dbdoc Manual

Version: 1.0

October 2025

Author: Subhasis Mukherjee

Gihub: https://github.com/subhasis68/dbdoc

dockerHub: https://hub.docker.com/r/subhnand/dbdoc-app

Table of Contents

About dbdoc

dbdoc is a browser based application that fetches metadata of a database and shows it on screen in a tree structure. When a node (database, tables, views, functions, procedures etc.) shown in the tree is right-clicked, a popup window opens up for the user to add / edit / view description of the node.

The tree structure can be stored, along with the user-feeded description for different database entities and relationships, in a database (Postgres) in jsonb format and retrieved at a later stage to show in the tree structure as usual.

Features

- Extract database metadata and display in a tree like structure
- The user can add description to entities and relationships
- Save database metadata along with description in a separate database in jsonb format
- Capable of generating and viewing graph database for the metadata in neo4j
- Capable of generating and viewing network diagram
- Capable of generating and viewing Entity-Relationship (ER) diagram
- Capable of generating metadata documentation in MS Word format

Limitations

- dbdoc supports databases for Postgres, MySQL, MongoDB, MS SQL Server, Oracle, IBM DB2

Installation

Step 1: Install Docker Desktop

Download and install docker desktop, if you do not have it already in your computer.

Step 2: Start Docker Desktop and the engine

Start the docker desktop which will automatically start the docker engine.

Step 3: Create a common docker network for use with all docker containers to use

Open a terminal or console and execute the following to create a docker network.

```
docker network create flink-net
```

Step 4: Create the docker compose file for postgres

Create a file called **02_postgres.yaml** and copy-paste the following and save the file.

```
services:
  postgres:
    image: postgres:17
    container_name: postgres_dbdoc
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=dbdocdb
    ports:
      - "5431:5432"
    networks:
      - flink-net
    volumes:
      - pgdata:/var/lib/postgresql/data

networks:
  flink-net:
    external: true

volumes:
  pgdata:
```

Step 5: Download and Start postgres in a docker container

Open a terminal or console and go to the folder containing the file 02_postgres.yaml and execute the following command:

```
docker compose -f 02_postgres.yaml up -d
```

Step 6: Create the postgres database and table to store database metadata

Open a postgres console (you may use tools like DBeaver)

I) Execute the following to create the database dbdocdb, if the dbdocdb does not exist already.

```
CREATE DATABASE dbdocdb WITH TEMPLATE = template0 ENCODING = 'UTF8'
LOCALE_PROVIDER = libc LOCALE = 'en_US.utf8';
```

II) With the database dbdocdb as the current database, execute the following in the order shown below. This will create the table metadata_store in the database dbdocdb with all requirements. The metadata_store stores the database metadata in jsonb format.

```
CREATE TABLE public.metadata_store (
    id bigint NOT NULL,
    connection_hash text NOT NULL,
    version integer NOT NULL,
    metadata_json jsonb NOT NULL,
    updated_at timestamp without time zone DEFAULT now(),
    updated_by text
);
```

```
ALTER TABLE public.metadata_store ALTER COLUMN id ADD GENERATED ALWAYS AS IDENTITY (
    SEQUENCE NAME public.metadata_store_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1
);

ALTER TABLE ONLY public.metadata_store
    ADD CONSTRAINT metadata_store_pkey PRIMARY KEY (id);

ALTER TABLE ONLY public.metadata_store
    ADD CONSTRAINT metadata_store_unique_version UNIQUE (connection_hash, version);
```

Step 7: Create the docker compose for neo4j

Create a file called **04_neo4j.yaml** and copy-paste the following and save the file.

```
services:
  neo4j:
    image: neo4j:5.26-enterprise
    container_name: neo4j
    restart: unless-stopped
    ports:
      - "7474:7474"   # HTTP (Neo4j Browser)
      - "7687:7687"   # Bolt protocol (drivers)
    environment:
      NEO4J_ACCEPT_LICENSE_AGREEMENT: "yes"
      NEO4J_AUTH: "none"

      # Enable APOC file imports
      NEO4J_apoc_import_file_enabled: "true"
      NEO4J_apoc_import_file_use__neo4j__config: "true"

      # Correct plugin setting
      NEO4J_PLUGINS: '["apoc", "graph-data-science"]'

    volumes:
      - neo4j_data:/data
      - neo4j_logs:/logs
      - neo4j_plugins:/plugins
      - ./import:/import   # Local folder for imports

    networks:
      - flink-net

volumes:
  neo4j_data:
  neo4j_logs:
  neo4j_plugins:

networks:
  flink-net:
    external: true
```

