays the results of the DDL commands. If there are any errors, go to the corresponding pane, fix the errors, and run the commands again. You can save to a text file or clear the output.

9.5 Schedules

The Schedules page displays the list of available schedules.

The Actions menu for a schedule has the following options: Viewing schedule details, Editing or Dropping the schedule.



9.5.1 Create or Edit Schedule

This section describes how to create or edit Oracle Scheduler schedule.

To create a schedule, Database Actions internally uses the DBMS_SCHEDULER.CREATE_SCHEDULE procedure.

1. In the Schedules page, at the top right, click **Create Schedule**.
2. In Program Properties, enter the following fields:

   Properties tab

   - **Name**: Name of the schedule. The name has to be unique in the SQL namespace. For example, a schedule cannot have the same name as a table in a schema.
   - **Description**: Optional text string that can be used to describe the schedule.
   - **Mode**: Specify when jobs that use this schedule are to run: **Repeating** (specify the repeat interval, start date, and end date), **Queue** (specify the queue name, agent, condition, start date, and end date), or **File Watcher** (specify the file watcher object name, condition, start date, and end date).
3. In the **DDL** pane, you can review and save the SQL statements that are generated.
   - For a new schedule, click **CREATE** to view the generated DDL statements.
   - When you edit a schedule, click **UPDATE** to view the generated ALTER statements.

   When you are finished, click **Apply**.
4. The **Output** pane displays the results of the DDL commands. If there are any errors, go to the corresponding pane, fix the errors, and run the commands again. You can save to a text file or clear the output.

9.6 Objects

The Objects menu in Scheduling provides links to the following four pages: Job Classes, File Watchers, Windows and Window Groups

Job Classes
A job class is an Oracle Scheduler object that enables the Scheduler administrator to group jobs for logical purposes, such as to assign the same set of attribute values to member jobs, to set service affinity for member jobs, to set resource allocation for member jobs, or to group jobs for prioritization.

> **Note:**
>
> Creating a job class requires the MANAGE SCHEDULER system privilege. Access to SYS.DBA_RSRC_CONSUMER_GROUPS and SYS.DBA_SERVICES dictionary views is required in order UI to provide all details and choices.

The Job Classes page displays the job classes created in a tabular format. You can perform the following actions:

- **Create**: To create a job class, see Create Job Class.
- **Edit**: To edit a job class, select **Edit** from the Actions menu at the end of a row. For a description of the fields, see Create Job Class.
- **Drop**: To drop a job class, select **Drop** from the Actions menu.

File Watchers
A file watcher is an Oracle Scheduler object that defines the location, name, and other properties of a file whose arrival on a system causes the Scheduler to start a job. You create a file watcher and then create any number of event-based jobs or event schedules that reference the file watcher. When the file watcher detects the arrival of the designated file, it raises a file arrival event. The job started by the file arrival event can retrieve the event message to learn about the newly arrived file.

The File Watchers page displays the file watchers created in a tabular format. You can perform the following actions:

- **Create**: To create a file watcher, see Create File Watcher.
- **Edit**: To edit a file watcher, select **Edit** from the Actions menu at the end of a row. For a description of the fields, see Create File Watcher.
- **Drop**: To drop a file watcher, select **Drop** from the Actions menu.
- **Show DDL**: Displays the Data Definition Language statements for the file watcher. To show DDL, select this option from the Actions menu.

Windows
A window is an Oracle Scheduler object that can be used to automatically start jobs or to change resource allocation among jobs during various time periods of the day, week, and so on.

> **Note:**
>
> Creating a window object requires the MANAGE SCHEDULER system privilege. Access to SYS.DBA_RSRC_PLANS dictionary view is required to provide all details and choices.

The Windows page displays windows created in a tabular format. You can perform the following actions:

- **Create**: To create a window, see Create Window.
- **Edit**: To edit a window, select **Edit** from the Actions menu at the end of a row. For a description of the fields, see Create Window.
- **Drop**: To drop a window, select **Drop** from the Actions menu.

- **Show DDL**: Displays the Data Definition Language statements for the window. To show DDL, select this option from the Actions menu.

Window Groups
A window group is an Oracle Scheduler object that is a list of Oracle Scheduler Windows. Scheduler jobs that are scheduled to be run in a window group will be activated in that time span and using that resource plan for all windows in the group

The Window Groups page displays windows created in a tabular format. You can perform the following actions:

- **Create**: To create a window group, see Create Window Group.
- **Edit**: To edit a window group, select **Edit** from the Actions menu at the end of a row. For a description of the fields, see Create Window Group.
- **Drop**: To drop a window group, select **Drop** from the Actions menu.
- **Show DDL**: Displays the Data Definition Language statements for the window group. To show DDL, select this option from the Actions menu.

9.6.1 Create Job Class

To create a job class:

1. In the Job Class page, click **Create Job Class**.
2. Enter the following fields:
   - **Name**: Name of the job class.
   - **Description**: Optional text string that can be used to describe the job class.
   - **Logging Level**: Specifies how much information is written to the job log:
     - **RUNS**: Detailed information for all runs of each job in this class.
     - **FULL**: RUNS plus all operations performed on all jobs in this class.
     - **OFF**: No logging.
   - **Log Retention Period (days)**: Number of days that job log entries for jobs in this class are retained. The range of valid values is 0 through 999. If set to 0, no history is kept. If NULL (the default), retention days are set by the `log_history` Scheduler attribute.
   - **Service Name**: The database service that the jobs in this class will have affinity to. If no service is specified, the job class will belong to the default service, which means it will have no service affinity and any one of the database instances within the cluster might run the job.
   - **Resource Consumer Group**: Resource consumer group this class is associated with. If no resource consumer group is specified, the job class is associated with the default resource consumer group.
3. Click **Create**.

9.6.2 Create File Watcher

To create a file watcher:

1. In the File Watchers page, click **Create File Watcher**.
2. Enter the following fields:

- **Enabled**: Enables the file watcher. (Causes the DBMS_SCHEDULER.ENABLE procedure to be called after the file watcher is created.)
- **Name**: Name of the file watcher.
- **Description**: Optional descriptive text.
- **Destination**: Name of an external destination. You create an external destination by registering a remote Scheduler agent with the database.The view ALL_SCHEDULER_EXTERNAL_DESTS lists valid external destination names. If this parameter is null, the file watcher is created on the local host.
- **Credential Name**: Name of a valid credential object. The file watcher uses the credential to authenticate itself with the host operating system to access the watched-for file. The file watcher owner must have EXECUTE privileges on the credential.
- **Directory Path**: Directory in which the file is expected to arrive. The single wildcard '?' at the beginning of the path denotes the Oracle home path. For example, '?/rdbms/log' denotes the rdbms/log subdirectory of the Oracle home directory.
- **File Name**: Name of the file to look for. Two wildcards are permitted anywhere in the file name: '?' denotes any single character, and '*' denotes zero or more characters.
- **Min File Size**: Minimum size in bytes that the file must be before the file watcher considers the file found.
- **Steady State Duration**: Minimum time interval that the file must remain unchanged before the file watcher considers the file found. Cannot exceed one hour. If null, an internal value is used. The minimum value is 10 seconds. Oracle recommends similar steady state duration values for all file watchers for efficient file watcher job operation. Also, the repeat interval of the file watcher schedule must be equal or greater than the steady state duration value.

3. Click **Create**.

9.6.3 Create Window

To create a window:

1. In the Windows page, click **Create Window**.
2. Enter the following fields:
   - **Name**: Name of the window.
   - **Description**: Optional descriptive text.
   - **Enabled**: Enables the window. (Causes the DBMS_SCHEDULER.ENABLE procedure to be called after the window is created.)
   - **Resource Plan**: The resource plan that automatically activates when the window opens. When the window closes, the system switches to the appropriate resource plan, which is usually the plan that was in effect before the window opened, but can also be the plan of a different window. Only one resource plan can be associated with a window. It null, the resource plan in effect when the window opens stays in effect for the duration of the window. If an empty string, the resource manager is disabled for the duration of the window. If the window is open and the resource plan is dropped, then the resource allocation for the duration of the window is not affected.
   - **Duration**: The length of time that the window stays open. Can range from one minute to 99 days.

- **Priority**: Relevant when two windows overlap. Because only one window can be in effect at one time, the window priority determines which window opens. The two possible values for this attribute are `HIGH` and `LOW`. A high priority window has precedence over a low priority window, therefore, the low priority window does not open if it overlaps a high priority window.
- **Execution Mode**: `Repeating` or `Schedule`:
  - For `Repeating`, specify the **Repeat Interval** and optionally the **Start Date** and **End Date**.
  - For `Schedule`, select the name of the **Schedule** to be used.

3. Click **Create**.

9.6.4 Create Window Group

To create a window group:

1. In the Window Groups page, click **Create Window Group**.
2. Enter the following fields:
   - **Name**: Name of the window group.
   - **Description**: Optional comments.
   - **Enabled**: Enables the window group. (Causes the DBMS_SCHEDULER.ENABLE procedure to be called after the window group is created.)
   - **Available Windows**: Lists all Scheduler windows.
   - **Selected Windows**: List windows to be added to the window group. Use the arrow icons to move selected windows or all windows from one list to the other.
3. Click **Create**.

