NoSQL Workbench for DynamoDB

NoSQL Workbench for Amazon DynamoDB is a cross-platform, client-side GUI application that you can use for modern database development and operations. It's available for Windows, macOS, and Linux. NoSQL Workbench is a visual development tool that provides data modeling, data visualization, and query development features to help you design, create, query, and manage DynamoDB tables. NoSQL Workbench now includes DynamoDB local as an optional part of the installation process, which makes it easier to model your data in DynamoDB local. To learn more about DynamoDB local and its requirements, see Setting up DynamoDB local (downloadable version) .

Data modeling

With NoSQL Workbench for DynamoDB, you can build new data models from, or design models based on, existing data models that satisfy your application's data access patterns. You can also import and export the designed data model at the end of the process. For more information, see Building data models with NoSQL Workbench.

Data visualization

The data model visualizer provides a canvas where you can map queries and visualize the access patterns (facets) of the application without having to write code. Every facet corresponds to a different access pattern in DynamoDB. You can autogenerate sample data for use in your data model. For more information, see Visualizing data access patterns.

Operation building

NoSQL Workbench provides a rich graphical user interface for you to develop and test queries. You can use the *operation builder* to view, explore, and query live datasets. The structured operation builder supports projection expression, condition expression, and generates sample code in multiple languages. You can directly clone tables from one Amazon DynamoDB account to another one in different Regions. You can also directly clone tables between DynamoDB local and an Amazon DynamoDB account for faster copying of your table's key schema (and optionally GSI schema and items) between your development environments. For more information, see Exploring datasets and building operations with NoSQL Workbench.

The video below details concepts of data modeling with NoSQL Workbench.

Topics

Download NoSQL Workbench for DynamoDB

Follow these instructions to download NoSQL Workbench and DynamoDB local* for Amazon DynamoDB.

Prerequisites

There are two prerequisite pieces of software required for Ubuntu installs: libfuse2 and curl.

libfuse2

As of Ubuntu 22.04, libfuse2 is no longer installed by default. To solve this, run `sudo add-apt-repository universe && sudo apt install libfuse2` to install for the newest Ubuntu version.

curl

Update Ubuntu, run `sudo apt update && sudo apt upgrade`

Next, install cURL, execute: `sudo apt install curl`

To download NoSQL Workbench and DynamoDB local

1. Download the appropriate version of NoSQL Workbench for your operating system.

Operating system	Download link
macOS (Intel)**	Download for macOS (Intel)

Operating system	Download link
macOS (Apple silicon)	Download for macOS (Apple silicon)
Windows	Download for Windows
Linux***	Download for Linux

* NoSQL Workbench includes DynamoDB local as an optional part of the installation process.

** If a warning message appears when you try to open NoSQL Workbench stating that the app isn't registered with Apple by an identified developer, do the following:

1. Locate the app and then open it.
2. Control+click the app icon, then choose Open from the shortcut menu.

   This saves the app as an exception to your security settings. Open the app by double-clicking it just as you can open any registered app.

*** NoSQL Workbench supports Ubuntu 12.04, Fedora 21, and Debian 8, or any newer versions of these Linux distributions.

2. Start the application that you downloaded, and then follow the steps in Install NoSQL Workbench.

> **Note**
>
> Java Runtime Environment (JRE) version 11.x or newer is required for running DynamoDB local.

Install NoSQL Workbench for DynamoDB

Follow these steps to install NoSQL Workbench and DynamoDB local on a supported platform.

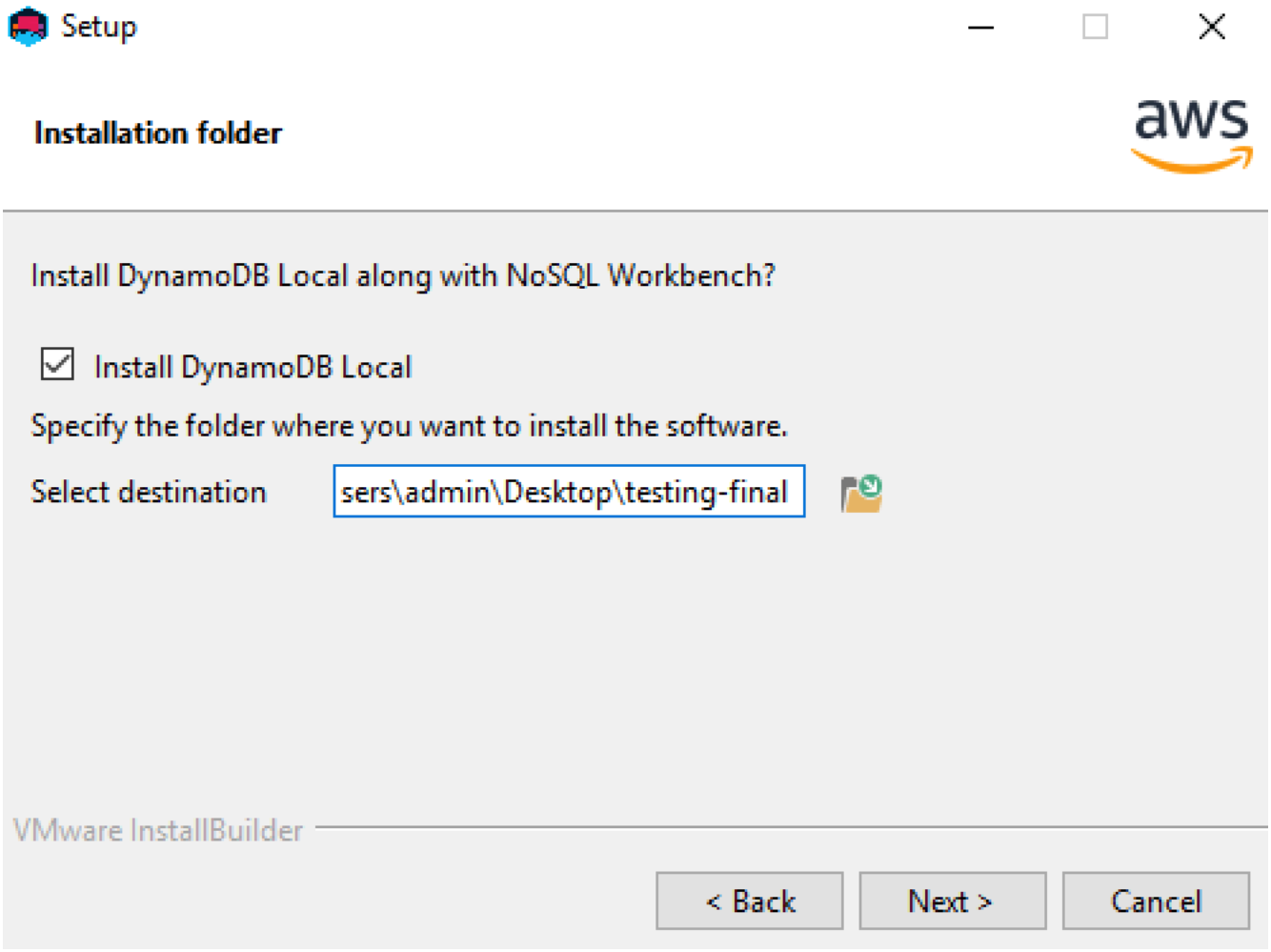
Windows

To install NoSQL Workbench on Windows

1. Run the NoSQL Workbench installer application and choose the setup language. Then choose **OK** to begin the setup. For more information about downloading NoSQL Workbench, see Download NoSQL Workbench for DynamoDB.
2. Choose **Next** to continue the setup, and then choose **Next** on the following screen.
3. By default, the **Install DynamoDB Local** check box is selected to include DynamoDB local as part of the installation. Keeping this option selected ensures that DynamoDB local will be installed, and the destination path will be the same as the installation path of NoSQL Workbench. Clearing the check box for this option will skip the installation of DynamoDB local, and the installation path will be for NoSQL Workbench only.

   Choose the destination where you want the software installed, and choose **Next**.

   > **Note**
   >
   > If you opted to not include DynamoDB local as part of the setup, clear the **Install DynamoDB Local** check box, choose **Next**, and skip to step 6. You can download DynamoDB local separately as a standalone installation at a later time. For more information, see Setting up DynamoDB local (downloadable version) .



4. Choose the port number for DynamoDB local to use. The default port is 8000. After you enter the port number, choose **Next**.
5. Choose **Next** to begin setup.
6. When the setup has completed, choose **Finish** to close the setup screen.
7. Open the application in your installation path, such as */programs/DynamoDBWorkbench/*.

macOS

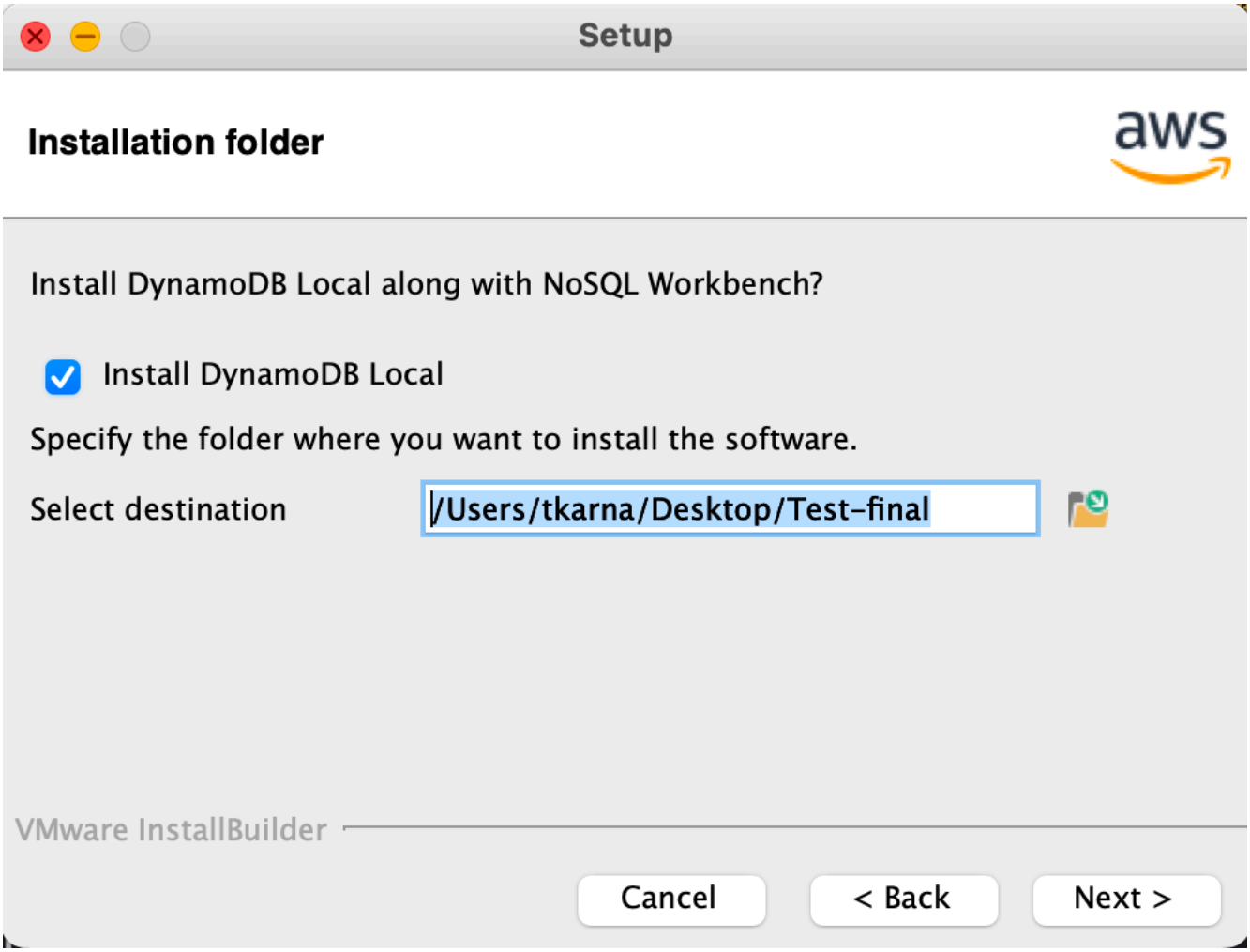
To install NoSQL Workbench on macOS

1. Run the NoSQL Workbench installer application and choose the setup language. Then choose **OK** to begin the setup. For more information about downloading NoSQL Workbench, see Download NoSQL Workbench for DynamoDB.

2. Choose **Next** to continue the setup, and then choose **Next** on the following screen.
3. By default, the **Install DynamoDB local** check box is selected to include DynamoDB local as part of the installation. Keeping this option selected ensures that DynamoDB local will be installed, and the destination path will be the same as the installation path of NoSQL Workbench. Clearing this option will skip the installation of DynamoDB local, and the installation path will be for NoSQL Workbench only.

   Choose the destination where you want the software installed, and choose **Next**.

   > **Note**
   >
   > If you opted to not include DynamoDB local as part of the setup, clear the **Install DynamoDB local** check box, choose **Next**, and skip to step 6. You can download DynamoDB local separately as a standalone installation at a later time. For more information, see Setting up DynamoDB local (downloadable version) .



4. Choose the port number for DynamoDB local to use. The default port is 8000. After you enter the port number, choose **Next**.
5. Choose **Next** to begin setup.
6. When the setup has completed, choose **Finish** to close the setup screen.
7. Open the application in your installation path, such as */Applications/DynamoDBWorkbench/*.

> **Note**
>
> NoSQL Workbench for macOS performs auto-updates. To get notification about updates, enable notification access to NoSQL Workbench in System Preferences > Notifications.