Step 8: Download and Start neo4j in a docker container

Open a terminal or console and go to the folder containing the file 04_neo4j.yaml and execute the following command:

```
docker compose -f 04_neo4j.yaml up -d
```

Step 9: Create the docker compose file for dbdoc-app

Create a file called **05_dbdoc-app.yaml** and copy-paste the following and save the file.

```
services:
  web:
    image: subhnand/dbdoc-app:v1.0
    container_name: dbdoc-web
    ports:
      - "3000:3000"
    networks:
      - flink-net
    restart: unless-stopped

networks:
  flink-net:
    external: true
```

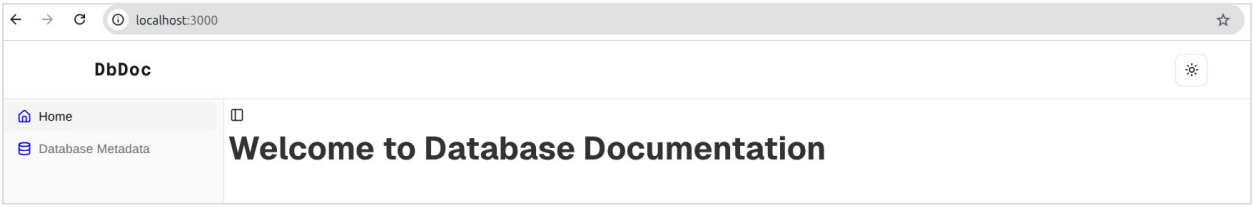
Step 10: Download and Start dbdoc-app in a docker container

Open a terminal or console and go to the folder containing the file 05_dbdoc-app.yaml and execute the following command:

```
docker compose -f 05_dbdoc-app.yaml up -d
```

Step 11: Test the installation

Open a browser (preferably Google Chrome) and open the link http://localhost:3000 You will see a page as shown below.