Part II
Administration

This part provides information about the following topics:

Topics:

10

The Database Users Page

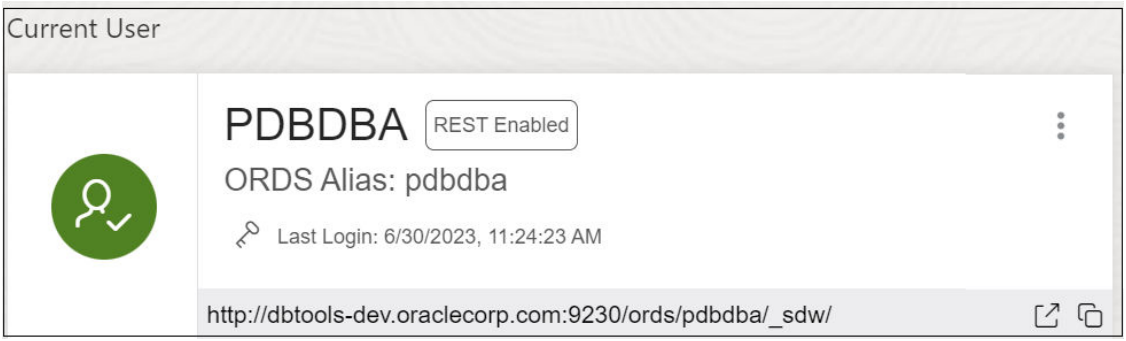
The Database Users page enables you to perform user management tasks such as create, edit, enable REST, and delete, and create and manage self service schemas. For user management, the actions available are based on the user privileges (CREATE, ALTER, DROP) granted to you.
Users with no assigned privileges can still access the Database Users page to browse all users. However, the only action available to them is changing their password.

To navigate to the Database Users page, do either of the following:

- In the Launchpad page, select the **Administration** tab and click **Database Users**.
- Click **Selector** to display the navigation menu. Under Administration, select **Database users**.

The Database Users page consists of two parts: Current User and All Users.

Current User



Displays information about the current user such as user name, whether REST Enabled or not, REST Alias, account expiration (in days), and the last login date and time. The icon on the left displays the user status with one of the following colours: green (Open), blue (Locked), and red (Expired).

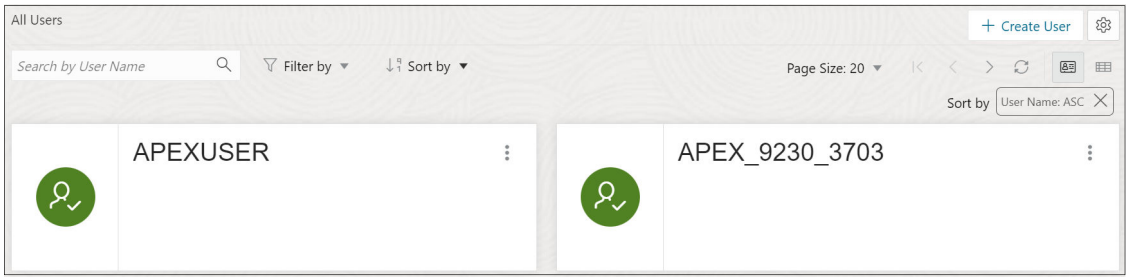
The URL at the bottom is displayed only if the user is REST Enabled. It provides the URL to the Database Actions user login page. Use **Copy to Clipboard** to copy the URL to the user's clipboard.

Click **Actions** to open the context menu. The actions available are:

- **Edit**: Opens the Edit User Dialog, where you can edit current user information. See Creating or Editing a User.
- **Enable REST**: Enables REST for a user where disabled. When this option is selected the first time, it opens the Enable REST dialog. See Enabling REST.
- **Disable REST**: Disables REST where enabled for a user.

- **Drop REST Services**: Removes the REST data for a user, such as REST Alias, Base Path and so on, which is stored in Oracle REST Data Services (ORDS).
- **Delete**: Opens the Delete User dialog, where you can delete the user. See Deleting a User.

All Users



Displays information about all other users that have been created in the database. You can use the Search field, which is case insensitive, to search for users, or sort the users in ascending or descending order using the sort icons, or filter by user status or REST status.

To create a user, click **Create User** to open the Create User dialog. For more information, see Creating or Editing a User.

There are two views available:

(Card View): Displays the user information in a card view. This is the default display view. Each user card provides details such as user status, password expiry, user name, and the context menu.

(Grid View): Displays the user information in a tabular format. The last column in each row contains the context menu icon.

10.1 Creating or Editing a User

You can create a new database user or edit an existing database user.

- To create a new database user, click **Create User** in the Database Users page.
- To edit an existing database user, select **Edit** from the context menu for the associated user.

The user properties are grouped under two tabs: User and Granted Roles.

User Tab

Specifies general properties for the database user.

- **User Name**: The user name string. For an existing user, this field is read-only. To change the name, you must drop the user and create a new user with the desired name.
- **Password**: Password string for the new user, or new password for an existing user. You must also type the same password string for **Confirm Password**.
- **Password Expired**: If this option is selected, the password is marked as expired, and the user must change the password before being permitted to connect to the database.

- **Account is Locked**: If this option is selected, the user will not be permitted to connect to the database until a DBA user unlocks the account associated with this user.
- **Quota on tablespace USERS**: Enter or select to assign quota on the user's default tablespace. By default, the user has no quota on the default tablespace.
- **Web Access**: If this option is selected, the user is enabled for REST access. Expand **Web access advanced features** to specify the related fields: **Authorization required**, **REST Alias**, and **URL Mapping Type**.

Granted Roles Tab

Specifies roles to be granted to the user.

Use **Filter by role** to quickly locate the required roles.

For each role, you can select **Granted** to grant the role, **Admin** to permit the user to grant the role to other users, and **Default** to use the default settings for Granted and Admin. A new user is granted **CONNECT** and **RESOURCE** roles when Web Access is selected.

Show code: Select this option to view the PL/SQL code equivalent of the Create/Edit User slider. You can copy and execute this PL/SQL code in the worksheet to perform the same action that occurs when you click **Create/Edit** in the Create/Edit User slider.

10.2 Enabling REST

You can enable REST for a user that has not been REST enabled.

In the Database Users page, select **Enable REST** from the user's context menu.

When you select Enable REST for a user for the first time, REST Enable User dialog is displayed. Subsequently, if you disable REST and then select Enable REST again, you receive a message stating that REST is enabled. In this case, the REST data previously provided is used for enabling REST. To enter new REST data, select **Drop REST Services** and then select **Enable REST** again.

Schema Alias: Enter the alias for the schema name to use in the URL.

The alias is entered at the time of signing in. See Signing In to Database Actions

URL Mapping Type: Select **BASE_PATH** or **BASE_URL**.

Authorization Required: For a schema, controls whether Oracle REST Data Services should require user authorization before allowing access to the Oracle REST Data Services metadata catalog of this schema.

10.3 Deleting a User

Use this option to delete users.

In the Database Users page, select **Delete** from the user's context menu to delete a user.

> **WARNING:**
>
> The number of user's active sessions is displayed in the Delete User dialog window.

- **Cascade**: If this option is selected, all dependent objects are also deleted.

- **Drop REST Services**: If this option is selected, all user REST data is removed from ORDS.

> **Note:**
>
> If you do not select this option when deleting a user, the next time you create a user with the same user name, it will still retain the REST-enabled property.

Click **Delete User** and a confirmation notification is displayed.

> **Note:**
>
> An error notification is displayed, if the user has active sessions. In such cases, you must close the active sessions before you can delete the user.

10.4 Creating a Self Service Schema

> **Note:**
>
> Available only if you have the **DBA** and **PDB_DBA** roles assigned.

The Self Service Schema page enables you to create a database schema on user request by generating a secure URL that is automatically sent by email to the user. Users can access a public page to request for a schema.

The steps are broken down into the following sections:

- Activate the Self Service Schema in Oracle REST Data Services (ORDS)
- Enable the Self Service Schema feature in Database Actions
- Request database schema by user
- Manage schema requests on the Self Service Schema Page

Activate the Self Service Schema in ORDS

1. By default, this feature is not available unless activated. To activate, set the configuration property in ORDS by using the command line or by adding the property to the XML configuration file.
   - From the command line, use the following command:

     ```
     ords config set feature.sdw.selfServiceSchema true
     ```

   - In the configuration XML file, add the property as shown below:

     ```
     <?xml version="1.0" encoding="UTF-8"?>
     <!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
     <properties>
     <comment>Saved on Wed Nov 23 16:57:17 UTC 2022</comment>
     <entry key="database.api.enabled">true</entry>
     ```

```
<entry key="feature.sdw.selfServiceSchema">true</entry>
<entry key="sdw.dev.doc.root">C:\Users\jsmith\Desktop\static-resources</
entry>
<entry key="standalone.context.path">/ords</entry>
<entry key="standalone.http.port">8080</entry>
</properties>
```

After the ORDS configuration property is set to true, you can enable or disable the feature from the Database Users page in the Database Actions user interface.

Enable the Self Service Schema Feature in Database Actions

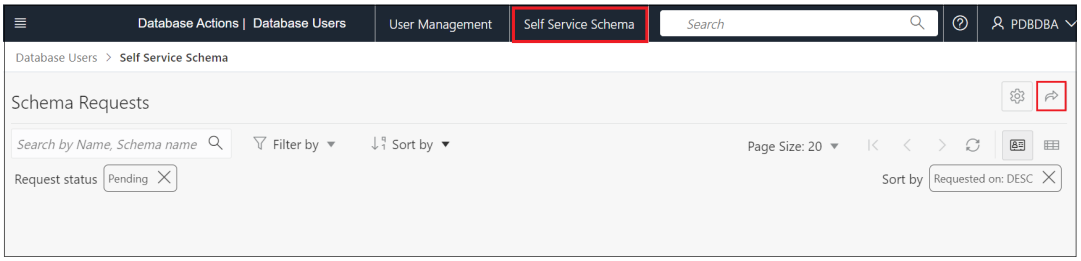
2. In the Database Users screen, click **Self Service Schema Settings**, which is located next to Create User.
3. In the Self Service Schema settings slider:
   a. Select **Enable Feature** to enable the Self Service Schema feature.