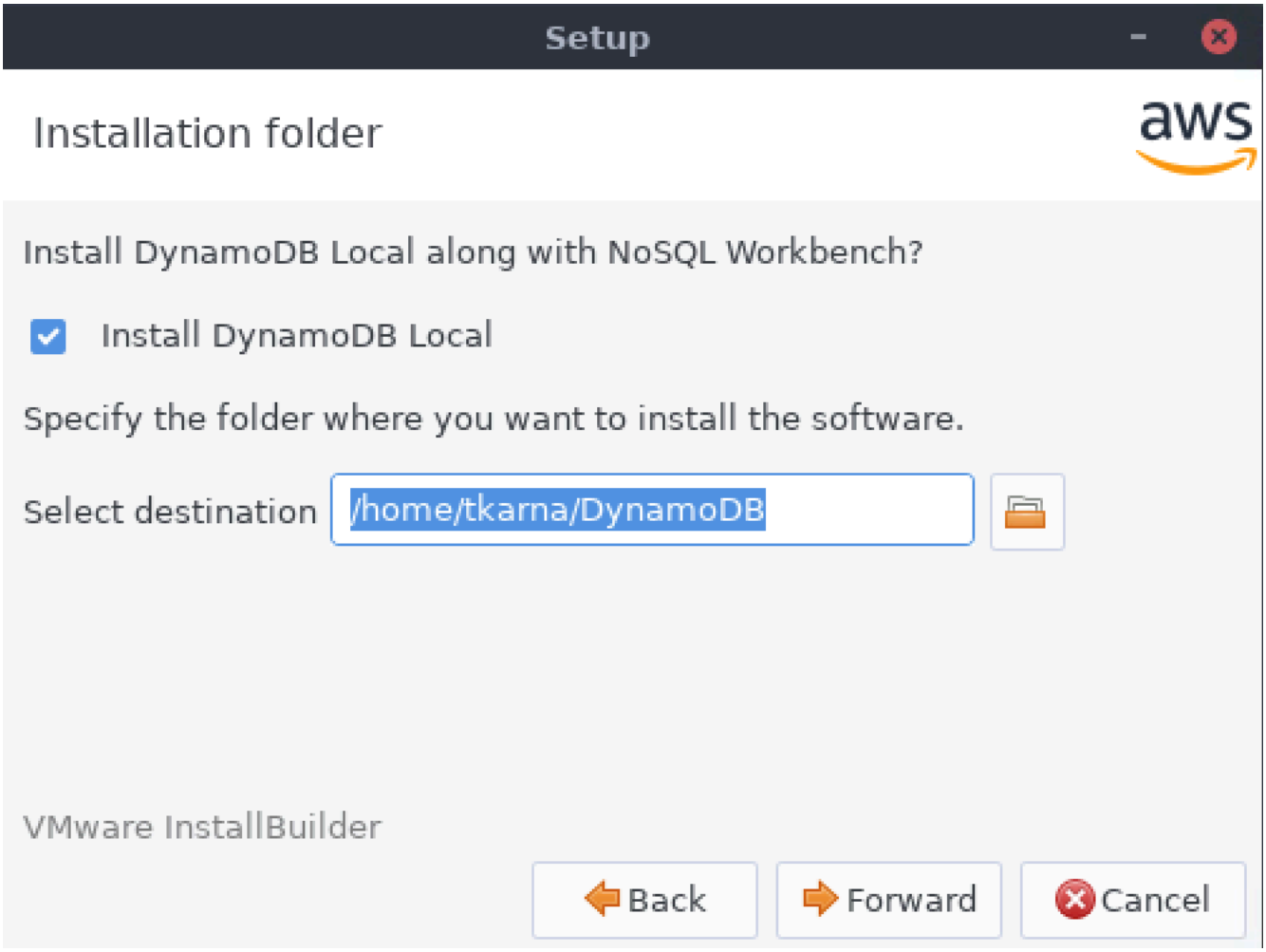
Linux

To install NoSQL Workbench on Linux

1. Run the NoSQL Workbench installer application and choose the setup language. Then choose **OK** to begin the setup. For more information about downloading NoSQL Workbench, see Download NoSQL Workbench for DynamoDB.
2. Choose **Forward** to continue the setup, and choose **Forward** on the following screen.
3. By default, the **Install DynamoDB local** check box is selected to include DynamoDB local as part of the installation. Keeping this option selected ensures that DynamoDB local will be installed, and the destination path will be the same as the installation path of NoSQL Workbench. Clearing this option will skip the installation of DynamoDB local, and the installation path will be for NoSQL Workbench only.

   Choose the destination where you want the software installed, and choose **Forward**.

   > **Note**
   >
   > If you opted to not include DynamoDB local as part of the setup, clear the **Install DynamoDB local** check box, choose **Forward**, and skip to step 6. You can download DynamoDB local separately as a standalone installation at a later time. For more information, see Setting up DynamoDB local (downloadable version) .



4. Choose the port number for DynamoDB local to use. The default port is 8000. After you enter the port number entered, choose **Forward**.
5. Choose **Forward** to begin setup.
6. When the setup has completed, choose **Finish** to close the setup screen.
7. Open the application in your installation path, such as */usr/local/programs/DynamoDBWorkbench/*.

To start an AppImage on Linux

1. Make the AppImage file executable:

   chmod +x noSQL-workbench-linux.AppImage

Replace noSQL-workbench-linux.AppImage with the actual file name of the AppImage you downloaded.

2. Run the AppImage:

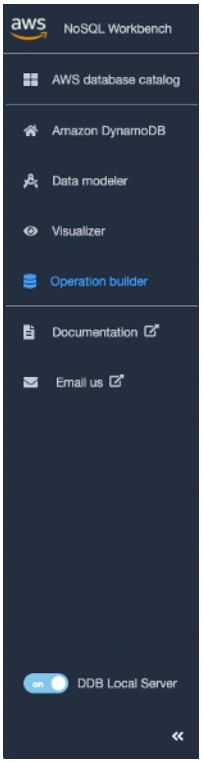
   ./noSQL-workbench-linux.AppImage

   This will launch the NoSQL Workbench application.

> **Note**
>
> Depending on your Linux distribution, you may need to install additional dependencies for the AppImage to run properly. If you encounter any issues, refer to the documentation provided by the AppImage developers or seek support from the community.

> **Note**
>
> If you opted to install DynamoDB local as part of the installation of NoSQL Workbench, DynamoDB local will be preconfigured with default options. To edit the default options, modify the *DDBLocalStart* script located in the */resources/DDBLocal_Scripts/* directory. You can find this in the path that you provided during installation. To learn more about DynamoDB local options, see DynamoDB local usage notes .

If you opted to install DynamoDB local as part of the NoSQL Workbench installation, you will have access to a toggle to enable and disable DynamoDB local as shown in the following image.



Building data models with NoSQL Workbench

You can use the data modeler tool in NoSQL Workbench for Amazon DynamoDB to build new data models, or to design models based on existing data models that satisfy your applications' data access patterns. The data modeler includes a few sample data models to help you get started.

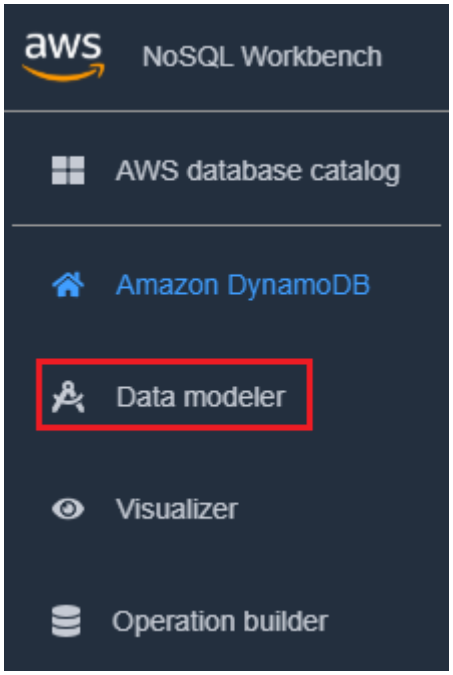
Topics

Creating a new data model

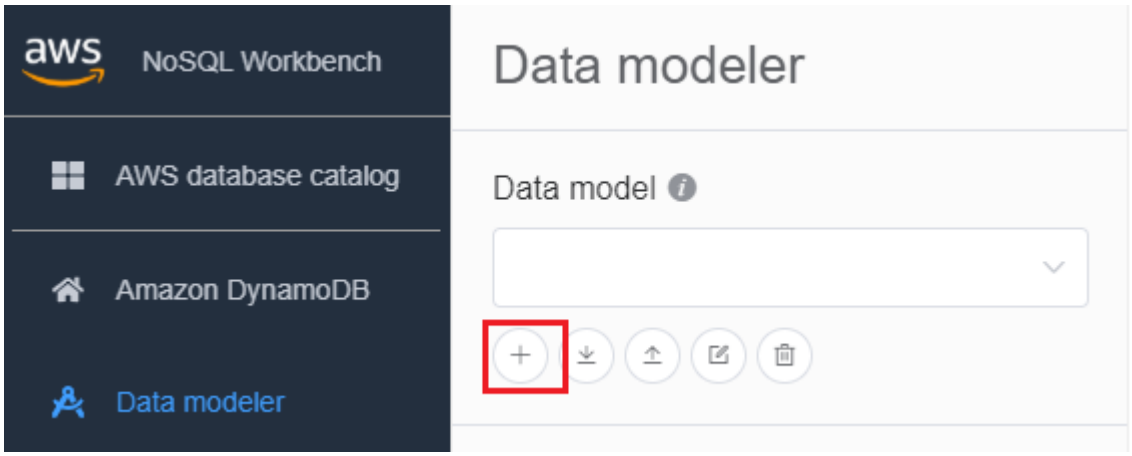
Follow these steps to create a new data model in Amazon DynamoDB using NoSQL Workbench.

To create a new data model

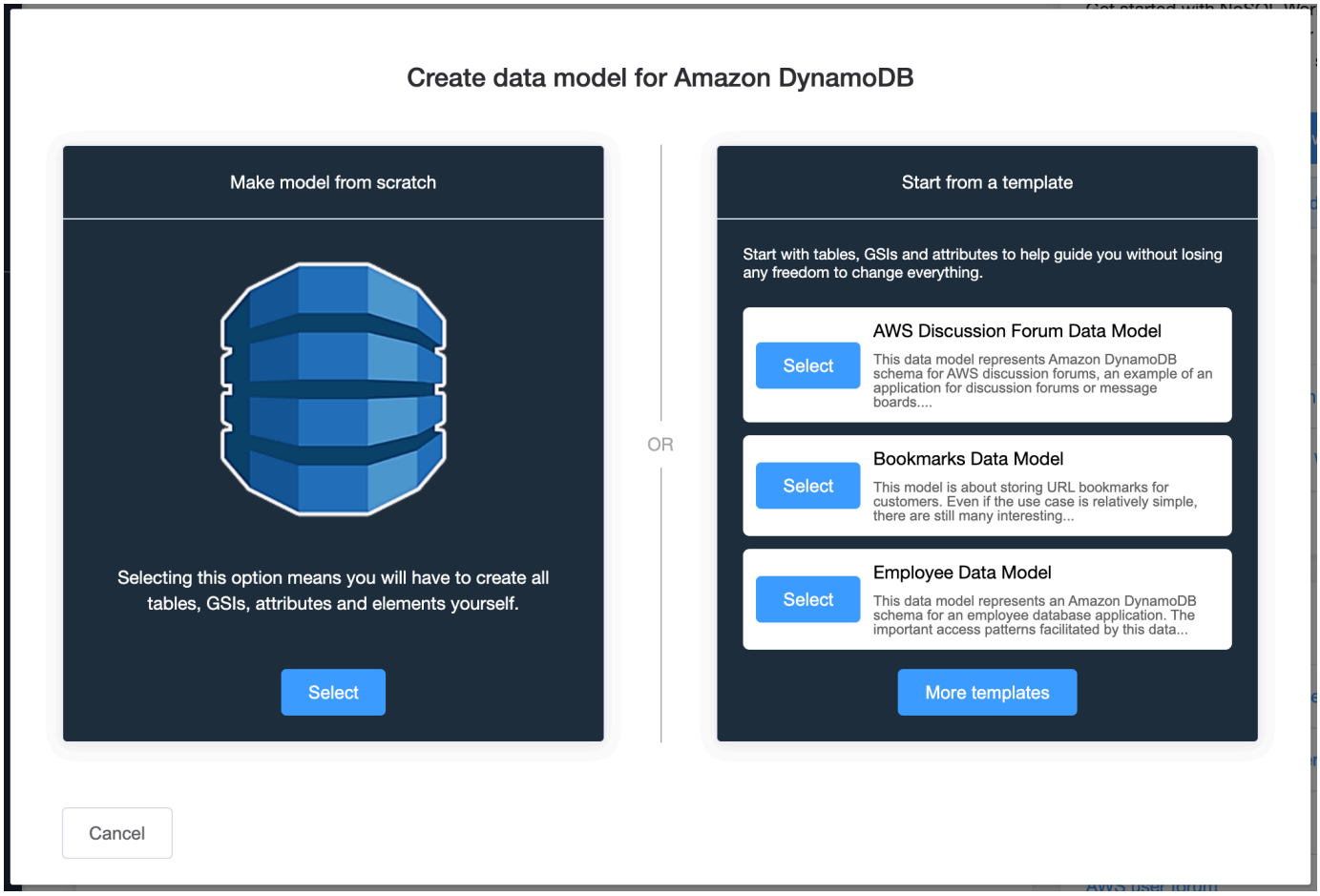
1. Open NoSQL Workbench, and in the navigation pane on the left side, choose the **Data modeler** icon.



2. Choose **Create data model**.

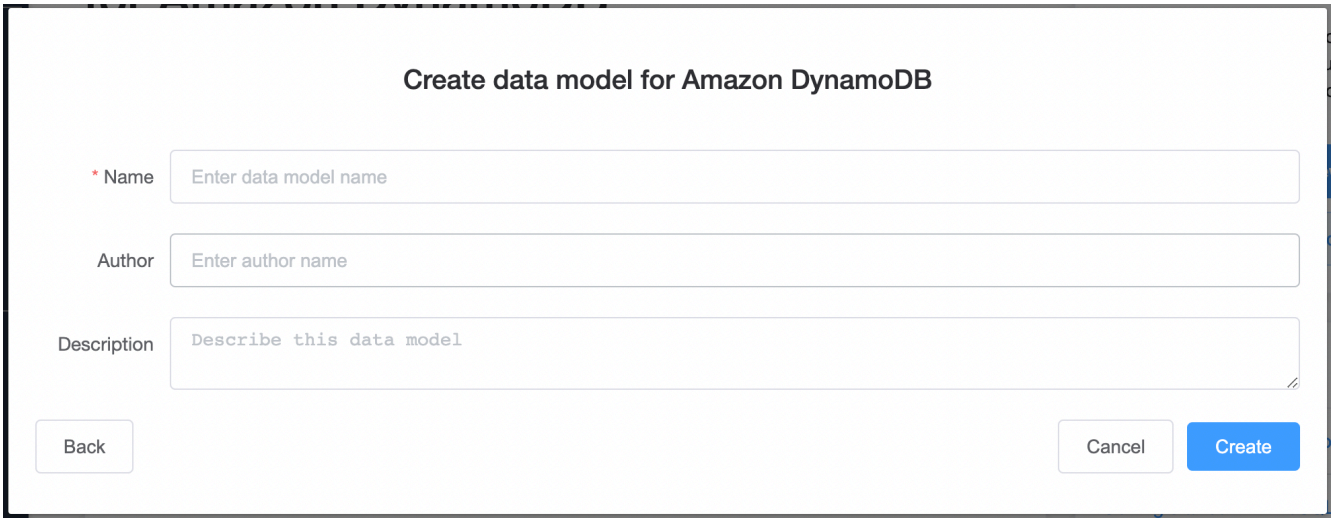


Create data model has two choices: Make model from scratch and Start from a template.



Make model from scratch

To make a model from scratch, enter a name, author, and description for the data model. Choose **Create** when finished.