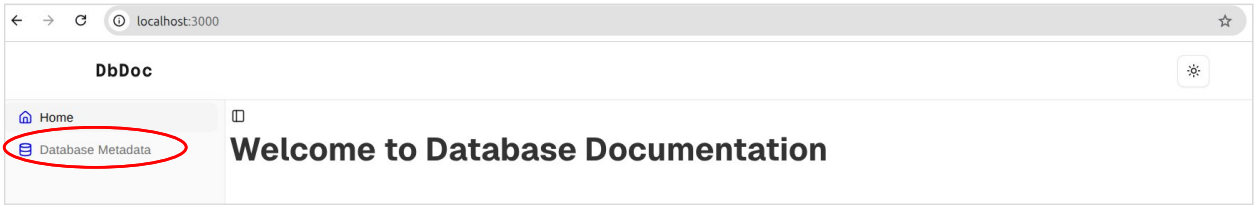
Go to top

Usages

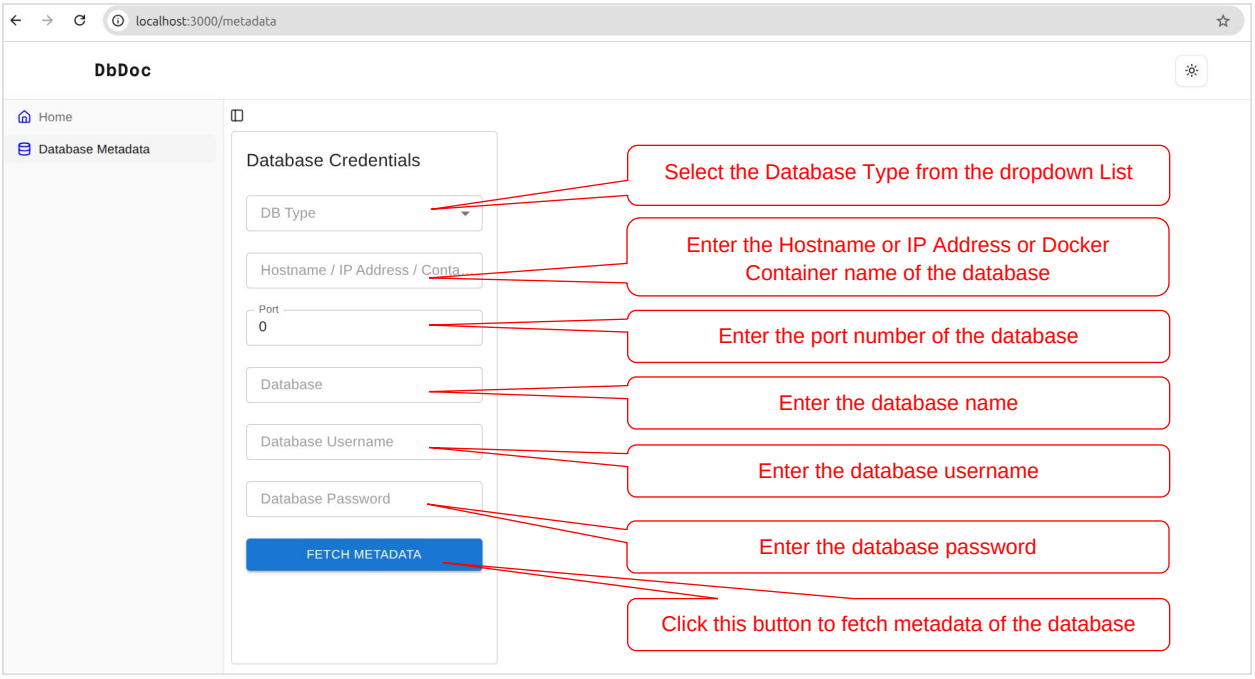
Open a browser (preferably Google Chrome) and enter the url **http://localhost:3000**.

Usage 1: Fetch database metadata and view in a tree structure

Click on the Database Metadata option on the left pane of the screen as highlighted below.



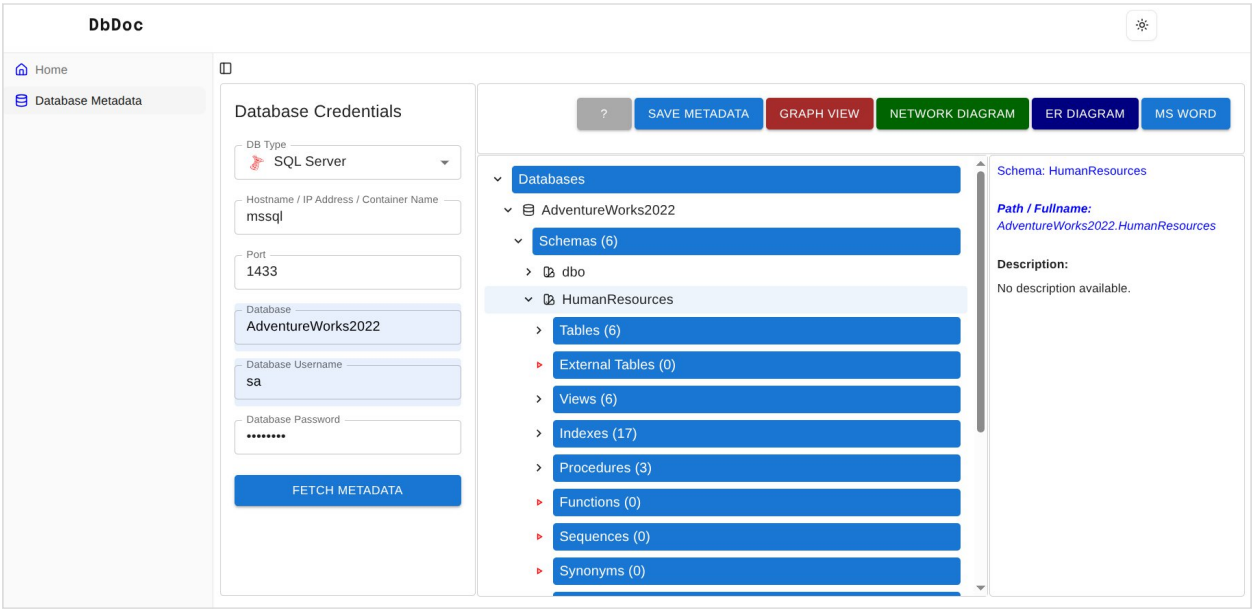
The following screen will appear. Follow the instructions as shown below.