      The URL to the Request Schema page appears, which can be shared with the user.

   b. Click **Save**.

   The **Self Service Schema** tab appears in the header next to User Management on the Database Users screen. Navigate to the Self Service Schema page using this tab.

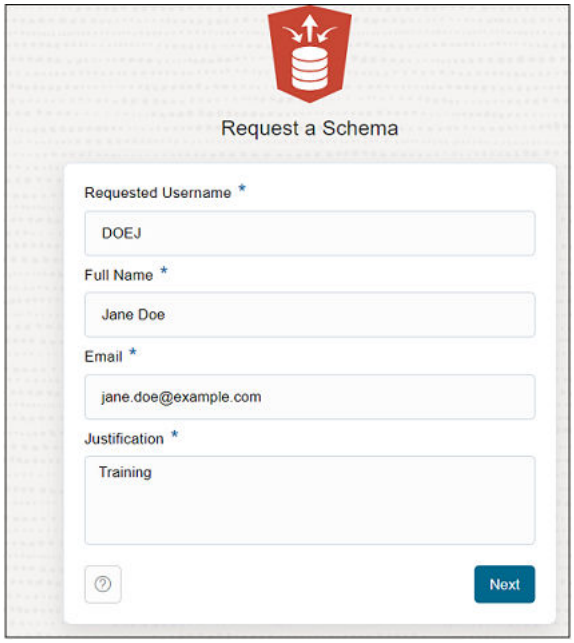
Figure 10-1 Self Service Schema Tab



Request Database Schema by User

4. In the Request a Schema page, a user has to enter details such as Requested User Name, Full Name, Email, and Justification, click **Next** and then click **Submit**.

> **Note:**
>
> Click towards the top right corner in the Self Schema Service page (see Figure 10-1) to access the Request page URL.

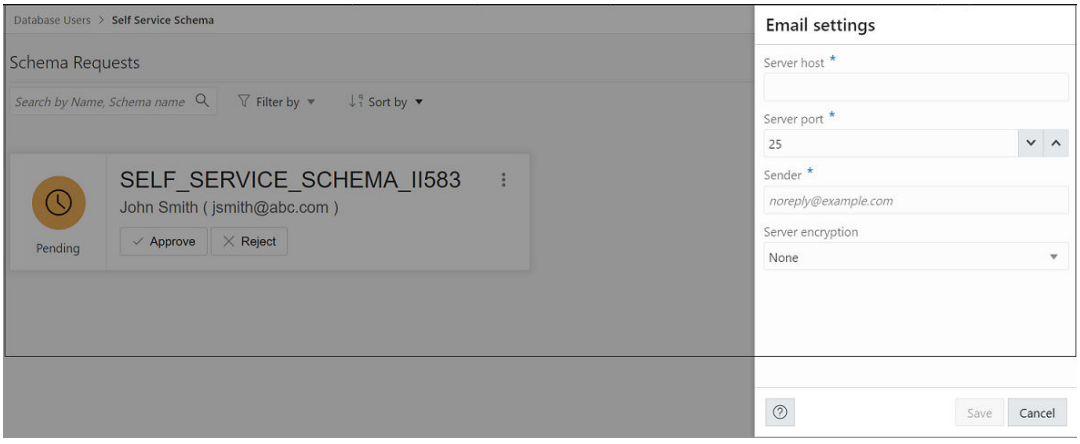


A confirmation message appears. The user request becomes available on the Self Service Schema page in a card format.

Manage Schema Requests on the Self Service Schema Page

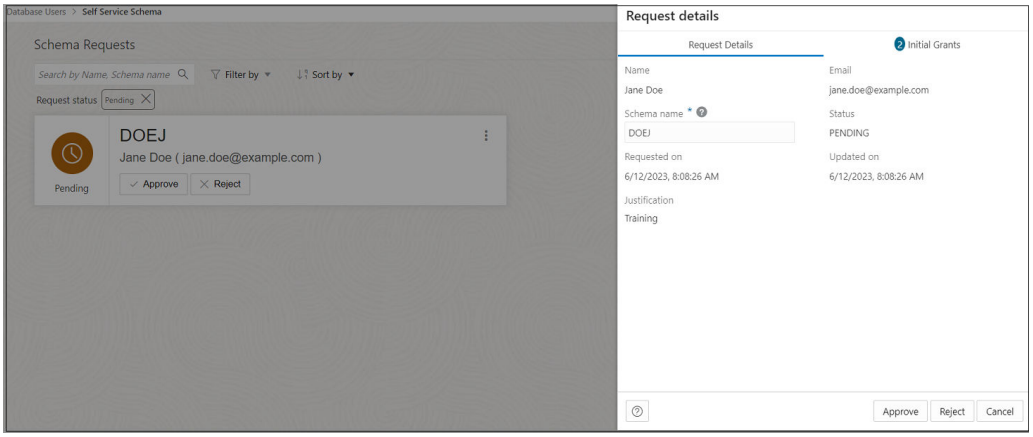
5. You can see and manage user requests on the Schema Requests section in the Self Service Schema page.

   You can set up an SMTP server to automatically send emails to users to notify whether the schema request has been approved or rejected. To define SMTP settings, click on the top right corner of the Schema Requests page. Enter fields such as Server Host, Server Port, Sender Name and Server Encryption to set up the server.



6. After SMTP settings are set, you can approve or reject the request by clicking the appropriate button on the card.

   - If you click **Approve**, a panel with two tabs appears: Request Details and Initial Grants. You can make changes to the schema name and customize the initial roles for the requested schema. By default, the Resource and Connect roles are pre-selected. Click **Approve**.

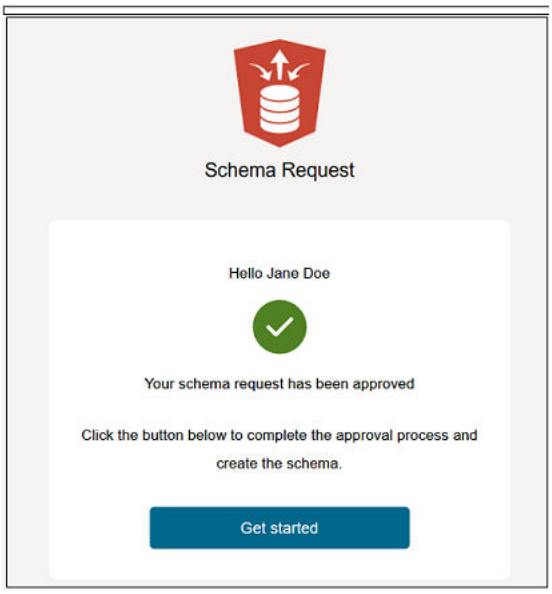


- If you click **Reject**, you are prompted to specify a reason for rejection, which will be sent to the user.

The schema card is updated and shows the status of the request when it is approved or rejected.

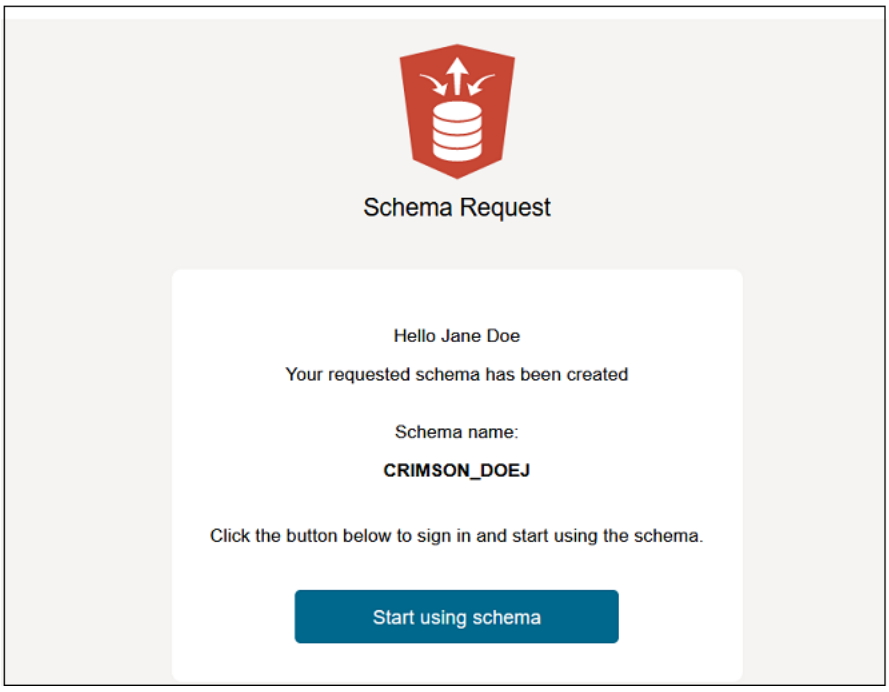
> **Note:**
>
> - If you cannot see the request card on the Schema page, check if any filters are set and clear them.
> - After the schema is approved, you can still reject the schema. To do that, click the **Actions** icon on the card and select **Open Request Details**. Click **Reject**, enter a reason for rejection, and click **Submit**. The status of the card changes to Rejected and an email notification is sent to the user.