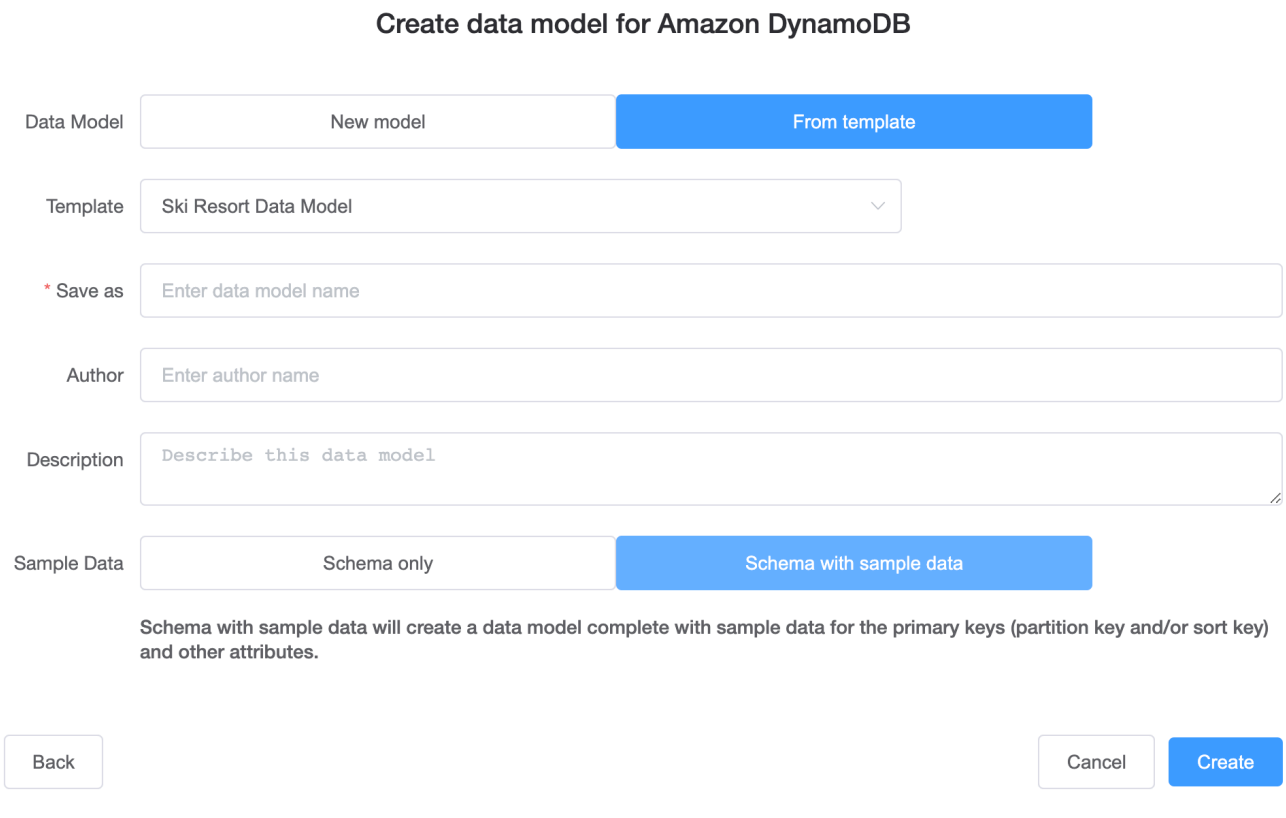
Start from a template

Starting from a template lets you choose a sample model to start from. Choose **More templates** to see more template options. Choose **Select** for the template that you want to use.

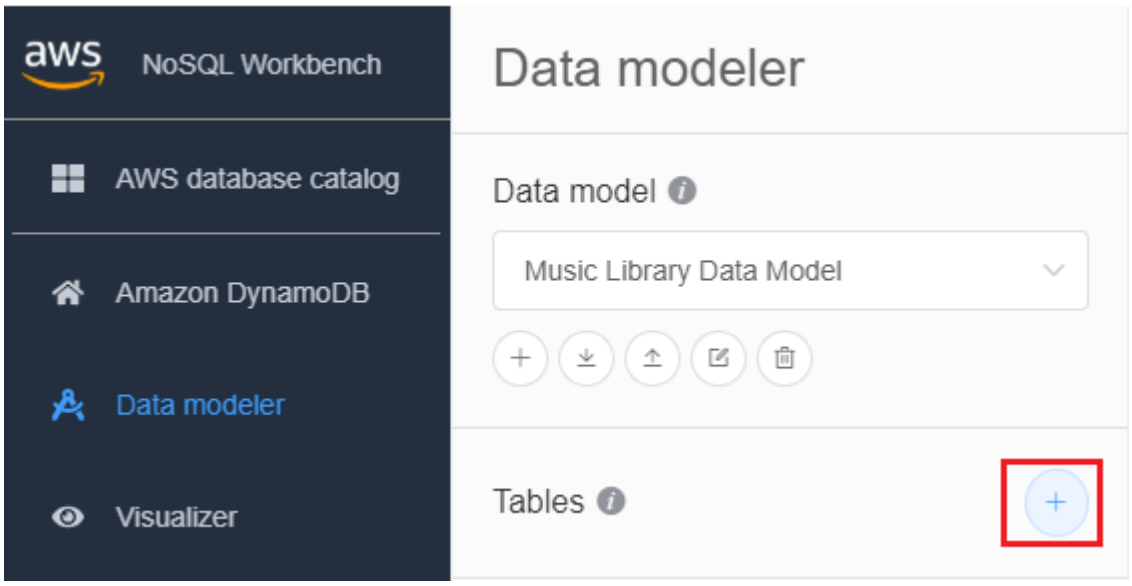
Enter a data model name, author, and description for the template you selected. You can choose between **Schema only** and **Schema with sample data**.

- **Schema only** creates an empty data model with the primary key (partition and sort key) and other attributes.
- **Schema with sample data** will create a data model complete with sample data for the primary key (partition and sort key) and other attributes.

When this information is complete, choose **Create** to create the model.



3. With the model created, choose **Add table**.



For more information about tables, see Working with tables in DynamoDB.

4. Specify the following:

   - **Table name** – Enter a unique name for the table.
   - **Partition key** – Enter a partition key name, and specify its type. Optionally, you can also select a more granular data type format for sample data generation.
   - If you want to add a sort key:
     1. Select **Add sort key**.
     2. Specify the sort key name and its type. Optionally, you can select a more granular data type format for sample data generation.

   > **Note**
   >
   > To learn more about primary key design, designing and using partition keys effectively, and using sort keys, see the following:
   >
   > - Primary key
   > - Best practices for designing and using partition keys effectively in DynamoDB
   > - Best practices for using sort keys to organize data in DynamoDB

5. To add other attributes, do the following for each attribute:

   1. Choose **Add an attribute**.

2. Specify the attribute name and its type. Optionally, you can select a more granular data type format for sample data generation.

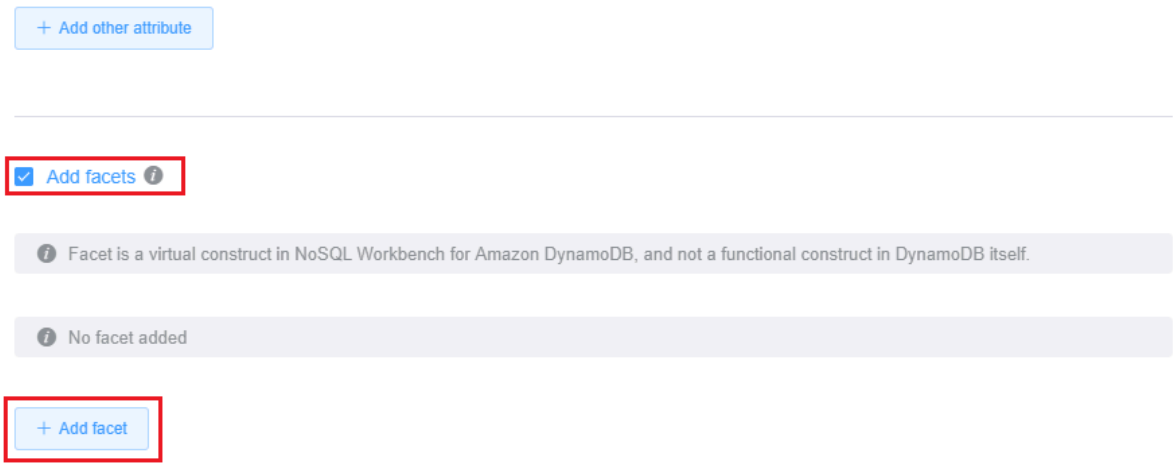
6. Add a facet:

You can optionally add a facet. A facet is a virtual construct in NoSQL Workbench. It is not a functional construct in DynamoDB itself.

> **Note**
>
> *Facets* in NoSQL Workbench help you visualize an application's different data access patterns for Amazon DynamoDB with only a subset of the data in a table. To learn more about facets, see Viewing data access patterns.

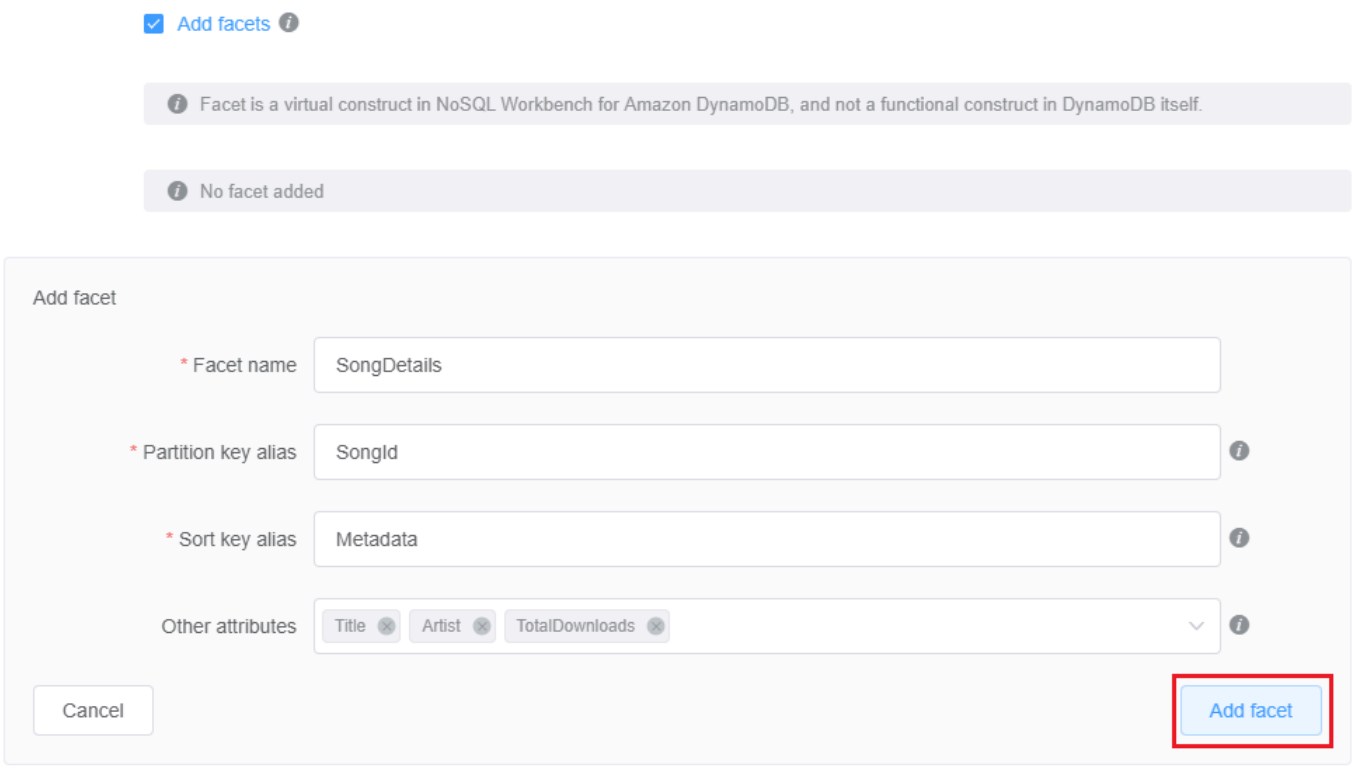
To add a facet,

- Select **Add facets**.
- Choose **Add facet**.



- Specify the following:
  - The **Facet name**.
  - A **Partition key alias** to help distinguish this facet view.
  - A **Sort key alias**.
  - Choose the **Other attributes** that are part of this facet.

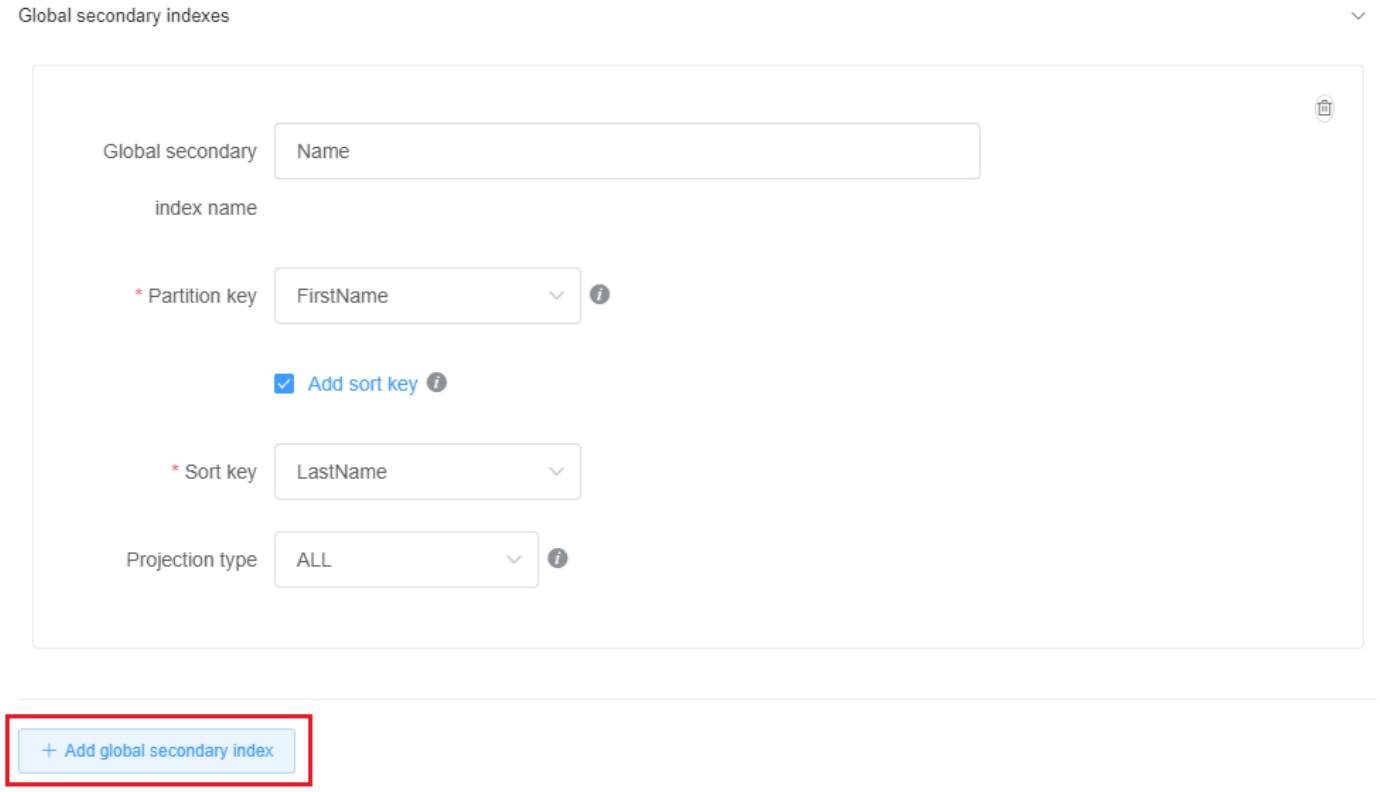
Choose **Add facet**.



Repeat this step if you want to add more facets.

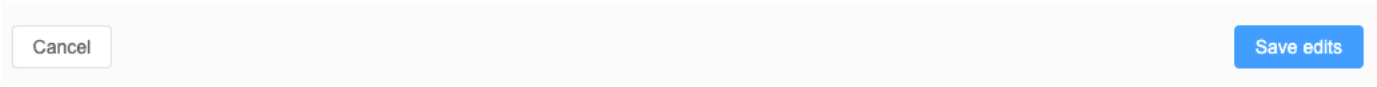
7. If you want to add a global secondary index, choose **Add global secondary index**.

   Specify the **Global secondary index name**, **Partition key** attribute, and **Projection type**.



For more information about working with global secondary indexes in DynamoDB, see Global secondary indexes.