Note: As the dbdoc app runs as a docker image. The hostname can never be localhost and the IP Address can never be 127.0.0.1. If the database is running in the same machine that runs the dbdoc app, the hostname can be used as **host.docker.internal** . If the database is running in another docker container, please provide the docker container name of the database. In that case the dbdoc app docker container and the database docker container should be on the same docker network.

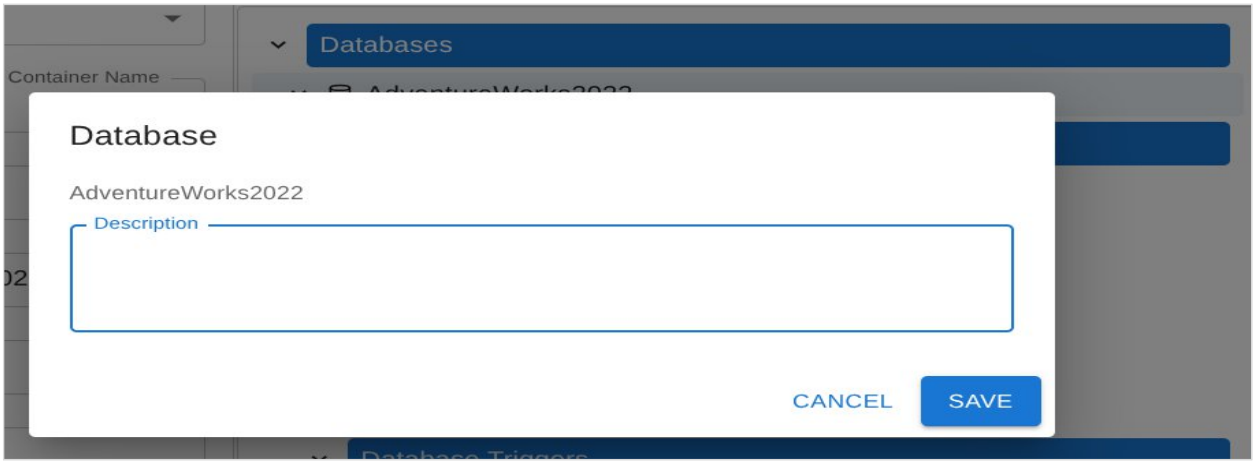
An example follows as below.

Example: Consider a MS SQL Server database running in a **docker container** name **mssql**. The following image shows the database credentials entered and the result shown on the right pane.



Right click on any metadata object like database, table, column, view, function, procedure etc. and a popup screen will appear to type in the description of the object, as shown below.

Left Click on any metadata object to see the description on the right pane.



Type in the description and click the save button to save the description.

Note: *Saving the description does not save the metadata in the background postgres database (dbdocdb). To save the metadata along with the description click on the "SAVE METADATA" button. This will save the metadata in jsonb format in the dbdocdb database in the metadata_store table and will also create a neo4j graph database for the metadata.*

Go to top

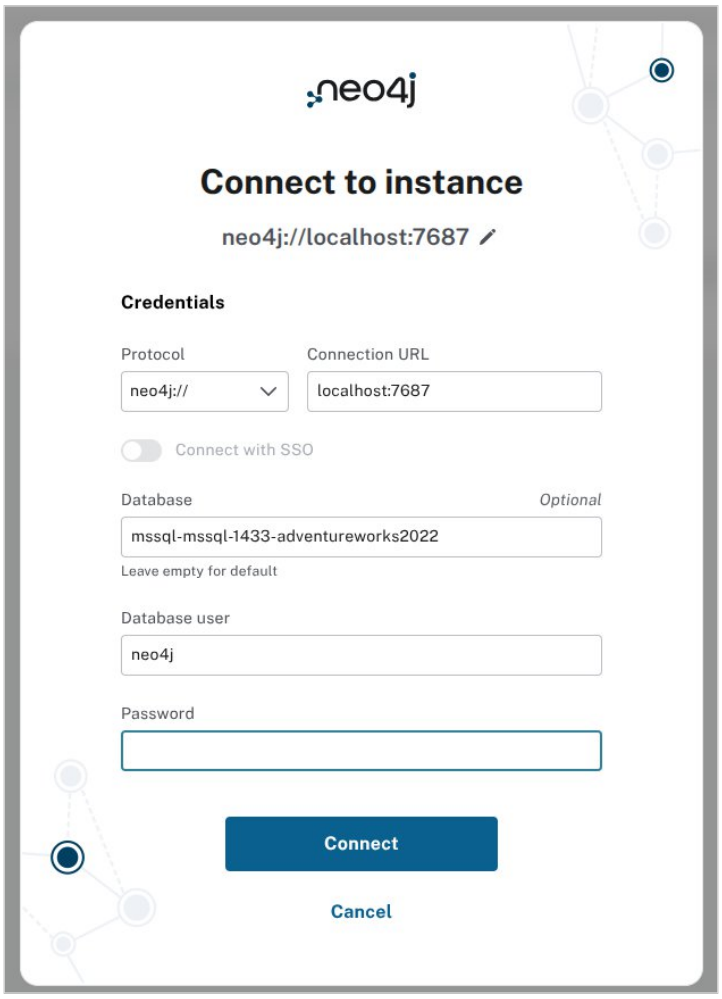
Usage 2: Graph View of the database metadata in neo4j