When you click **Approve**, a notification mail is sent to the user's email address.



7. Click **Get Started**. You are taken to a page where you need to set the password for the new schema. Enter **Password** and **Confirm Password** and click **Submit**.

8. Another mail is sent to the user notifying that the schema has been created and providing a **Start Using Schema** link. The user can use the link to log in to the schema after providing the new password.



After the schema is created, the status of the user request card on the Schema Requests page changes to **Fulfilled**.

> **Note:**
>
> If you want to remove a schema request after the status changes to Fulfilled, you will need to use queries in the SQL worksheet.

11
The APEX Workspaces Page

> **Note:**
>
> You can create and delete workspaces only if you have the **APEX_Administrator_Role** assigned.

You can create APEX workspaces using the APEX Workspaces page. For more information about APEX workspaces, see Workspace and Application Adminstration in the *Oracle Application Express Administration Guide*.

To navigate to this page, do either of the following:

- In the Launchpad page, select the **Administration** tab and click **APEX Workspaces**.
- Click **Selector** to display the navigation menu. Under **Administration**, select **APEX Workspaces**.

For a specific workspace, you can perform the following actions using the **Actions** context menu:

- **Create a Workspace**: See Create a Workspace
- **Delete a Workspace**: Removes the workspace and all associated applications from the database.
- **View Schema Details**: Navigates to the Database Users page to view the related schema information.

11.1 Create a Workspace

> **Note:**
>
> You require Administrator privileges to create a workspace.

To create an APEX workspace:

1. In the APEX Workspaces page, at the top right, click **Create Workspace**.
2. Enter the following fields:
   - **Workspace name**: Enter a name for the workspace.
   - **Database user**: Select a schema from the drop-down list, or type to enter a new schema.
   - **Workspace password**: Enter a password for the workspace.
   - **APEX Administrator**: Enter the administrator user details.

Show code: Select this option to view the PL/SQL code equivalent of the Create Workspace slider. You can copy and execute this PL/SQL code in the worksheet to perform the same action that occurs when you click **Create** in the Create Workspace slider.

3. Click **Create**.

   The new workspace is displayed in the APEX workspaces page. The workspace card includes details such as the number of applications in the workspace, the number of developers using the workspace, and the number of administrators responsible for the workspace. Depending on your role, the **Actions** context menu provides options such as creating and deleting a workspace, and viewing schema details.

12

The Data Pump Page

> **Note:**
>
> This feature is only available for Oracle Database 12.2 and later releases.

The Data Pump page enables you to to monitor Data Pump jobs that were initiated through the available Database API endpoints, the `DBMS_DATAPUMP` package, or the SQL Developer Data Pump Export and Import wizards.

To import data using Data Pump, click **Import Data**. See Importing Data.

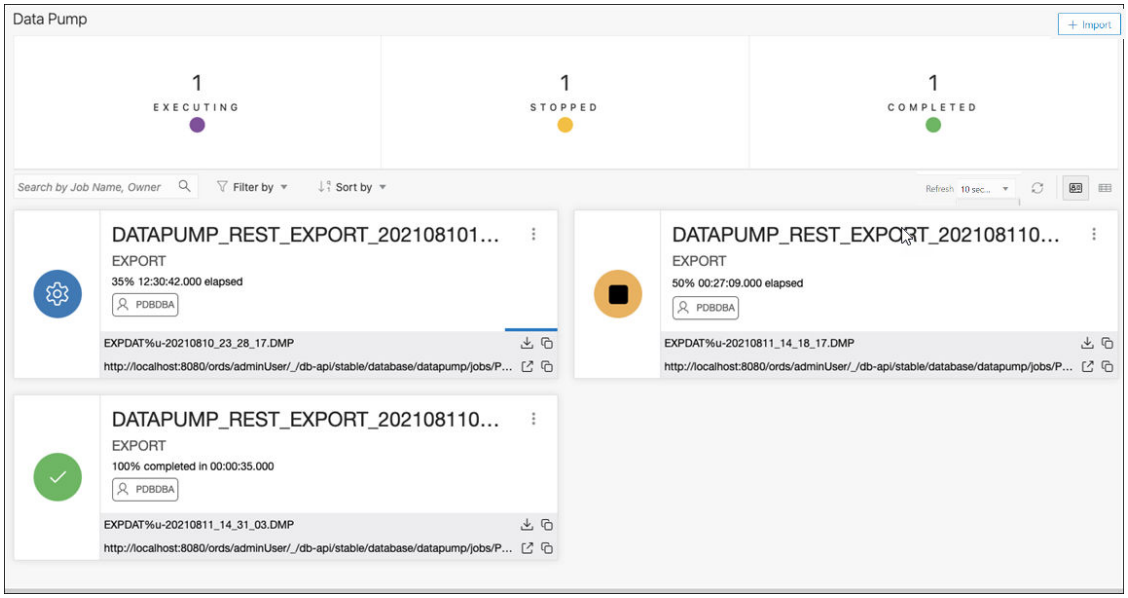
The section at the top displays the total number of executing jobs, stopped jobs, and completed jobs. Click a tile (example, STOPPED) to filter and view the corresponding list of STOPPED jobs in the default card format.

You can filter or sort the jobs and set the time period by which to refresh the data.

A job card displays the following details: Job name, import or export operation, percentage of completion and time elapsed, and links to dump files and logs. The status of the job is indicated by the colour of the icon on the left side of the card. Green indicates successful jobs, yellow indicates that the jobs need to be reviewed, and blue indicates that the jobs are in progress.

In a job card, you can:

- Use **Download** to access dump files for completed jobs.
- Use **Log** to access the log files.



12.1 Importing Data

This section provides the steps for importing data using Oracle Data Pump into your on-premises database.

Before you begin, you must have an export job. To create an export job using cURL, see Create an Export Data Pump Job

1. In the Data Pump page, on the top right, click **Import**.

   The Import wizard appears.

2. In the Source step, enter the following fields:

   a. **Directory**: Select the directory that contains the source dump files.

   b. **Import Pattern**: Type the import pattern.

   Click **Next**.

3. In the Import step, enter the following fields:

   - **Import Name**: Enter a name for the import job.
   - **Import Type**: Select the type of import. The options are Full, Tables, Schemas, and Tablespaces.

     > **Note:**
     >
     > If you select **Full**, you skip the Filter step in the wizard and directly go to the Mapping step.

   - **Content**: Select **Data Only**, **DDL Only**, or **Data and DDL**.
   - **Encrypt**: Select if encrypted and enter an encryption password.

   Click **Next**.

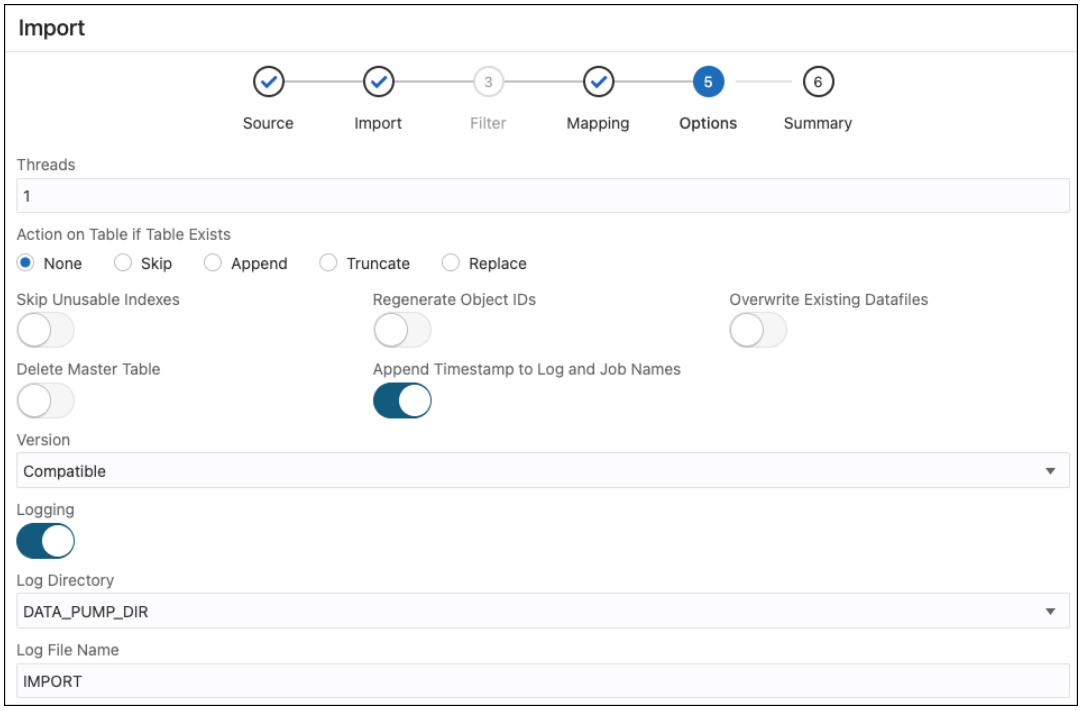
4. In the Filter step, depending on the import type, all the schemas, tables, or tablespaces for the import job are listed. Select the ones that apply. Click **Next**.

5. In the Mapping step, select the source schema and enter a new name for the target schema. If needed, do the same for tablespaces. Click **Next**.