8. Save the edits to your table settings..



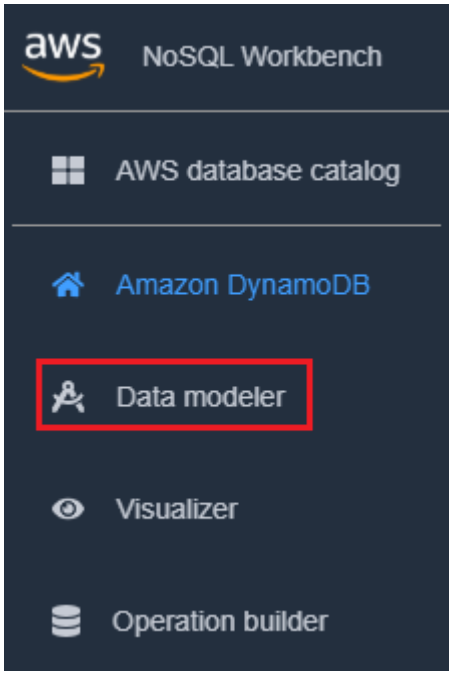
For more information about the `CreateTable` API operation, see CreateTable in the *Amazon DynamoDB API Reference*.

Importing an existing data model

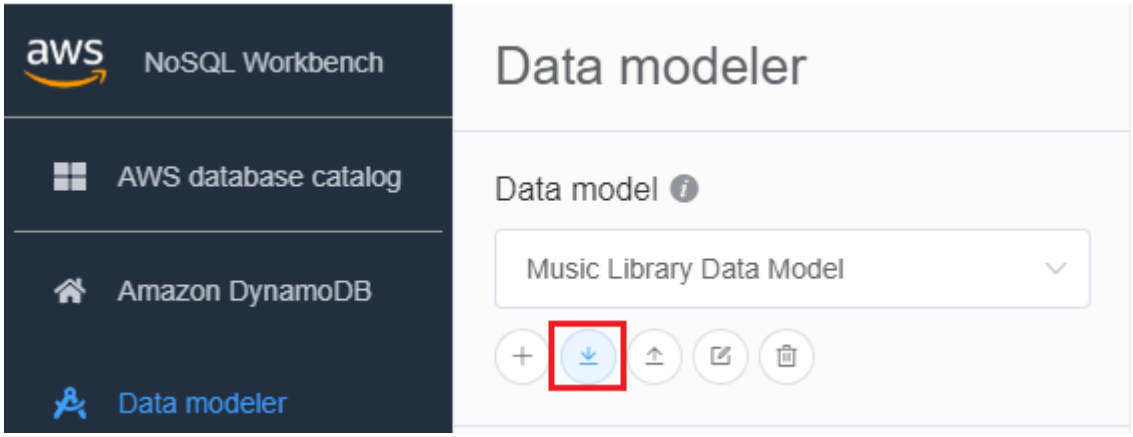
You can use NoSQL Workbench for Amazon DynamoDB to build a data model by importing and modifying an existing model. You can import data models in either NoSQL Workbench model format or in AWS CloudFormation JSON template format.

To import a data model

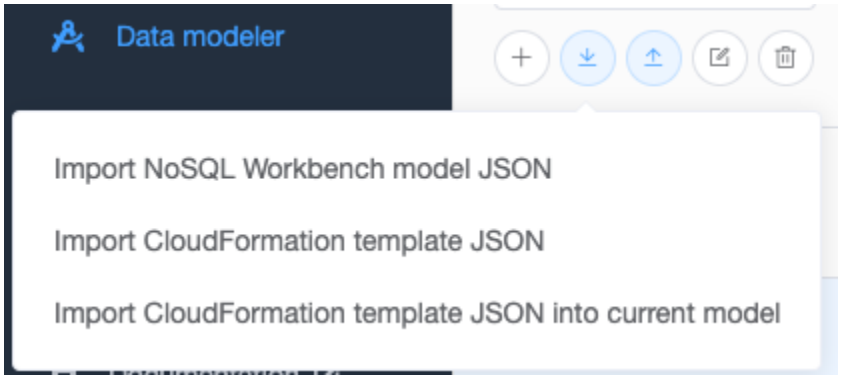
1. In NoSQL Workbench, in the navigation pane on the left side, choose the **Data modeler** icon.



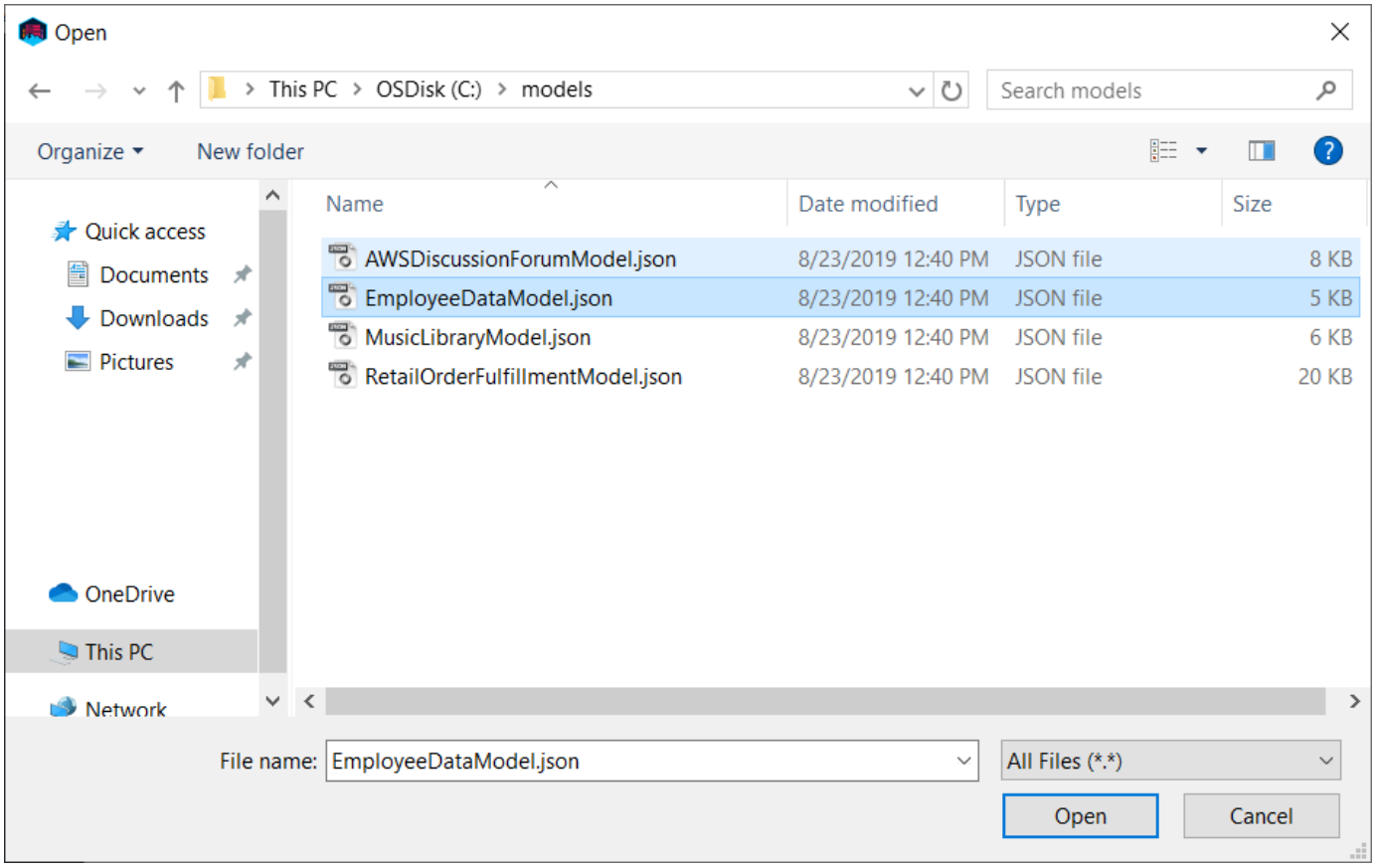
2. Hover your pointer over **Import data model**.



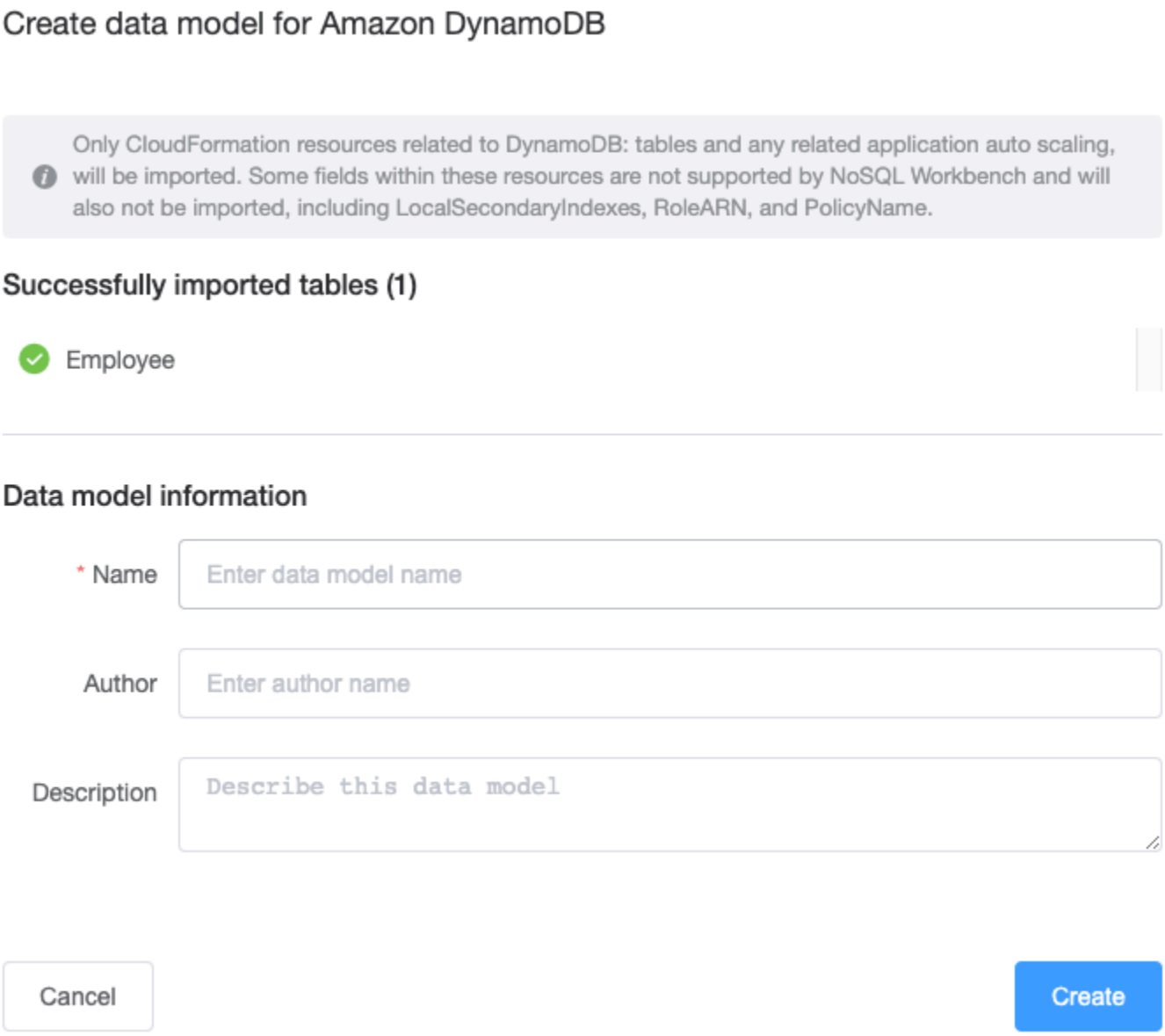
In the dropdown list, choose whether the model you want to import is in NoSQL Workbench model format or CloudFormation JSON template format. If you have an existing data model open in NoSQL Workbench, you'll have the option to import a CloudFormation template into the current model.



3. Choose a model to import.



4. If the model you're importing is in CloudFormation template format, you'll see a list of tables to be imported and have an opportunity to specify a data model name, author, and description.

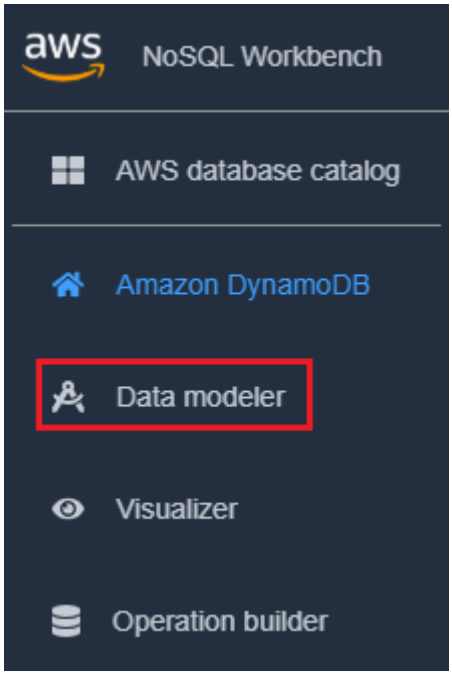


Exporting a data model

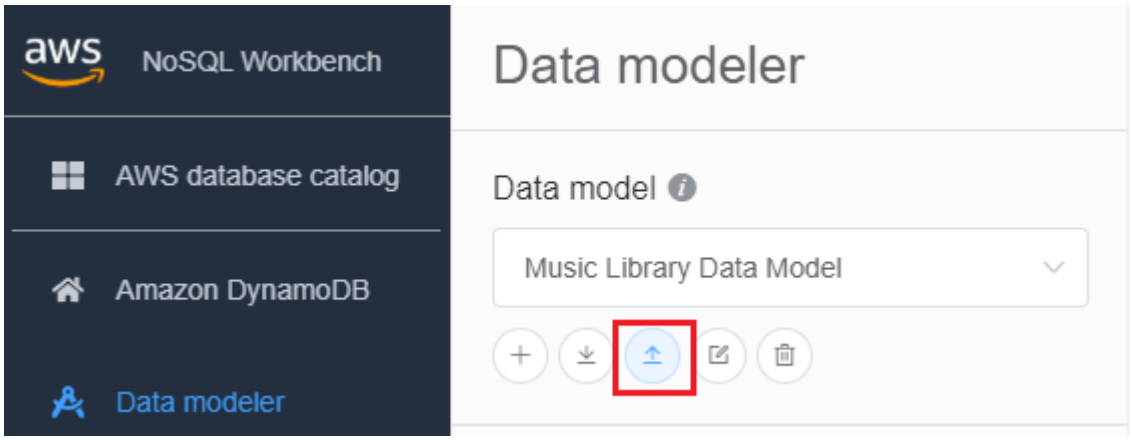
After you create a data model using NoSQL Workbench for Amazon DynamoDB, you can save and export the model in either NoSQL Workbench model format or AWS CloudFormation JSON template format.

To export a data model

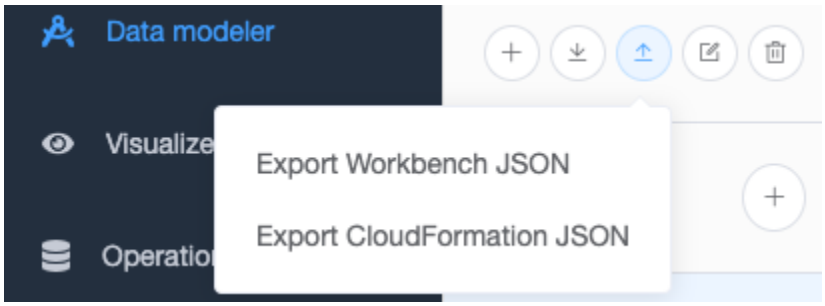
1. In NoSQL Workbench, in the navigation pane on the left side, choose the **Data modeler** icon.