Pre-requisites:

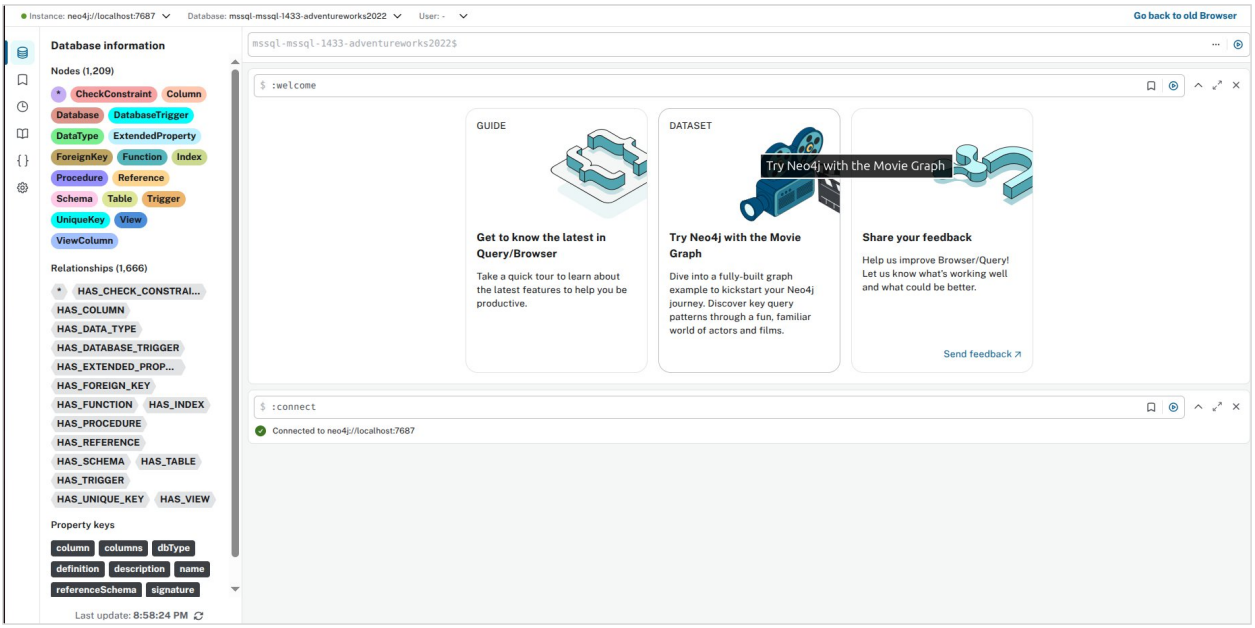
1. The metadata must be saved using the **SAVE METADATA** button.

2. The neo4j database engine must be up and running. Please check the docker desktop and ensure that the neo4j database engine is up and running.

Click on the **GRAPH VIEW** button. The following screen should appear in a separate browser tab.