6. In the Options step, enter the following fields:

   - **Threads**: Specifiy the maximum number of threads of active execution operating on behalf of the import job. The default is 1.
   - **Action on Table if Table Exists**: Specify the action needed if that table that import is trying to create already exists.
   - **Skip Unusable indexes**: Select to specify whether the import skips loading tables that have indexes that were set to the Index Unusable state.
   - **Regenerate Object IDs**: Select to create new object identifies for the imported database objects.
   - **Delete Master Table**: Select to indicate whether the Data Pump control job table should be deleted or retained at the end of an Oracle Data Pump job that completes successfully.

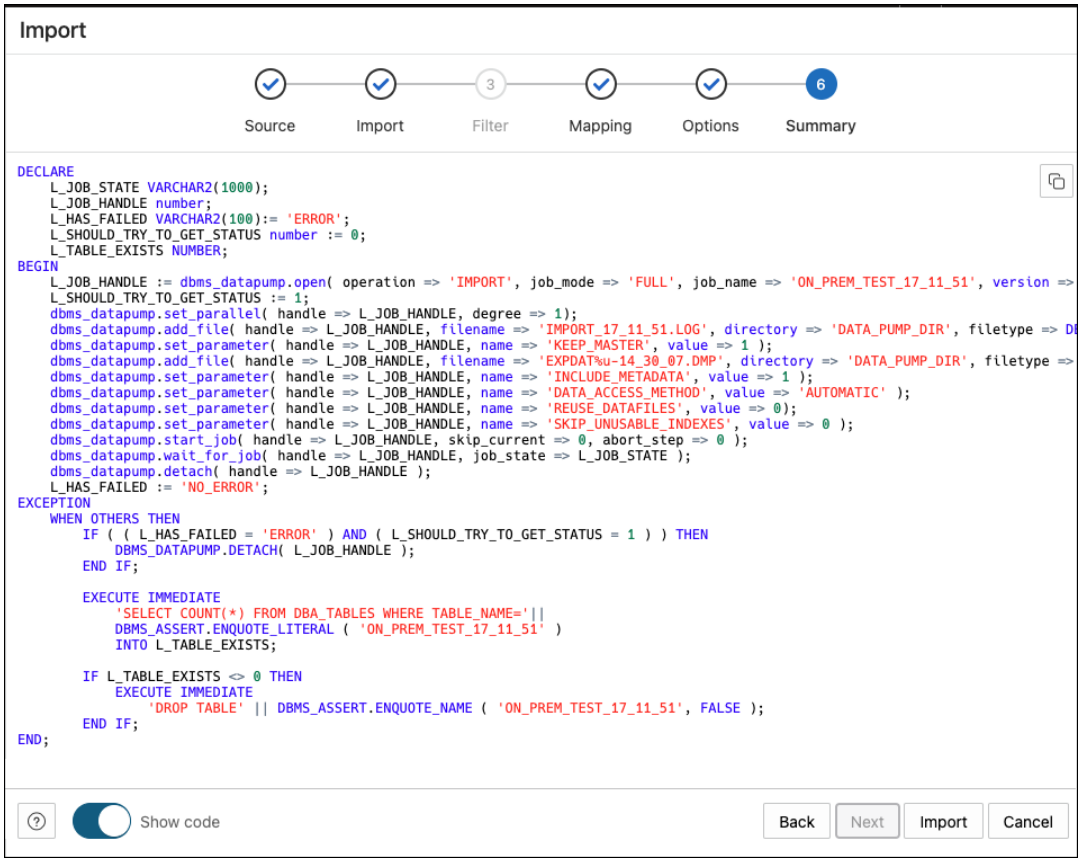
- **Overwrite Existing Datafiles**: Select to indicate that if a table already exists in the destination schema, overwrite it.
- **Version**: Select the version of database objects to import.
- **Logging**: Select to create a log file. Enter the log directory and log file name.



Click **Next**.

7. The Summary step displays a summary of all the selections made in the previous steps.

   Select **Show Code** at the bottom to see the PL/SQL code equivalent of the form.



Click **Import**.

The start of the job execution is displayed on the Data Pump page.

Part III
Monitoring

This part provides information about the monitoring features available with Database Actions.

Topics:

13

The Monitoring Pages

The Monitoring menu provides access to several pages to view the performance and other characteristics of your database.

13.1 The Monitoring Overview Page

> **Note:**
>
> Available only if you signed in as a database user with **DBA** and **PDB_DBA** roles.

The Monitoring Overview page displays general information about the database.

To navigate to the Overview page, click **Selector** to see the navigation menu and then select **Monitoring**.

The widgets on this page show snapshot information about the database status, online database storage, sessions, wait events, user accounts, alerts, and expiring passwords. Click a widget to go to its page where you can see a more detailed view of the data.

- **Database Status**: Shows the status of the database.
- **Used Online Database Storage**: Displays how much storage is being used by the database. You can click the title to open The Storage Page.
- **Sessions**: Displays the number of sessions by session status. You can click the title to open The Sessions Page.
- **Waits**: Displays how many wait events are occurring in the database by wait event class. When you click the title:
  - For Oracle Database 19c and later releases, the Performance Hub page is displayed.

  - For Oracle Database 18c and previous releases, the Activity-Waits page is displayed.
- **Users**: Displays how many user accounts are in the open, locked and expired statuses. You can hover over a status to see a list of the user accounts with that status.
  - **Open**: This status indicates that the user's account is unlocked and access to the database is enabled.
  - **Locked**: This status indicates that the user's account is locked and access to the database is disabled. The account must be unlocked to enable access to the database.
  - **Expired**: This status indicates that the user's password has expired and must be changed before the user can log in to the database.
- **Alerts**: Displays a summary of alerts over the last 7 days. You can click the title to open The Alerts Page
- **Expiring Passwords**: Shows a list of user accounts and whether a user account password has expired or the number of days before it will expire.

13.2 The Performance Hub Page

> **Note:**
>
> Available only if you signed in as a database user with **DBA** and **PDB_DBA** roles for Oracle Database 19c and later releases.

The Performance Hub page shows performance data for a time period you specify.

To navigate to the Performance Hub page, do either of the following:

- In the Launchpad page, select the **Monitoring** tab and click **Performance Hub**.
- Click **Selector** to display the navigation menu. Under Monitoring, select **Performance Hub**.

> **Note:**
>
> The Performance Hub page is available in the following user interface languages: French, Japanese, Korean, Traditional Chinese, and Simplified Chinese. If you change the language to German, Spanish, Italian, or Portuguese in Preferences, the Performance Hub page reverts to English.

The Performance Hub page consists of these parts:

- **Time Range Area**: Use the controls in time range area at the top of the page to specify the time period for which you want to view performance data.
- **ASH Analytics Tab**: Use this tab to explore ASH (Active Session History) information across a variety of different dimensions for the specified time period.
- **SQL Monitoring Tab**: Use this tab to view the top 100 SQL statement executions by different dimensions for the specified time period, and to view details of SQL statement executions you select.

See Performance Hub in the Database service documentation for more information.

13.3 The Instance Viewer Page

> **Note:**
>
> Available only if you signed in as a database user with **DBA** and **PDB_DBA** roles.

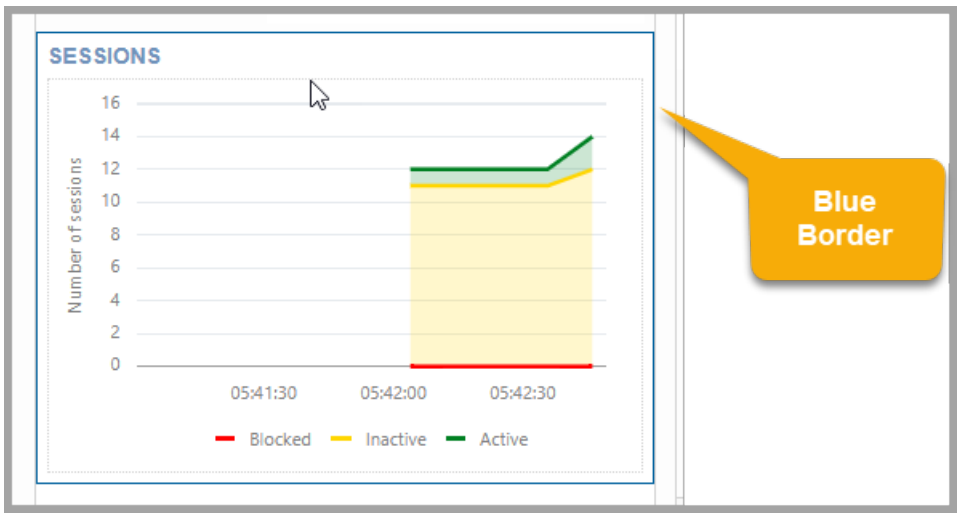
The Instance Viewer is a performance dashboard that enables a DBA user to see a graphical representation of information about the instance associated with a connection.

To navigate to the Instance Viewer page, do either of the following:

- In the Launchpad page, select the **Monitoring** tab and click **Instance Viewer**.
- Click **Selector** to display the navigation menu. Under Monitoring, select **Instance Viewer**.

When hovering over a chart, a dark blue border indicates that you can click to drill down to the relevant page for more detailed information.

Figure 13-1 Dark Blue Border to Drill Down



The types of information displayed include:

- Database
- Clients
- Sessions
- Processes (Counts, Execution Rate, Parse Rate, Open Cursors, Commit Rate)
- Waits
- Memory (DB Block Rate, Logical Reads, Allocation, Redo Generation)
- Storage (Files, Redo Log)

- DB CPU Ratio (database operations as a percentage of CPU activity)
- Top SQL (provides performance metrics on SQL operations, which are sorted based on the CPU time required)

13.4 The Logins Page

> **Note:**
>
> Available only if the following three conditions are met:
>
> - You are signed in as a database user with **DBA** and **PDB_DBA** roles for Oracle Database 12c and later releases.
> - Auditing is enabled in the database.
> - At least one of the four actions displayed on the Logins page must have an audit policy that is enabled. By default, Failed Logins is enabled.