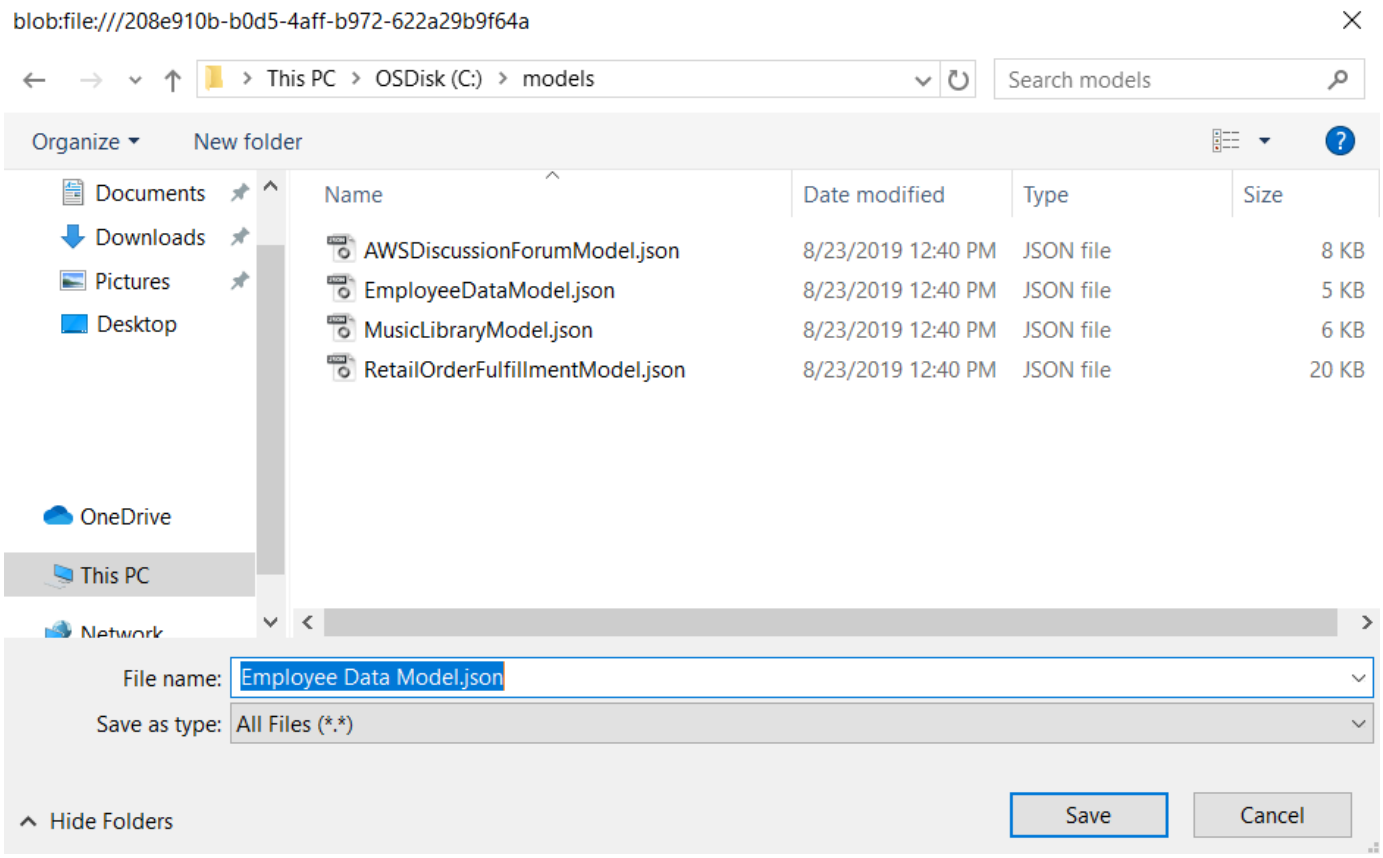
2. Hover your pointer over **Export data model**.



In the dropdown list, choose whether to export your data model in NoSQL Workbench model format or CloudFormation JSON template format.



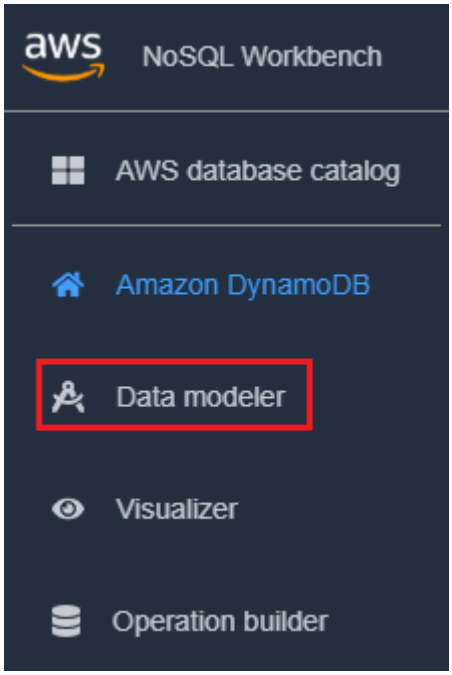
3. Choose a location to save your model.



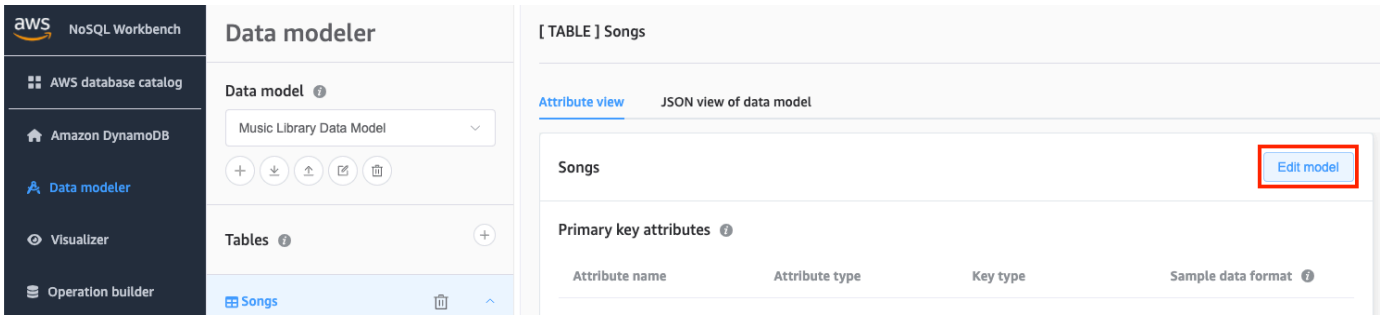
Editing an existing data model

To edit an existing model

1. In NoSQL Workbench, in the navigation pane on the left side, choose the **Data modeler** button.



2. Select the data model and choose the table that you want to edit. Choose **Edit model**



3. Make the needed edits, and then choose **Save edits**.

To manually edit an existing model and add a facet

1. Export your model. For more information, see Exporting a data model.
2. Open the exported file in an editor.
3. Locate the `DataModel` Object for the table that you want to create a facet for.

   Add a `TableFacets` array representing all the facets for the table.

   For each facet, add an object to the `TableFacets` array. Each array element has the following properties:

   - `FacetName` – A name for your facet. This value must be unique across the model.

- `PartitionKeyAlias` – A friendly name for the table's partition key. This alias is displayed when you view the facet in NoSQL Workbench.
- `SortKeyAlias` – A friendly name for the table's sort key. This alias is displayed when you view the facet in NoSQL Workbench. This property is not needed if the table has no sort key defined.
- `NonKeyAttributes` – An array of attribute names that are needed for the access pattern. These names must map to the attribute names that are defined for the table.

```
{
  "ModelName": "Music Library Data Model",
  "DataModel": [
    {
      "TableName": "Songs",
      "KeyAttributes": {
        "PartitionKey": {
          "AttributeName": "Id",
          "AttributeType": "S"
        },
        "SortKey": {
          "AttributeName": "Metadata",
          "AttributeType": "S"
        }
      },
      "NonKeyAttributes": [
        {
          "AttributeName": "DownloadMonth",
          "AttributeType": "S"
        },
        {
          "AttributeName": "TotalDownloadsInMonth",
          "AttributeType": "S"
        },
        {
          "AttributeName": "Title",
          "AttributeType": "S"
        },
        {
          "AttributeName": "Artist",
          "AttributeType": "S"
        },
        {
```

```
          "AttributeName": "TotalDownloads",
          "AttributeType": "S"
        },
        {
          "AttributeName": "DownloadTimestamp",
          "AttributeType": "S"
        }
      ],
      "TableFacets": [
        {
          "FacetName": "SongDetails",
          "KeyAttributeAlias": {
            "PartitionKeyAlias": "SongId",
            "SortKeyAlias": "Metadata"
          },
          "NonKeyAttributes": [
            "Title",
            "Artist",
            "TotalDownloads"
          ]
        },
        {
          "FacetName": "Downloads",
          "KeyAttributeAlias": {
            "PartitionKeyAlias": "SongId",
            "SortKeyAlias": "Metadata"
          },
          "NonKeyAttributes": [
            "DownloadTimestamp"
          ]
        }
      ]
    }
  ]
}
```

4. You can now import the modified model into NoSQL Workbench. For more information, see Importing an existing data model.

Visualizing data access patterns

You can use the visualizer tool in NoSQL Workbench for Amazon DynamoDB to map queries and visualize different access patterns (known as *facets*) of an application. Every facet corresponds to a different access pattern in DynamoDB. You can also manually add data to your data model or import data from MySQL.

Topics

Adding sample data to a data model

By adding sample data to your model, you can display data when visualizing the model and its various data access patterns, or *facets*.

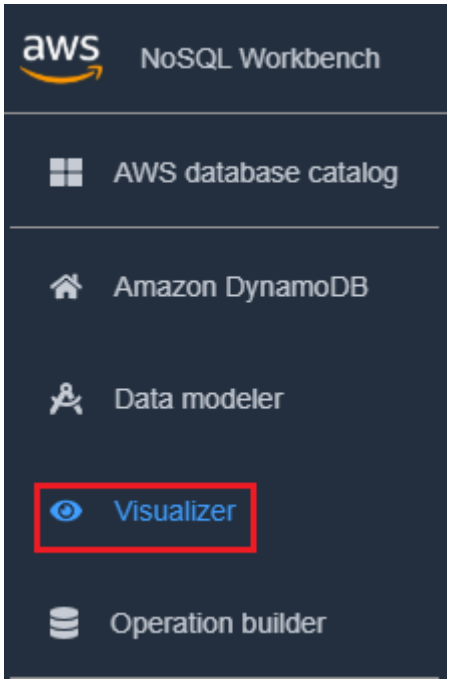
There are two ways to add sample data. One is using our sample data auto generation tool. The other is adding data one at a time.

Follow these steps to add sample data to a data model using NoSQL Workbench for Amazon DynamoDB.

To auto generate sample data

Auto generating sample data helps you generate between 1 to 5000 rows of data for immediate use. You can specify a granular sample data type to create realistic data based on your design and testing needs. To utilize the capability to generate realistic data, you need to specify the sample data type format for your attributes in the Data modeler. See Creating a new data modelfor specifying sample data type formats.

1. In the navigation pane on the left side, choose the **visualizer** icon.

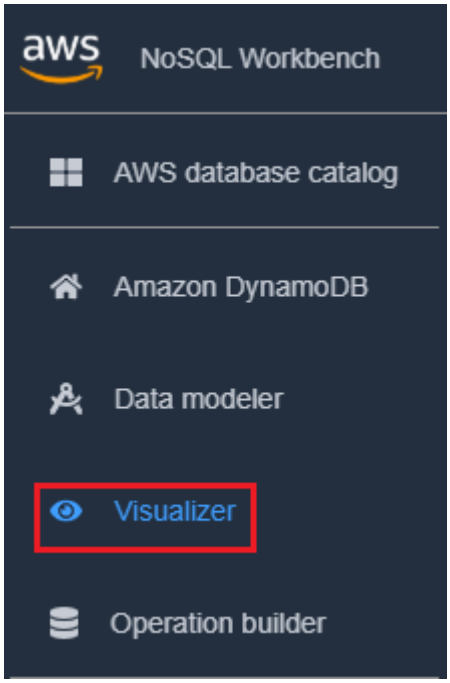


2. In the visualizer, select the data model and choose the table.
3. Choose the **Action** drop down, and select **Add sample data**.

4. Enter the number or items of sample data that you would like to generate, then select **Confirm**.

To add sample data one at a time

1. In the navigation pane on the left side, choose the **visualizer** icon.

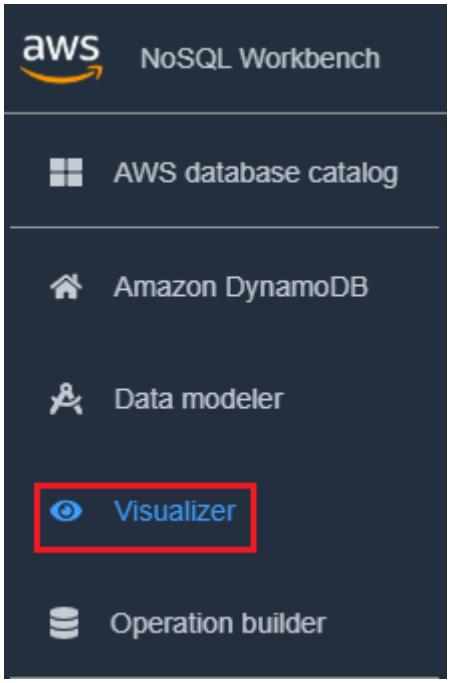


2. In the visualizer, select the data model and choose the table.
3. Choose the **Action** drop down, and select **Edit data**.

4. Choose **Add new row**. Enter the sample data into the empty text boxes, and choose **Add new row** again to add additional rows. When done choose **Save changes**.

To delete sample data

1. In the navigation pane on the left side, choose the **visualizer** icon.



2. In the visualizer, select the data model and choose the table.
3. Choose the **Action** drop down, and select **Edit data**.

4. Select the delete icon next to each row of data you want to delete.

Importing sample data from a CSV file

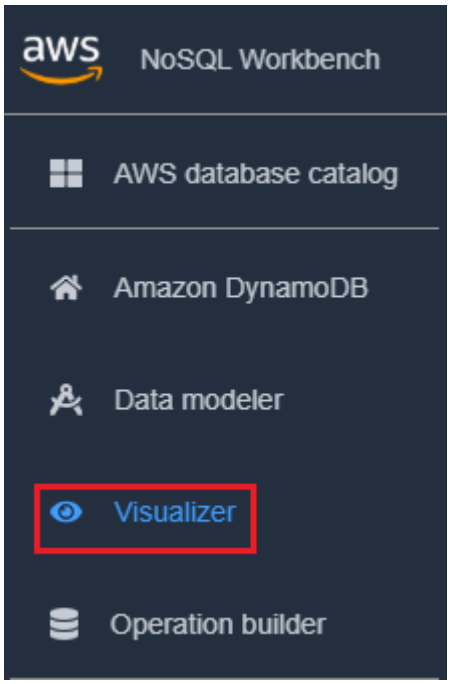
If you have preexisting sample data in a CSV file, you can import it into NoSQL Workbench. This enables you to quickly populate your model with sample data without having to enter it line by line.

The column names in the CSV file must match the attribute names in your data model, but they do not need to be in the same order. For example, if your data model has attributes called `LoginAlias`, `FirstName`, and `LastName`, your CSV columns could be `LastName`, `FirstName`, and `LoginAlias`.

Data import from a CSV file is limited to 150 rows at a time.