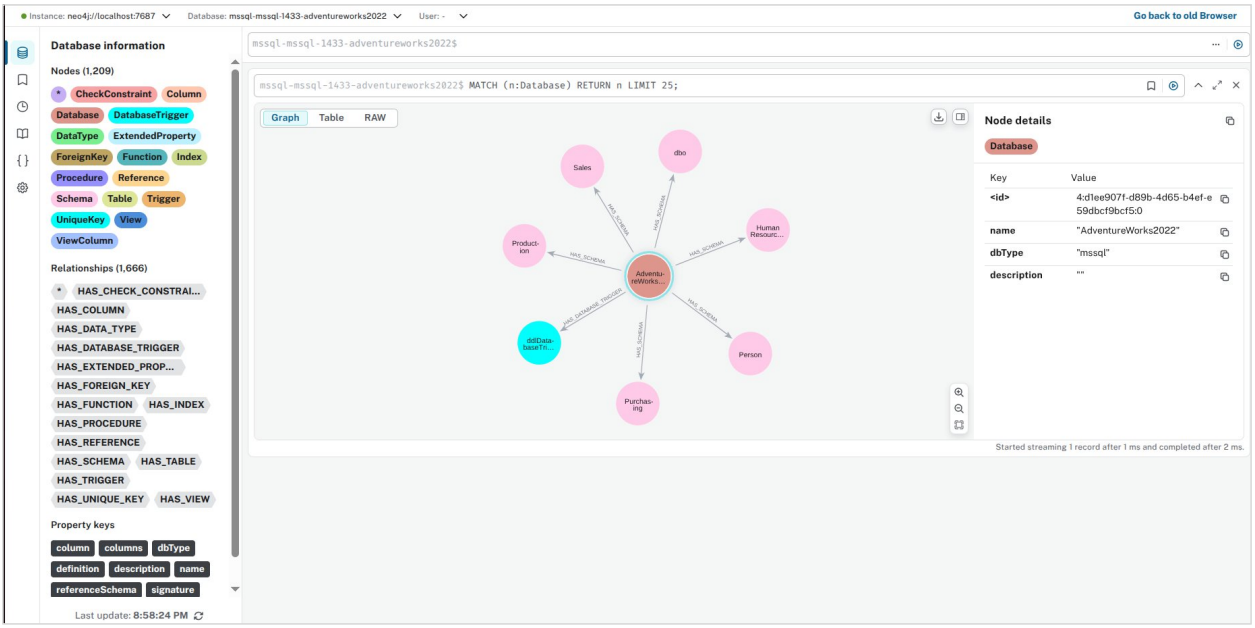
Click on the **Connect** button. The following neo4j console will appear.



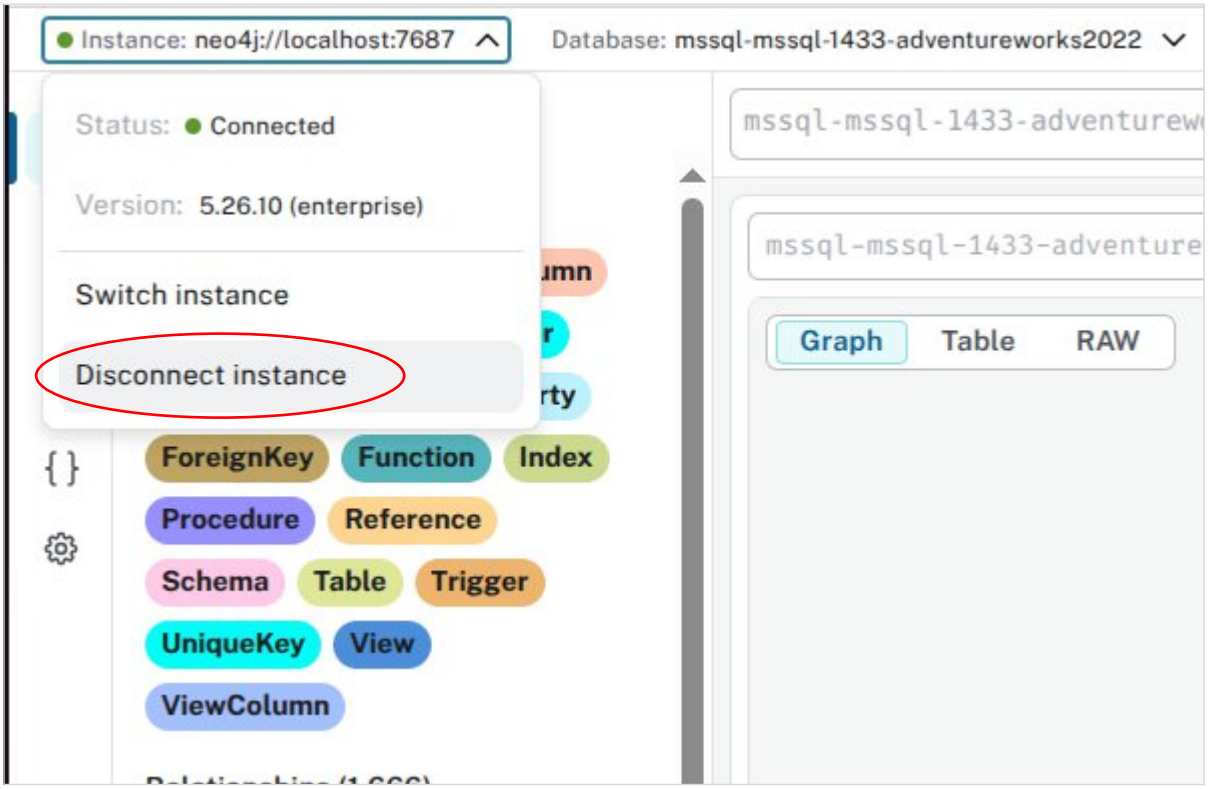
You may close the welcome frame.