The Logins page shows the number of successful logins, failed logins, timed-out logoffs, and logoffs that have occurred within the last hour.

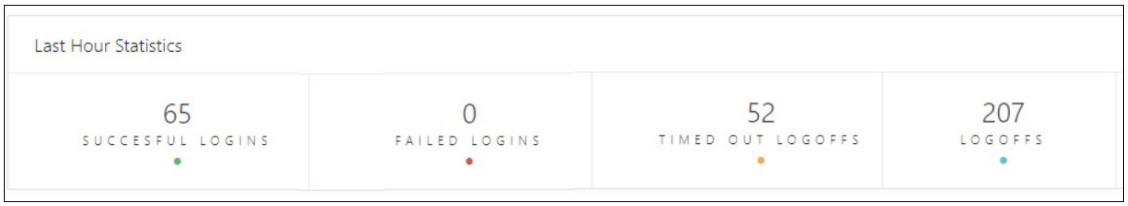
To navigate to the Logins page, do either of the following:

- In the Launchpad page, select the **Monitoring** tab and click **Logins**.
- Click **Selector** to display the navigation menu. Under Monitoring, select **Logins**.

The Last Hour Statistics widget only displays the actions being audited. If any of the tabs (succesful logins, failed logins, timedout logoffs or logoffs) are disabled, the database is currently not auditing that action. If you want to activate the action, you must create an audit policy with the desired action. To create an audit policy, see Auditing Activities.

Click a tab in Last Hour Statistics to view the corresponding entries in the table.

Figure 13-2 Tabs in Last Hour Statistics

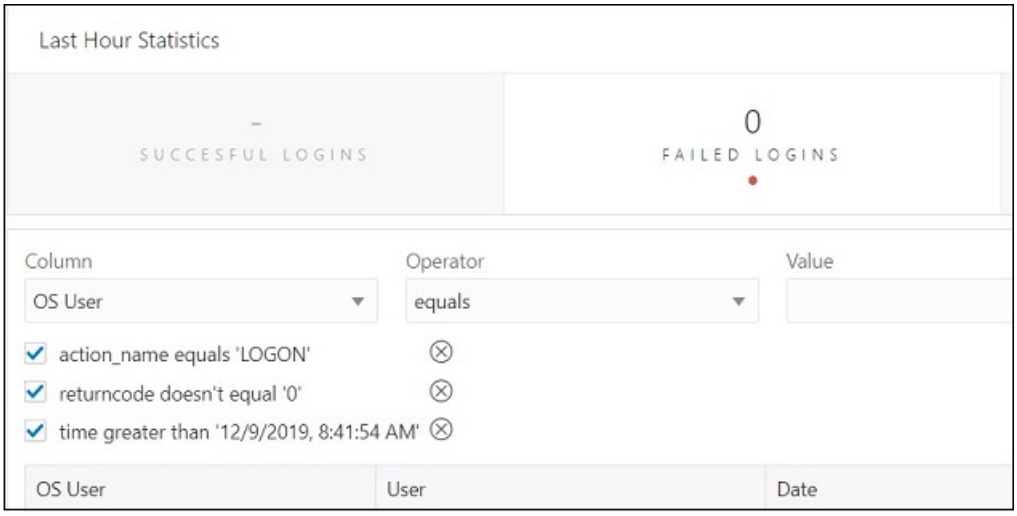


You can search for results in the table by entering values in the Column, Operator and Value fields and clicking Filter. You can also filter by selecting one or more of the available filter options for each action.

Click Refresh to refresh the data in the grid.

The following figure shows the three filters for the Failed Logins action.

Figure 13-3 Filters for Failed Logins



To return to the previous page, remove filters by clicking Remove Filter ⊗ or refresh the browser.

13.5 The Alerts Page

> **Note:**
>
> Available only if you signed in as a database user with **DBA** and **PDB_DBA** roles.

The Alerts page is a chronological log of messages and errors and is commonly used to learn whether the background processes have encountered errors. You can review the alert log periodically to verify that your database system is operating normally.

To navigate to the Alerts page, do either of the following:

- In the Launchpad page, select the **Monitoring** tab and click **Alerts**.
- Click **Selector** ☰ to display the navigation menu. Under Monitoring, select **Alerts**.

The alert log includes the following:

- Nondefault initialization parameters used at startup
- Administrative operations, such as STARTUP, SHUTDOWN, ARCHIVE LOG, RECOVER, and CREATE/ALTER/ DROP DATABASE/TABLESPACE
- Messages and errors relating to the functions of certain background processes, such as LGWR
- Internal errors (ORA-600), block corruption errors (ORA-1578), and deadlock errors (ORA-60)

Click Refresh ↻ at the top right of the page to refresh the data.

You can search for a specific value in the log by selecting the display column in the first drop-down list, selecting the condition in the second drop-down list, entering the search value in the box, and clicking the search icon.

In the display table, if you right-click the header row, you see:

Columns: Enables you to select columns to show or hide.

Sort: Displays a dialog box for selecting columns to sort by. For each column, you can specify ascending or descending order, and you can specify that null values be displayed first.

If you right-click any other part of the display table, you see:

Count Rows: Displays the number of rows in the table.

Single Record View: Enables you to view data for a table or view, one record at a time.

Copy: Enables you to copy data from a cell or a row or a range of rows. To copy from more than one row, select the rows you want to copy, right-click by pressing the SHIFT or CTRL key, and select **Copy**.

13.6 The Sessions Page

> **Note:**
>
> Available only if you signed in as a database user with **DBA** and **PDB_DBA** roles.

The Sessions page shows information about all currently open sessions in the database.

To navigate to the Sessions page, do either of the following:

- In the Launchpad page, select the **Monitoring** tab and click **Sessions**.
- Click **Selector** to display the navigation menu. Under Monitoring, select **Sessions**.

The data is automatically refreshed at intervals ranging from 10 seconds to 2 minutes. You can also refresh the data by clicking Refresh at the top right of the screen.

The table shows summarized data about each open session. Select a session in the table to see more detailed data in the Session Details table below, such as the last SQL statement, explain plan, waits, contention, and so on. You can use the Column, Operator and Value fields to search for the required sessions.

In the display table, if you right-click the header row, you see:

Columns: Enables you to select columns to show or hide.

Sort: Displays a dialog box for selecting columns to sort by. For each column, you can specify ascending or descending order, and you can specify that null values be displayed first.

If you right-click any other part of the display table, you see:

Count Rows: Displays the number of rows in the table.

Single Record View: Enables you to view data for a table or view, one record at a time.

Copy: Enables you to copy data from a cell or a row or a range of rows. To copy from more than one row, select the rows you want to copy by pressing the SHIFT or CTRL key, right-click and select **Copy**.

13.7 The Storage Page

> **Note:**
>
> Available only if you signed in as a database user with **DBA** and **PDB_DBA** roles.

The Storage page shows the storage used based on the current allocation of tablespaces along with additional drill-down capabilities to view segments.

To navigate to the Storage page, do either of the following:

- In the Launchpad page, select the **Monitoring** tab and click **Storage**.
- Click **Selector** to display the navigation menu. Under Monitoring, select **Storage**.

You can refine the list of segments shown by using the filter feature. Click **View Datafiles** to view the datafiles in each tablespace.

You can view tablespace and segment space usage.

To view space usage information

1. From the Database drop-down menu, click **Storage**.

   The Storage page displays. If the Oracle database is version 12c or later, the Storage page shows the used and allocated storage space for tablespaces in any pluggable database. If the Oracle database is version 11g, the Storage page shows the used and allocated space for the entire database.
2. You can click a tablespace to view its storage information. An interactive report appears, showing the segments that exist within the tablespace. Most segments are user objects, and they include tables, LOBs, and indexes.
3. On the Segments page, you can refine the list of segments shown by using the filter feature.

   For example, you can search for all the segments for a specific owner (schema) by selecting OWNER from the first drop-down list, entering the owner (schema) name in the box, and clicking the search icon.

13.8 The Parameters Page

> **Note:**
>
> Available only if you signed in as a database user with **DBA** and **PDB_DBA** roles.

The Parameters pages displays initialization parameters, which are used to configure the database instance, including memory structures, and define locations for database files.

To navigate to the Parameters page, do either of the following:

- In the Launchpad page, select the **Monitoring** tab and click **Parameters**.
- Click **Selector** to display the navigation menu. Under Monitoring, select **Parameters**.

Values for initialization parameters are stored in a text-based initialization parameter file (PFILE) or binary server parameter file (SPFILE). The initialization parameter file is read at database instance startup.

Click Refresh at the top right of the page to refresh the data.

To perform a search, enter values in the search criteria columns and click the search icon to locate the initialization parameter.

In the display table, if you right-click the header row, you see:

Columns: Enables you to select columns to show or hide.

Sort: Displays a dialog box for selecting columns to sort by. For each column, you can specify ascending or descending order, and you can specify that null values be displayed first.

If you right-click any other part of the display table, you see:

Count Rows: Displays the number of rows in the table.

Single Record View: Enables you to view data for a table or view, one record at a time.