To import data from a CSV file into NoSQL Workbench

1. In the navigation pane on the left side, choose the **visualizer** icon.



2. In the visualizer, select the data model and choose the table.
3. Choose the **Action** drop down, and select **Edit Data**.
4. Choose the **Action** drop down again, and select **Import CSV file**.
5. Select your CSV file and choose **Open**. The data in the CSV file will be appended to your table.

> **Note**
>
> If your CSV file contains one or more rows that have the same keys as items already in your table, you will have the option of overwriting the existing items or appending them to the end of the table. If you choose to append the items, the suffix "-Copy" will be added to each duplicate item's key to differentiate the duplicate items from the items that were already in the table.

Viewing data access patterns

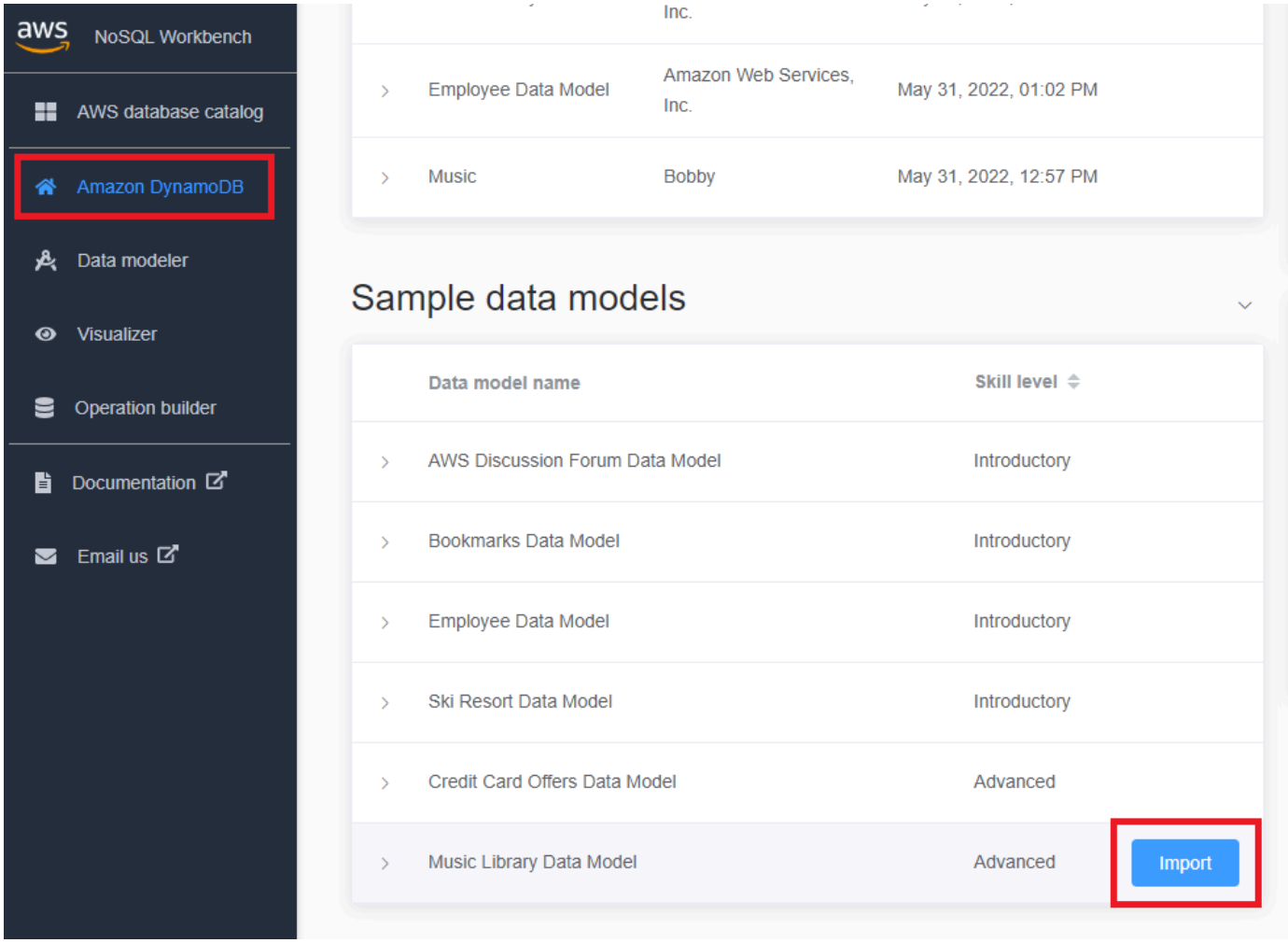
In NoSQL Workbench, *facets* represent an application's different data access patterns for Amazon DynamoDB. Facets can help you visualize your data model when multiple data types are represented by a sort key. Facets give you a way to view a subset of the data in a table, without

having to see records that don't meet the constraints of the facet. Facets are considered a visual data modeling tool, and don't exist as a usable construct in DynamoDB, as they are purely an aid to modeling of access patterns.

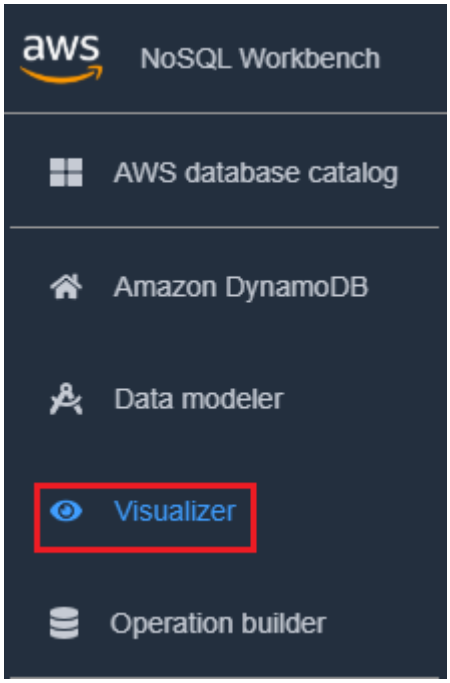
To see an example of facets, you can import one of our sample data models with facets as part of the data model template.

Import sample data model

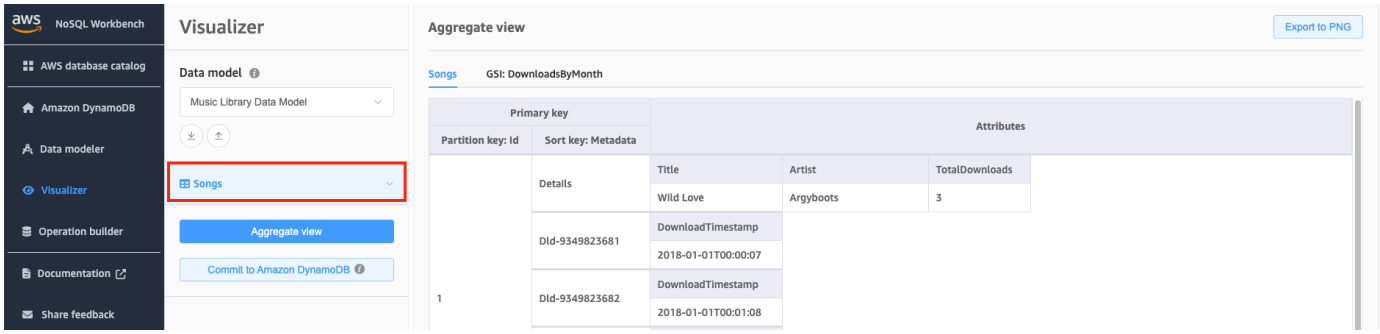
1. On the left, choose **Amazon DynamoDB**.
2. In the Sample data models section, hover your pointer over Music Library Data Model and choose **Import**.



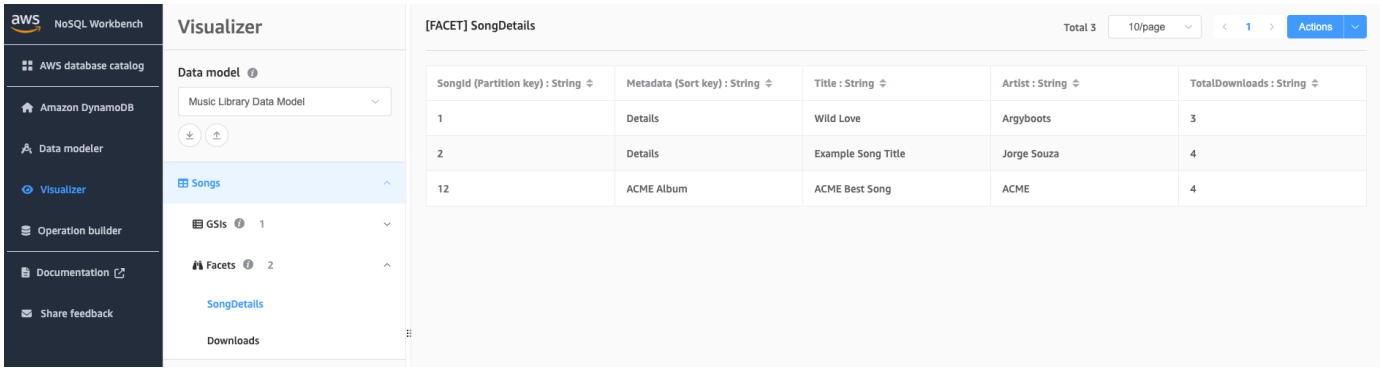
3. In the navigation pane on the left side, choose the **visualizer** icon.



4. Choose the Songs table to expand it. You'll be shown an aggregate view of your data.



5. Choose **Facets** drop-down arrow to expand the available facets.
6. Choose the SongDetails facet to visualize the data with the SongDetails facet applied.



You can also edit the facet definitions using the Data Modeler. For more information, see Editing an existing data model.

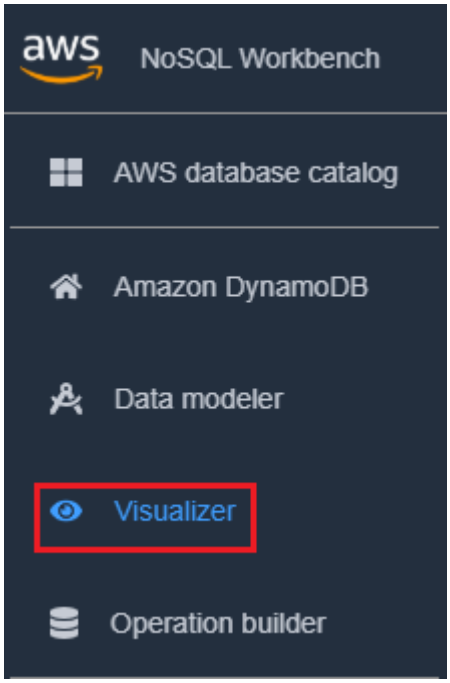
Viewing all tables in a data model using aggregate view

The aggregate view in NoSQL Workbench for Amazon DynamoDB represents all the tables in a data model. For each table, the following information appears:

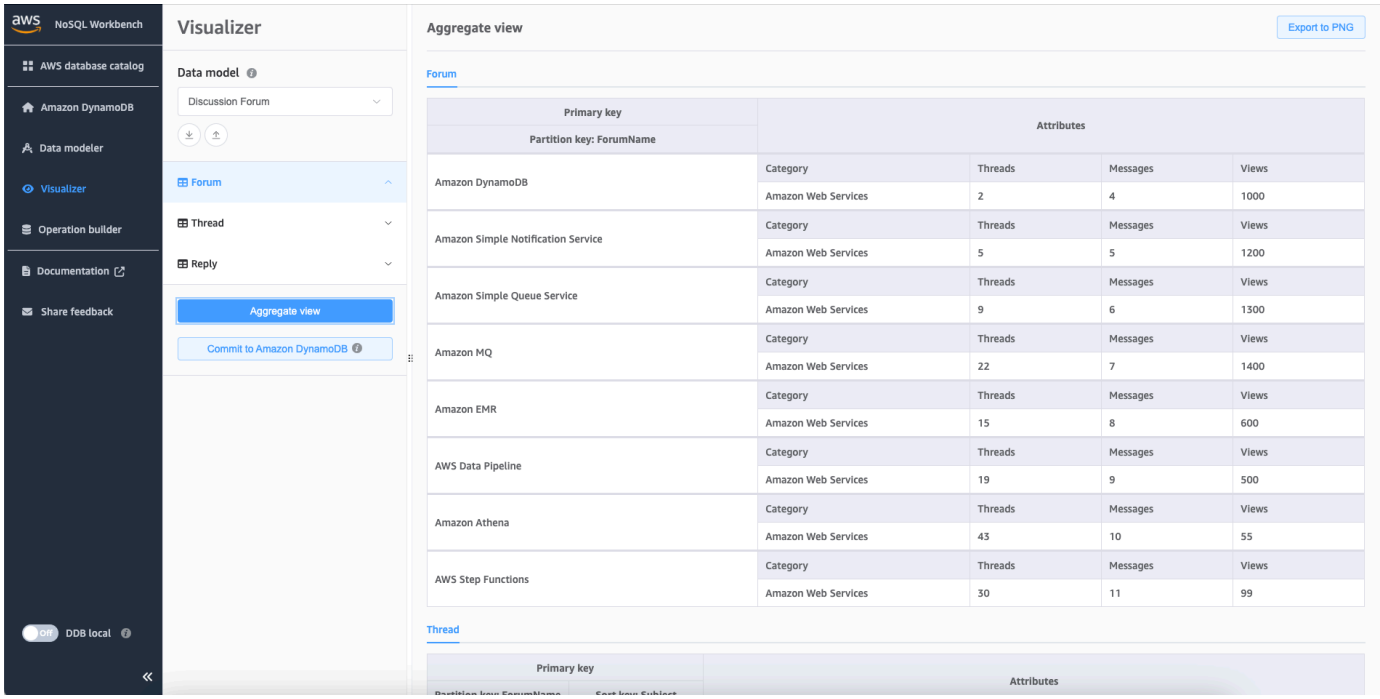
- Table column names
- Sample data
- All global secondary indexes that are associated with the table. The following information is displayed for each index:
  - Index column names
  - Sample data

To view all table information

1. In the navigation pane on the left side, choose the **visualizer** icon.



2. In the visualizer, choose **Aggregate view**.



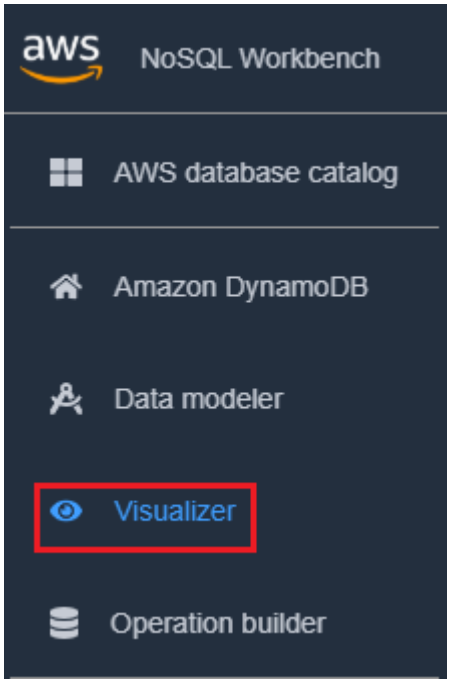
Committing a data model to DynamoDB

When you are satisfied with your data model, you can commit the model to Amazon DynamoDB.