Copy: Enables you to copy data from a cell or a row or a range of rows. To copy data from more than one row, select the rows you want to copy by pressing Shift or Ctrl, right-click and select **Copy**.

13.9 The Real Time SQL Monitoring Page

> **Note:**
>
> Available for database users with DBA and PDB_DBA roles for Oracle Database 18c and previous releases. For non-administrator users, this feature is available only for Oracle Database 19c and later releases.

The Real Time SQL Monitoring page shows in real time the SQL statements that are being monitored in the database.

To navigate to the Real Time SQL Monitoring page, do either of the following:

- In the Launchpad page, select the **Monitoring** tab and click **Real Time SQL Monitor**.
- Click **Selector** to display the navigation menu. Under Monitoring, select **Real Time SQL Monitor**.

In the Auto Refresh drop-down list, you can select the time (in seconds) to periodically refresh the data. Select `0 seconds` to disable the auto-refresh.

This tool helps identify run-time issues for SQL statements by providing two major functions:

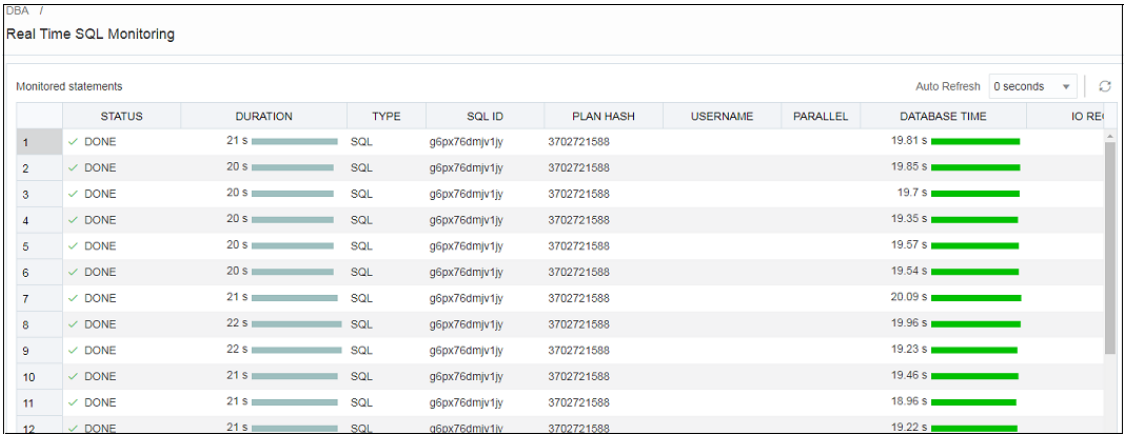
- General view of monitored statements
- View of SQL execution details

General View of Monitored Statements

The page contains a table of SQL statements currently running. This table shows the following information:

- **Status**: Current state of the SQL statement execution. For example, a SQL statement that has already finished its execution will show a status of "DONE".
- **Duration**: Amount of time a SQL statement is taking, or has taken, to execute.
- **SQL ID**: SQL identifier of the statement being monitored.
- **Session ID**: Session identifier that is executing, or has executed, the SQL statement.
- **Session Serial Number**: Uniquely identifies a session's objects.
- **Instance Degree of Parallelism**: This Degree of Parallelism (DOP) column shows how many instances and parallel execution servers are allocated. It is shown in the form of “number of instances” | “number of parallel servers”.
- **Database Time**: Place the cursor over the database time to see a breakdown of the time and wait events.
- **CPU Time**: CPU time consumed by the execution of the query.
- **I/O Time**: I/O time consumed by the execution of the query.
- **Start Time**: Time in which the execution of the SQL statement started.
- **SQL Statement**: SQL statement being monitored.

Figure 13-4 Real Time SQL Monitoring Table



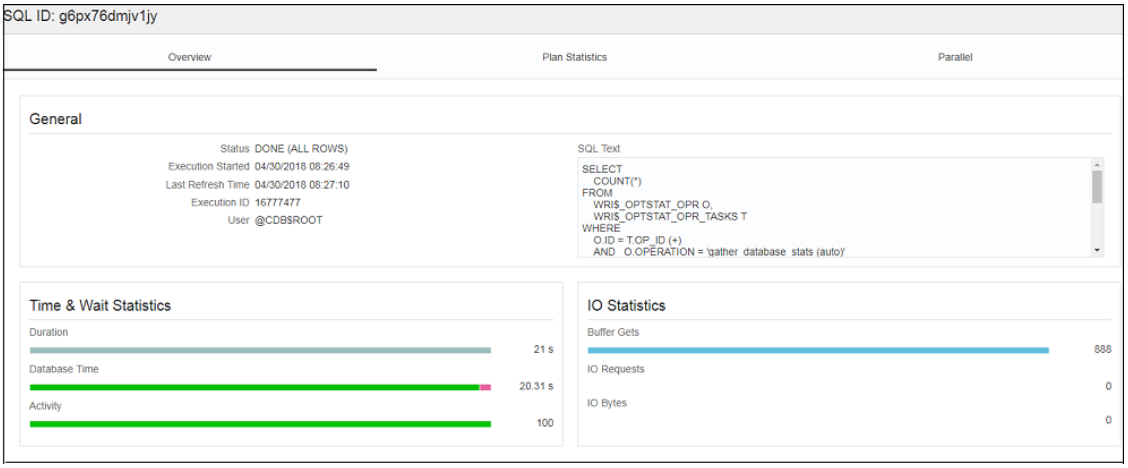
View of SQL Execution Details

When a SQL statement is drilled down from the main monitor table, a detailed view is shown. The SQL ID, Start Time and the SQL Execution ID represent the execution key that uniquely identify this SQL statement. A detail view consists of the general characteristics that integrate the execution of a SQL statement.

General information about the query execution is provided:

- **Execution Plan**: Degree of Parallelism of the SQL statement
- **Execution Started**: Time that the SQL statement execution started
- **Last Refresh Time**: Last update time of the SQL monitor registry for the SQL statement
- **Execution ID**: Execution identifier
- **User**: User in the format USER@CONTAINER
- **SQL Text**: Formatted view of the SQL statement that is being executed.

Figure 13-5 SQL Execution Details in Real Time SQL Monitor



General statistics of the SQL statement are provided: total duration of execution, the number of buffered gets, number of Input/Output requests and bytes.

Detailed information of the statement: This space holds the information corresponding to the explain plan, parallel behavior and CPU activity involved in the execution of the statement:

- **Plan Statistics**: Explain plan of the execution of the SQL statement in the form of a table. Each row is a different operation involved in the execution of the SQL statement and it shows hierarchy dependency by adding a space at the beginning of the text in the Operation column.
- **Parallelism Details for the SQL statement**: Each execution consists of a parallel coordinator and one or more parallel sets. Each set can have one or more processes. When a row has dependents, each of its columns will be the sum of the values of its dependants. When this happens, a sigma symbol will appear to show that a value consists of the sum of others.

> **Note:**
>
> For more information, see "Monitoring the Database" in the *Oracle Database Administrator's Guide*.

13.10 The Top SQL Page

> **Note:**
>
> Available only if you signed in as a database user with **DBA** and **PDB_DBA** roles.

The Top SQL page displays SQL statements executed in the database based on CPU time consumed. The SQL statement that consumes the maximum CPU time is right at the top, and the remaining statements continue in descending order of CPU time.

To navigate to the Top SQL page, do either of the following:

- In the Launchpad page, select the **Monitoring** tab and click **Top SQL**.
- Click **Selector** to display the navigation menu. Under Monitoring, select **Top SQL**.

This feature enables you to focus your SQL tuning efforts on the statements that can have the most impact on database performance.

Click a row to see a formatted view of the SQL statement, the execution plan, the runtime history from Active Session History, and a SQL Tuning Advisor report. Active Session History is part of the Diagnostics Pack, and SQL Tuning Advisor is part of the Tuning Pack for Oracle Database.

13.11 The Waits Page

> **Note:**
>
> Available only if you signed in as a database user with **DBA** and **PDB_DBA** roles for Oracle Database 18c and previous releases.

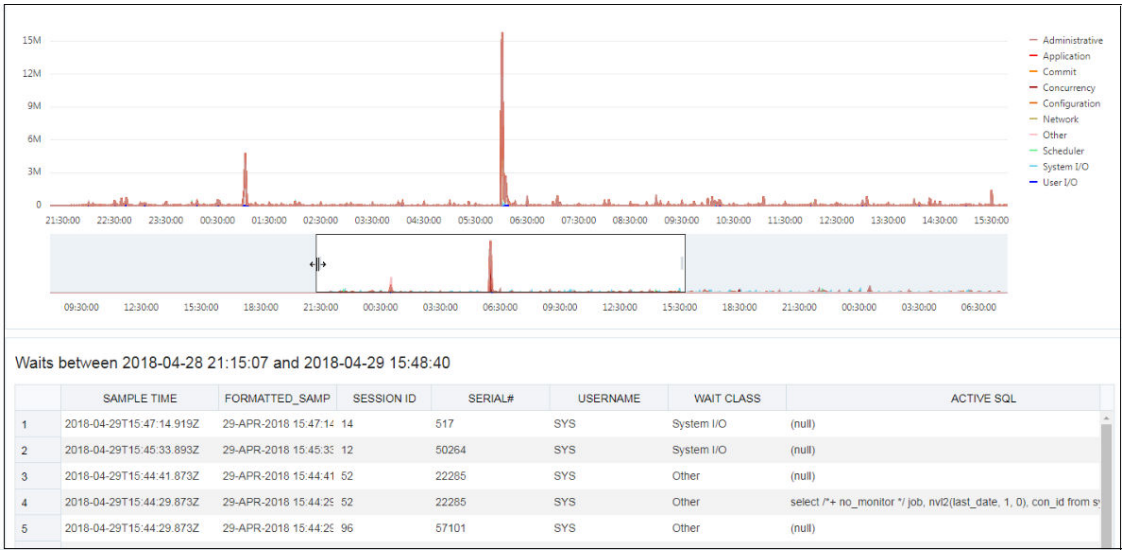
The Waits page shows a chart with the distribution of wait events in real time.

To navigate to the Waits page, do either of the following:

- In the Launchpad page, select the **Monitoring** tab and click **Waits**.
- Click **Selector** to display the navigation menu. Under Monitoring, select **Waits**.

Use the slider controls to zoom in on a specific time period. To use the slider controls, place the cursor over the handles at both sides of the box and drag the sides to the time period required. The chart above will refresh to the selected time period. The table will also automatically refresh and the wait events will filter to that period of time enabling you to easily identify the problem SQL statement.

Figure 13-6 Distribution of Wait Events Chart



When you place the cursor over data points in the chart, a pop-up box displays details about the wait event.

Figure 13-7 Display Wait Event Details

13.12 The AWR Page

> **Note:**
>
> Available only if you signed in as a database user with **DBA** and **PDB_DBA** roles.

The AWR page in Database Actions enables you to generate, view and download Automatic Workload Repository (AWR) reports.

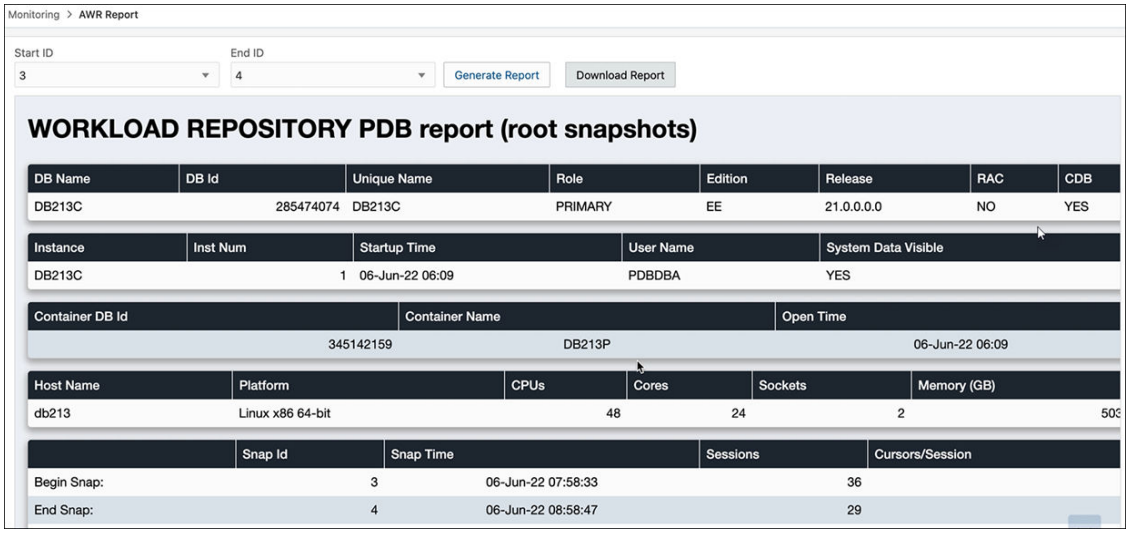
To navigate to the AWR page, do either of the following:

- In the Launchpad page, select the **Monitoring** tab and click **AWR**.
- Click **Selector** to display the navigation menu. Under Monitoring, select **AWR**.

The Automatic Workload Repository collects, processes, and maintains performance statistics for the database. The gathered data can be displayed in reports and views.

To generate or view a report in the AWR page, select the range of the time period required using the **Start ID** and **End ID** fields. The snapshot dropdown list for each field is sorted in descending order and starts with the most recent database snapshot.

> **Note:**
>
> The **Generate Report** button is available only when the value in the Start ID field is lower than the value in the End ID field.


Monitoring > AWR Report
Start ID
3
End ID
4
Generate Report
Download Report
WORKLOAD REPOSITORY PDB report (root snapshots)
DB Name DB Id Unique Name Role Edition Release RAC CDB
DB213C 285474074 DB213C PRIMARY EE 21.0.0.0.0 NO YES
Instance Inst Num Startup Time User Name System Data Visible
DB213C 1 06-Jun-22 06:09 PDBDBA YES
Container DB Id Container Name Open Time
345142159 DB213P 06-Jun-22 06:09
Host Name Platform CPUs Cores Sockets Memory (GB)
db213 Linux x86 64-bit 48 24 2
Snap Id Snap Time Sessions Cursors/Session
Begin Snap: 3 06-Jun-22 07:58:33 36
End Snap: 4 06-Jun-22 08:58:47 29

A

Supported SQL*Plus and SQLcl Commands in SQL Worksheet

This following sections list the SQL*Plus and SQLcl commands supported in the worksheet.

A.1 Supported SQL*Plus Commands

The SQL worksheet supports most of the SQL*Plus commands except those statements that are related to formatting.

- `/ (slash)`
- `@@ { url | file_name[.ext] } [arg ...]`
- `ACC[EPT] variable [NUM[BER] | CHAR | DATE | BINARY_FLOAT | BINARY_DOUBLE] [FOR[MAT] format] [DEF[AULT] default] [PROMPT text | NOPR[OMPT]] [HIDE]`
- `ARCHIVE LOG LIST`
- `BRE[AK] [ON report_element [action [action]]] ...`
- `BTI[TLE] [printspec [text | variable] ...] | [ON | OFF]`
- `C[HANGE] sepchar old [sepchar [new [sepchar]]`
- `CL[EAR] option ...`
- `COL[UMN] [{column | expr} [option ...]]`
- `COMP[UTE] [function [LAB[EL] text] ... OF {expr | column | alias} ...ON {expr | column | alias | REPORT | ROW} ...]`
- `COPY {FROM database | TO database | FROM database TO database} {APPEND | CREATE | INSERT | REPLACE} destination_table[(column, column, column, ...)] USING query`
- `DEF[INE] [variable] | [variable = text]`
- `DESC[RIBE] {[schema.]object[@connect_identifier]}`
- `DISC[ONNECT]`
- `EXEC[UTE] statement`
- `{EXIT | QUIT} [SUCCESS | FAILURE | WARNING | n | variable | :BindVariable] [COMMIT | ROLLBACK]`
- `GET [FILE] file_name[.ext] [LIST | NOLIST]`
- `HO[ST] [command]`
- `L[IST] [n | n m | n * | n LAST | * | * n | * LAST | LAST]`
- `PAU[SE] [text]`
- `PRINT [variable ...]`
- `PRO[MPT] [text]`

- {QUIT | EXIT} [SUCCESS | FAILURE | WARNING | n | variable | :BindVariable] [COMMIT | ROLLBACK]
- R[UN]
- SAV[E] [FILE] file_name[.ext] [CRE[ATE] | REP[LACE] | APP[END]]
- SET system_variable value
- SHO[W] [option]
- SHUTDOWN [ABORT | IMMEDIATE | NORMAL | TRANSACTIONAL [LOCAL]]
- STA[RT] { url | file_name[.ext] } [arg ...]
- STARTUP db_options | cdb_options | upgrade_options
- TIMI[NG] [START text | SHOW | STOP]
- TTI[TLE] [printspec [text | variable] ...] | [ON | OFF]
- UNDEF[INE] variable ...
- VAR[IABLE] [variable [type][=value]]
- XQUERY xquery_statement

A.2 Supported SQLcl Commands

The SQL worksheet supports many of the SQLcl commands.

- ALIAS
- APEX
- BRIDGE
- CTAS
- DDL
- FORMAT
- INFORMATION
- LOAD
- NET
- OERR
- RESERVED_WORDS
- SCRIPT
- SETERRORL
- SODA (See SODA Commands)
- TNSPING
- TOSUB
- WHICH

A.2.1 SODA Commands

SODA (Simple Oracle Document Access) commands are supported in the SQL code editor. SODA allows schemaless application development using the JSON data model. The commands are:

- `SODA create <collection_name>` — Creates a new collection
- `SODA list` — Lists all the collections
- `SODA get <collection_name> [-all | -f | -k | -klist] [{<key> | <k1> <k2> ... | <qbe>}]` — Lists documents in the collection. Optional arguments:
  - `all`: Lists the keys of all documents in the collection
  - `k`: Lists documents matching the specific <*key*>
  - `klist`: Lists documents matching the list of keys
  - `f`: Lists documents matching the <*qbe*>
- `SODA insert <collection_name> <json_str | filename>` — Inserts a new document within a collection
- `SODA drop <collection_name>` — Deletes existing collection
- `SODA count <collection_name> [<qbe>]` — Counts number of documents inside collection. Optional parameter `<qbe>` returns number of matching documents
- `SODA replace <collection_name> <oldkey> <new_{str | doc}>` — Replaces one document with another
- `SODA remove <collection_name> [-k | -klist | -f] {<key> | <k1> <k2> ...| <qbe>}` — Removes documents from collection. Optional arguments:
  - `k`: Removes document in collection matching the specific <*key*>
  - `klist`: Removes document in collection matching the list <*key1*> <*key2*> ...
  - `f`: Removes document in collection matching <*qbe*>