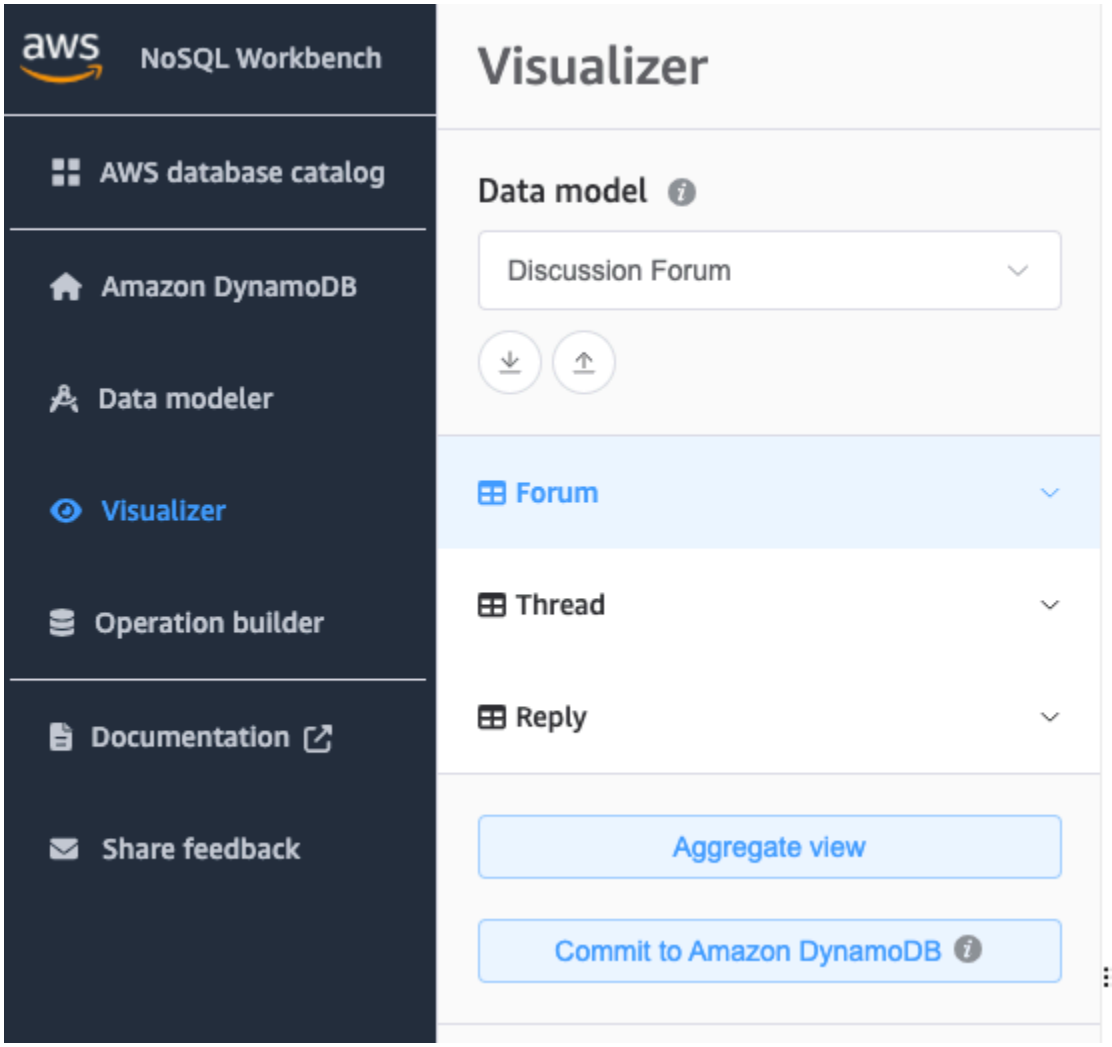
> **Note**
>
> - This action results in the creation of server-side resources in AWS for the tables and global secondary indexes represented in the data model.
> - Tables are created with the following characteristics:
>   - Auto scaling is set to 70 percent target utilization.
>   - Provisioned capacity is set to 5 read capacity units and 5 write capacity units.
> - Global secondary indexes are created with provisioned capacity of 10 read capacity units and 5 write capacity units.

To commit the data model to DynamoDB

1. In the navigation pane on the left side, choose the **visualizer** icon.



2. Choose **Commit to DynamoDB**.



3. Choose an already existing connection, or create a new connection by choosing the **Add new remote connection** tab.

   - To add a new connection, specify the following information:
     - **Account Alias**
     - **AWS Region**
     - **Access key ID**
     - **Secret access key**

     For more information about how to obtain the access keys, see Getting an AWS access key.
   - You can optionally specify the following:
     - **Session token**
     - **IAM role ARN**

- If you don't want to sign up for a free tier account, and prefer to use DynamoDB local (downloadable version):
  1. Choose the **Add a new DynamoDB local connection** tab.
  2. Specify the **Connection name** and **Port**.

4. Choose **Commit**.

> **Note**
>
> If you installed DynamoDB local as part of the NoSQL Workbench setup, you'll need to turn DynamoDB local on by using the **DynamoDB local Server** toggle at the bottom left of the NoSQL Workbench screen. See Install NoSQL Workbench for DynamoDB for more information on this toggle.

Exploring datasets and building operations with NoSQL Workbench

NoSQL Workbench for Amazon DynamoDB provides a rich graphical user interface for developing and testing queries. You can use the operation builder in NoSQL Workbench to view, explore, and query live datasets. The structured operation builder supports projection expression, condition expression, and generates sample code in multiple languages. You can directly clone tables from one Amazon DynamoDB account to another one in different Regions. You can also directly clone tables between DynamoDB local and an Amazon DynamoDB account for faster copying of your table's key schema (and optionally GSI schema and items) between your development environments.You can save as many as 50 DynamoDB data operations in the operation builder.

Topics

- Connecting to live datasets
- Building complex operations
- Cloning tables with NoSQL Workbench
- Exporting data to a CSV file

Connecting to live datasets

To connect to your Amazon DynamoDB tables with NoSQL Workbench, you must first connect to your AWS account.

To add a connection to your database

1. In NoSQL Workbench, in the navigation pane on the left side, choose the **Operation builder** icon.
2. Choose **Add connection**.
3. Specify the following information:

   - **Connection name**
   - **AWS Region**
   - **Access key ID**
   - **Secret access key**

   For more information about how to obtain the access keys, see Getting an AWS access key.

   You can optionally, specify the following:

   - **Session token**
   - **IAM role ARN**
4. Choose **Connect**.

   If you don't want to sign up for a free tier account, and prefer to use DynamoDB local (downloadable version):

   a. Choose the **Local** tab on the connection screen.

   b. Specify the following information:

      - **Connection name**
      - **Port**

   c. Choose the **connect** button.

> **Note**
>
> To connect to DynamoDB local, either manually launch DynamoDB local using your terminal (see deploying DynamoDB local on your computer) or launch DynamoDB local directly using the DDB local toggle in the NoSQL Workbench navigation menu. Ensure the connection port is the same as your DynamoDB local port.

5. On the created connection, choose **Open**.

After connecting to your DynamoDB database, the list of available tables appears in the left pane. Choose one of the tables to return a sample of the data stored in the table.

You can now run queries against the selected table.

To run queries on a table, see the next section on building operations see Building complex operations.

Building complex operations

The operation builder in NoSQL Workbench for Amazon DynamoDB provides a visual interface where you can perform complex data plane operations. It includes support for projection expressions and condition expressions. Once you've built an operation, you can save it for later use (up to 50 operations can be saved). You can then browse a list of your frequently used data-plane operations in the **Saved Operations** menu, and use them to automatically populate and build a new operation. You can also generate sample code for these operations, in multiple languages.

NoSQL Workbench supports building PartiQL for DynamoDB statements, which allows you to interact with DynamoDB using a SQL-compatible query language. NoSQL Workbench also supports building DynamoDB CRUD API operations.

To use NoSQL Workbench to build operations, in the navigation pane on the left side, choose the **Operation builder** icon.

Topics

- Building PartiQL statements
- Building API operations

Building PartiQL statements

To use NoSQL Workbench to build PartiQL for DynamoDB statements, choose **PartiQL editor** near the top of the NoSQL Workbench UI.

You can build the following PartiQL statement types in the operation builder.

Topics

Singleton statements

To run or generate code for a PartiQL statement, do the following.

1. Choose **PartiQL editor** near the top of the window.
2. Enter a valid PartiQL statement.
3. If your statement uses parameters:
    a. Choose **Optional request parameters**.
    b. Choose **Add new parameters**.
    c. Enter the attribute type and value.
    d. If you want to add additional parameters, repeat steps b and c.
4. If you want to generate code, choose **Generate code**.

    Select your desired language from the displayed tabs. You can now copy this code and use it in your application.
5. If you want the operation to be run immediately, choose **Run**.
6. If you want to save this operation for later use, choose **Save operation**. Then enter a name for your operation and choose **Save**.

Transactions

To run or generate code for a PartiQL transaction, do the following.

1. Choose **PartiQLTransaction** from the **More operations** dropdown.

2. Choose **Add a new statement**.
3. Enter a valid PartiQL statement.

> **Note**
>
> Read and write operations are not supported in the same PartiQL transaction request. A SELECT statement cannot be in the same request with INSERT, UPDATE, and DELETE statements. See Performing transactions with PartiQL for DynamoDB for more details.

4. If your statement uses parameters
   a. Choose **Optional request parameters**.
   b. Choose **Add new parameters**.
   c. Enter the attribute type and value.
   d. If you want to add additional parameters, repeat steps b and c.
5. If you want to add more statements, repeat steps 2 to 4.
6. If you want to generate code, choose **Generate code**.

   Select your desired language from the displayed tabs. You can now copy this code and use it in your application.
7. If you want the operation to be run immediately, choose **Run**.
8. If you want to save this operation for later use, choose **Save operation**. Then enter a name for your operation and choose **Save**.

Batch

To run or generate code for a PartiQL batch, do the following.

1. Choose **PartiQLBatch** from the **More operations** dropdown.
2. Choose **Add a new statement**.
3. Enter a valid PartiQL statement.

> **Note**
>
> Read and write operations are not supported in the same PartiQL batch request, which means a SELECT statement cannot be in the same request with INSERT, UPDATE, and

DELETE statements. Write operations to the same item are not allowed. As with the BatchGetItem operation, only singleton read operations are supported. Scan and query operations are not supported. See Running batch operations with PartiQL for DynamoDB for more details.

4. If your statement uses parameters:
   a. Choose **Optional request parameters**.
   b. Choose **Add new parameters**.
   c. Enter the attribute type and value.
   d. If you want to add additional parameters, repeat steps b and c.
5. If you want to add more statements, repeat steps 2 to 4.
6. If you want to generate code, choose **Generate code**.

   Select your desired language from the displayed tabs. You can now copy this code and use it in your application.
7. If you want the operation to be run immediately, choose **Run**.
8. If you want to save this operation for later use, choose **Save operation**. Then enter a name for your operation and choose **Save**.

Building API operations

To use NoSQL Workbench to build DynamoDB CRUD APIs, select **Operation builder** from the left of the NoSQL Workbench user interface.

Then select **Open** and choose a connection.

You can perform the following operations in the operation builder.

- Delete Table
- Create Table
- Update Table

- Put Item
- Update Item
- Delete Item

- Query
- Scan
- Transact Get Items
- Transact Write Items

Delete table

To run a `Delete Table` operation, do the following.

1. Find the table you want to delete from the **Tables** section.
2. Select **Delete Table** from the horizontal ellipsis menu.
3. Confirm you want to delete the table by entering the **Table name**.
4. Select **Delete**.

For more information about this operation, see Delete table in the *Amazon DynamoDB API Reference*.

Delete GSI

To run a `Delete GSI` operation, do the following.

1. Find the GSI of a table you want to delete from the **Tables** section.
2. Select **Delete GSI** from the horizontal ellipsis menu.
3. Confirm you want to delete the GSI by entering the **GSI name**.
4. Select **Delete**.

For more information about this operation, see Delete table in the *Amazon DynamoDB API Reference*.

Create table

To run a `Create Table` operation, do the following.

1. Choose the **+** icon next to the **Tables** section.
2. Enter the table name desired.
3. Create a partition key.

4. Optional: create a sort key.
5. To customize capacity settings, and uncheck the box next to **Use default capacity settings**.
   - You can now select either **Provisioned** or **On-demand capacity.**

     With Provisioned selected, you can set minimum and maximum read and write capacity units. You can also enable or disable auto scaling.
   - If the table is currently set to On-demand, you will be unable to specify a provisioned throughput.
   - If you switch from On-demand to Provisioned throughput, then Autoscaling will automatically be applied to all GSIs with: min: 1, max: 10; target: 70%.
6. Select **Skip GSIs and create** to create this table without a GSI. Optionally, you can select **Next** to create a GSI with this new table.

For more information about this operation, see Create table in the *Amazon DynamoDB API Reference*.

Create GSI

To run a `Create GSI` operation, do the following.

1. Find a table that you want to add a GSI to.
2. From the horizontal ellipsis menu, select **Create GSI**.
3. Name your GSI under **Index name**.
4. Create a partition key.
5. Optional: create a sort key.
6. Choose a projection type option from the dropdown.
7. Select **Create GSI**.

For more information about this operation, see Create table in the *Amazon DynamoDB API Reference*.

Update table

To update capacity settings for a table with an `Update Table` operation, do the following.

1. Find the table you want to update capacity settings for.

2. From the horizontal ellipsis menu, select **Update capacity settings**.
3. Select either **Provisioned** or **On-demand capacity.**

   With **Provisioned** selected, you can set minimum and maximum read and write capacity units. You can also enable or disable auto scaling.
4. Select **Update**.

For more information about this operation, see Update table in the *Amazon DynamoDB API Reference*.

Update GSI

To update capacity settings for a GSI with an `Update Table` operation, do the following.

> **Note**
>
> By default, global secondary indexes inherit the capacity settings of the base table. Global secondary indexes can have a different capacity mode only when the base table is in provisioned capacity mode. When you create a global secondary index on a provisioned mode table, you must specify read and write capacity units for the expected workload on that index. For more information, see Provisioned throughput considerations for Global Secondary Indexes.

1. Find the GSI you want to update capacity settings for.
2. From the horizontal ellipsis menu, select **Update capacity settings**.
3. You can now select either **Provisioned** or **On-demand capacity.**

   With **Provisioned** selected, you can set minimum and maximum read and write capacity units. You can also enable or disable auto scaling.
4. Select **Update**.

For more information about this operation, see Update table in the *Amazon DynamoDB API Reference*.

Put item

You create an item by using the `Put Item` operation. To run or generate code for a `Put Item` operation, do the following.

1. Find the table you want to create an item in.
2. From the **Actions** dropdown, select **Create item**.
3. Enter the partition key value.
4. Enter the sort key value, if one exists.
5. If you want to add non-key attributes, do the following:
   a. Select **+ Add other attributes**.
   b. Specify the **Attribute name**, **Type**, and **Value**.
6. If a condition expression must be satisfied for the `Put Item` operation to succeed, do the following:
   a. Choose **Condition**.
   b. Specify the attribute name, comparison operator, attribute type, and attribute value.
   c. If other conditions are needed, choose **Condition** again.

   For more information, see DynamoDB condition expression CLI example.
7. If you want to generate code, choose **Generate code**.

   Select your desired language from the displayed tabs. You can now copy this code and use it in your application.
8. If you want the operation to be run immediately, choose **Run**.
9. If you want to save this operation for later use, choose **Save operation**, then enter a name for your operation and choose **Save**.

For more information about this operation, see PutItem in the *Amazon DynamoDB API Reference*.

Update item

To run or generate code for an `Update Item` operation, do the following:

1. Find the table you want to update an item in.

2. Select the item.
3. Enter the attribute name and attribute value for the selected expression.
4. If you want to add more expressions, choose another expression in the **Update Expression** dropdown list, and then select the **+** icon.
5. If a condition expression must be satisfied for the `Update Item` operation to succeed, do the following:

   a. Choose **Condition**.

   b. Specify the attribute name, comparison operator, attribute type, and attribute value.

   c. If other conditions are needed, choose **Condition** again.

   For more information, see DynamoDB condition expression CLI example.
6. If you want to generate code, choose **Generate code**.

   Choose the tab for the language that you want. You can now copy this code and use it in your application.
7. If you want the operation to be run immediately, choose **Run**.
8. If you want to save this operation for later use, choose **Save operation**, then enter a name for your operation and choose **Save**.

For more information about this operation, see UpdateItem in the *Amazon DynamoDB API Reference*.

Delete item

To run a `Delete Item` operation, do the following.

1. Find the table you want to delete an item in.
2. Select the item.
3. From the **Actions** dropdown, select **Delete item**.
4. Confirm you want to delete the item by selecting **Delete**.

For more information about this operation, see DeleteItem in the *Amazon DynamoDB API Reference*.

Duplicate item

You can duplicate an item by creating a new item with the same attributes. To duplicate an item, do the following.

1. Find the table you want to duplicate an item in.
2. Select the item.
3. From the **Actions** dropdown, select **Duplicate item**.
4. Specify a new partition key.
5. Specify a new sort key (if necessary).
6. Select **Run**.

For more information about this operation, see DeleteItem in the *Amazon DynamoDB API Reference*.

Query

To run or generate code for a `Query` operation, do the following.

1. Select **Query** from the top of the NoSQL Workbench UI.
2. Specify the partition key value.
3. If a sort key is needed for the `Query` operation:
    a. Select **Sort key**.
    b. Specify the comparison operator, and attribute value.
4. Select **Query** to run this query operation. If more options are needed, check the **More options** checkbox and continue on with the following steps.
5. If not all the attributes should be returned with the operation result, select **Projection expression**.
6. Choose the **+** icon.
7. Enter the attribute to return with the query result.
8. If more attributes are needed, choose the **+** .
9. If a condition expression must be satisfied for the `Query` operation to succeed, do the following:
    a. Choose **Condition**.

   b. Specify the attribute name, comparison operator, attribute type, and attribute value.
   c. If other conditions are needed, choose **Condition** again.

   For more information, see DynamoDB condition expression CLI example.

10. If you want to generate code, choose **Generate code**.

    Choose the tab for the language that you want. You can now copy this code and use it in your application.

11. If you want the operation to be run immediately, choose **Run**.
12. If you want to save this operation for later use, choose **Save operation**, then enter a name for your operation and choose **Save**.

For more information about this operation, see Query in the *Amazon DynamoDB API Reference*.

Scan

To run or generate code for a `Scan` operation, do the following.

1. Select **Scan** from the top of the NoSQL Workbench UI.
2. Select the **Scan** button to perform this basic scan operation. If more options are needed, check the **More options** checkbox and continue on with the following steps.
3. Specify an attribute name to filter your scan results.
4. If not all the attributes should be returned with the operation result, select **Projection expression**.
5. If a condition expression must be satisfied for the scan operation to succeed, do the following:

   a. Choose **Condition**.
   b. Specify the attribute name, comparison operator, attribute type, and attribute value.
   c. If other conditions are needed, choose **Condition** again.

   For more information, see DynamoDB condition expression CLI example.

6. If you want to generate code, choose **Generate code**.

   Choose the tab for the language that you want. You can now copy this code and use it in your application.

7. If you want the operation to be run immediately, choose **Run**.
8. If you want to save this operation for later use, choose **Save operation**, then enter a name for your operation and choose **Save**.

TransactGetItems

To run or generate code for a `TransactGetItems` operation, do the following.

1. From the **More operations** dropdown at the top of the NoSQL Workbench UI, choose **TransactGetItems**.
2. Choose the **+** icon near **TransactGetItem**.
3. Specify a partition key.
4. Specify a sort key (if necessary).
5. Select **Run** to perform the operation, **Save operation** to save it, or **Generate code** to generate code for it.

For more information about transactions, see Amazon DynamoDB transactions.

TransactWriteItems

To run or generate code for a `TransactWriteItems` operation, do the following.

1. From the **More operations** dropdown at the top of the NoSQL Workbench UI, choose **TransactWriteItems**.
2. Choose an operation from the **Actions** dropdown.
3. Choose the **+** icon near **TransactWriteItem**.
4. In the **Actions** dropdown, choose the operation that you want to perform.

   - For `DeleteItem`, follow the instructions for the Delete item operation.
   - For `PutItem`, follow the instructions for the Put item operation.
   - For `UpdateItem`, follow the instructions for the Update item operation.

   To change the order of actions, choose an action in the list on the left side, and then choose the up or down arrows to move it up or down in the list.

   To delete an action, choose the action in the list, and then choose the **Delete** (trash can) icon.

5. Select **Run** to perform the operation, **Save operation** to save it, or **Generate code** to generate code for it.

For more information about transactions, see Amazon DynamoDB transactions.

Cloning tables with NoSQL Workbench

Cloning tables will copy a table's key schema (and optionally GSI schema and items) between your development environments. You can clone a table between DynamoDB local to an Amazon DynamoDB account, and even clone a table from one account to another in different Regions for faster experimentation.

To clone a table

1. In the **Operation Builder**, select your connection and Region (Region selection is not available for DynamoDB local).
2. Once you are connected to DynamoDB, browse your tables and select the table you want to clone.
3. From the horizontal ellipsis menu, select the **Clone** option.
4. Input your clone destination details:
    a. Select a connection.
    b. Select a Region (Region is not available for DynamoDB local).
    c. Enter a new table name.
    d. Choose a clone option:
        i. **Key schema** is selected by default and cannot be unselected. By default, cloning a table will copy your primary key and sort key if they are available.
        ii. **GSI schema** is selected by default if your table to be cloned has a GSI. Cloning a table will copy your GSI primary key and sort key if they are available. You have the option to deselect GSI schema to skip cloning the GSI schema. Cloning a table will copy your base table's capacity settings as the GSI's capacity settings. You can use the `UpdateTable` operation in Operation Builder to update the table's GSI capacity setting after cloning is complete.

5. Enter the number of items to clone. To only clone the key schema and optionally the GSI schema, you can keep the **Items to clone** value at 0. The maximum number of items that can be cloned is 5000.
6. Choose a capacity mode:
   a. **On-demand mode** is selected by default. DynamoDB on-demand offers pay-per-request pricing for read and write requests so that you pay only for what you use. To learn more, see DynamoDB On-demand mode .
   b. **Provisioned mode** lets you specify the number of reads and writes per second that you require for your application. You can use auto scaling to adjust your table's provisioned capacity automatically in response to traffic changes. To learn more, see DynamoDB Provisioned mode.
7. Select **Clone** to begin cloning.
8. The cloning process will run in the background. The **Operation builder** tab will show a notification when there is a change in the cloning table status. You can access this status by selecting the **Operation builder** tab and then selecting the arrow button. The arrow button is located on the cloning table status widget located near the bottom of the menu sidebar.

Exporting data to a CSV file

You can export the results of a query from Operation Builder to a CSV file. This enables you to load the data into a spreadsheet or process it using your preferred programming language.

Exporting to CSV

1. In the Operation Builder, run an operation of your choice, such as a Scan or Query.

   Note

   - You can only export results from read API operations and PartiQL statements to a CSV file. You can't export results from transaction read statements.
   - Currently, you can export results one page at a time to a CSV file. If there are multiple pages of results, you must export each page individually.

2. Select the items you want to export from the results.
3. In the **Actions** dropdown, choose **Export as CSV**.

4. Choose a filename and location for your CSV file and select **Save**.

Sample data models for NoSQL Workbench

The home page for the modeler and visualizer display a number of sample models that ship with the NoSQL Workbench. This section describes these models and their potential uses.

Topics

Employee data model

This data model is an introductory model. It represents an employee's basic details such as a unique alias, first name, last name, designation, manager, and skills.

This data model depicts a few techniques such as handling complex attribute such as having more than one skill. This model is also an example of one-to-many relationship through the manager and their reporting employees that has been achieved by the secondary index DirectReports.

The access patterns facilitated by this data model are:

- Retrieval of an employee record using the employee's login alias, facilitated by a table called `Employee`.
- Search for employees by name, facilitated by the Employee table's global secondary index called `Name`.
- Retrieval of all direct reports of a manager using the manager's login alias, facilitated by the Employee table's global secondary index called `DirectReports`.

Discussion forum data model

This data model represents a discussion forums. Using this model customers can engage with the developer community, ask questions, and respond to other customers' posts. Each AWS service has a dedicated forum. Anyone can start a new discussion thread by posting a message in a forum, and each thread receives any number of replies.

The access patterns facilitated by this data model are:

- Retrieval of a forum record using the forum's name, facilitated by a table called `Forum`.
- Retrieval of a specific thread or all threads for a forum, facilitated by a table called `Thread`.
- Search for replies using the posting user's email address, facilitated by the Reply table's global secondary index called `PostedBy-Message-Index`.

Music library data model

This data model represents a music library that has a large collection of songs and showcases its most downloaded songs in near-real time.

The access patterns facilitated by this data model are:

- Retrieval of a song record, facilitated by a table called `Songs`.
- Retrieval of a specific download record or all download records for a song, facilitated by a table called `Songs`.
- Retrieval of a specific monthly download count record or all monthly download count records for a song, facilitated by a table called `Song`.
- Retrieval of all records (including song record, download records, and monthly download count records) for a song, facilitated by a table called `Songs`.
- Search for most downloaded songs, facilitated by the Songs table's global secondary index called `DownloadsByMonth`.

Ski resort data model

This data model represents a ski resort that has a large collection of data for each ski lift collected daily.

The access patterns facilitated by this data model are:

- Retrieval of all data for a given ski lift or overall resort, dynamic and static, facilitated by a table called `SkiLifts`.
- Retrieval of all dynamic data (including unique lift riders, snow coverage, avalanche danger, and lift status) for a ski lift or the overall resort on a specific date, facilitated by a table called `SkiLifts`.
- Retrieval of all static data (including if the lift is for experienced riders only, vertical feet the lift rises, and lift riding time) for a specific ski lift, facilitated by a table called `SkiLifts`.
- Retrieval of date of data recorded for a specific ski lift or the overall resort sorted by total unique riders, facilitated by the SkiLifts table's global secondary index called `SkiLiftsByRiders`.

Credit card offers data model

This data model is used by a Credit Card Offers Application.

A credit card provider produces offers over time. These offers include balance transfers without fees, increased credit limits, lower interest rates, cash back, and airline miles. After a customer accepts or declines these offers, the respective offer status is updated accordingly.

The access patterns facilitated by this data model are:

- Retrieval of account records using `AccountId`, as facilitated by the main table.
- Retrieval of all the accounts with few projected items, as facilitated by the secondary index `AccountIndex`.
- Retrieval of accounts and all the offer records associated with those accounts by using `AccountId`, as facilitated by the main table.
- Retrieval of accounts and specific offer records associated with those accounts by using `AccountId` and `OfferId`, as facilitated by the main table.
- Retrieval of all `ACCEPTED/DECLINED` offer records of specific `OfferType` associated with accounts using `AccountId`, `OfferType`, and `Status`, as facilitated by the secondary index `GSI1`.
- Retrieval of offers and associated offer item records using `OfferId`, as facilitated by the main table.

Bookmarks data model

This data model is used store bookmarks for customers.

A customer can have many bookmarks and a bookmark can belong to many customers. This data model represents a many-to-many relationship.

The access patterns facilitated by this data model are:

- A single query by `customerId` can now return customer data as well as bookmarks.
- A query `ByEmail` index returns customer data by email address. Note that bookmarks are not retrieved by this index.
- A query `ByUrl` index gets bookmarks data by URL. Note that we have customerId as the sort key for the index because the same URL can be bookmarked by multiple customers.
- A query `ByCustomerFolder` index gets bookmarks by folder for each customer.

Release history for NoSQL Workbench

The following table describes the important changes in each release of the *NoSQL Workbench* client tool.

Version	Change	Description	Date
3.13.5	Capacity mode for default table settings is now on-demand	When you create a table with default settings, DynamoDB creates a table that uses on-demand capacity mode instead of provision ed capacity mode.	February 24, 2025
3.13.0	NoSQL Workbench operation builder improvements	NoSQL Workbench now includes native support for dark mode. Improved table and item operations in the operations builder. Item results and operation builder	April 24, 2024

Version	Change	Description	Date
		request information is available in JSON format.	
3.12.0	Cloning tables with NoSQL Workbench and returning capacity consumed	You can now clone tables between DynamoDB local and a DynamoDB web service account or between DynamoDB web service accounts for faster developme nt iterations. View RCU or WCU consumed after running an operation using the Operations Builder. We fixed the overwrite data issue when importing from a CSV file.	February 26, 2024
3.11.0	DynamoDB local improvements	You can now specify port when launching the built-in DynamoDB local instance. NoSQL Workbench can now be installed on Windows without admin rights. We have updated the data model templates.	January 17, 2024

Version	Change	Description	Date
3.10.0	Native support for Apple silicon	NoSQL Workbench now includes native support for Mac with Apple silicon. You can now configure sample data generation format for attributes of type `Number`.	December 5, 2023
3.9.0	Data modeler improvements	Visualizer now supports committin g data models to DynamoDB local with the option to overwrite existing tables.	November 3, 2023
3.8.0	Sample data generation	NoSQL Workbench now supports generating sample data for your DynamoDB data models.	September 25, 2023
3.6.0	Improvements in the Operations builder	Connections management improvements in the Operations builder. Attribute names in Data Modeler can now be changed without deleting data. Other bug fixes.	April 11, 2023

Version	Change	Description	Date
3.5.0	Support for new AWS Regions	NoSQL Workbench now supports the ap-south-2, ap-southeast-3, ap-southeast-4, eu-central-2, eu-south-2, me-central-1, and me-west-1 regions.	February 23, 2023
3.4.0	Support for DynamoDB local	NoSQL Workbench now supports installing DynamoDB local as part of the installation process.	December 6, 2022
3.3.0	Support for control plane operations	Operation Builder now supports control plane operations.	June 1, 2022
3.2.0	CSV import and export	You can now import sample data from a CSV file in the Visualizer tool, and also export the results of an Operation Builder query to a CSV file.	October 11, 2021
3.1.0	Save operations	The Operation Builder in NoSQL Workbench now supports saving operations for later use.	July 12, 2021

Version	Change	Description	Date
3.0.0	Capacity settings and CloudFormation import/export	NoSQL Workbench for Amazon DynamoDB now supports specifying a read/write capacity mode for tables, and can now import and export data models in CloudFormation format.	April 21, 2021
2.2.0	Support for PartiQL	NoSQL Workbench for Amazon DynamoDB adds support for building PartiQL statements for DynamoDB.	December 4, 2020
1.1.0	Support for Linux.	NoSQL Workbench for Amazon DynamoDB is supported on Linux—Ubuntu, Fedora, and Debian.	May 4, 2020
1.0.0	NoSQL Workbench for Amazon DynamoDB – GA.	NoSQL Workbench for Amazon DynamoDB is generally available.	March 2, 2020

Version	Change	Description	Date
0.4.1	Support for IAM roles and temporary security credentials.	NoSQL Workbench for Amazon DynamoDB adds support for AWS Identity and Access Management (IAM) roles and temporary security credentials.	December 19, 2019
0.3.1	Support for DynamoDB local (Downloadable Version).	The NoSQL Workbench now supports connecting to DynamoDB local (Downloadable Version) to design, create, query, and manage DynamoDB tables.	November 8, 2019
0.2.1	NoSQL Workbench preview released.	This is the initial release of NoSQL Workbench for DynamoDB. Use NoSQL Workbench to design, create, query, and manage DynamoDB tables.	September 16, 2019