Click on the Database or other nodes on the left to view the graph. The rest of the operations to view the different metadata aspects are as done in neo4j.

An example view of the Database graph is as shown below.



Note: *After you are done with the neo4j instance, please disconnect the instance as shown below. Otherwise, when you try to view graph from dbdoc at a later stage, the correct neo4j database will not get opened.*



Go to top

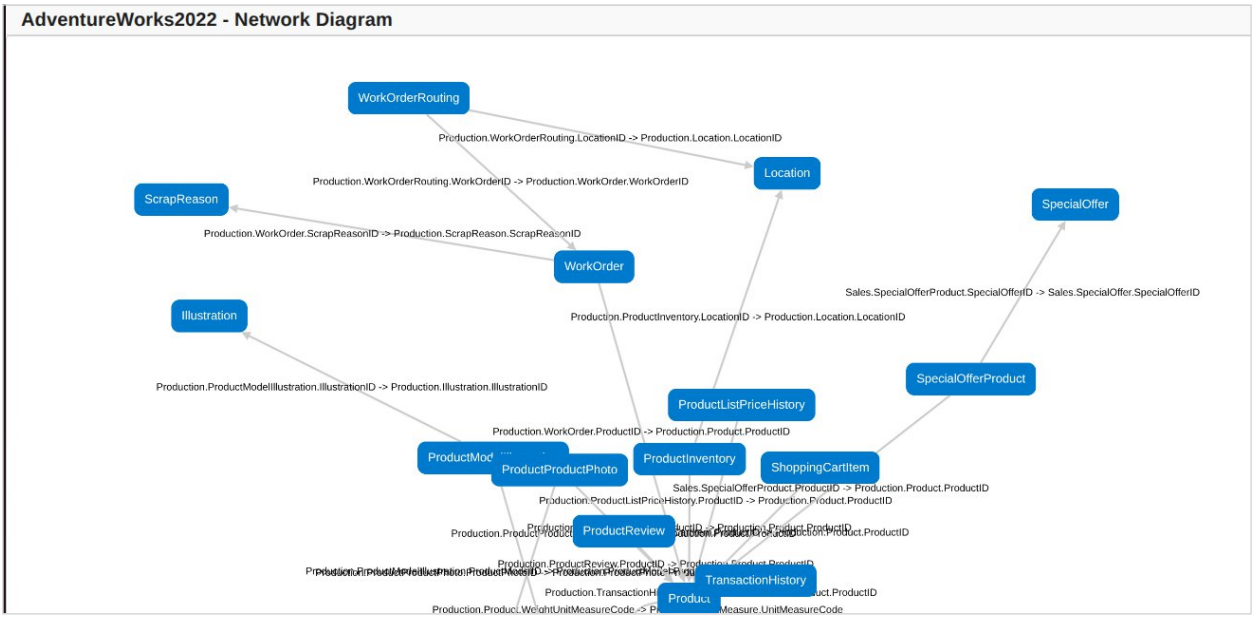
Usage 3: Network Diagram of the database metadata

Click on the **NETWORK DIAGRAM** button in dbdoc.

The network diagram of the database metadata will be opened in a separate browser tab.

You can click and drag to reposition the nodes of the network diagram. Also you can zoom in or zoom out the diagram by using the mouse wheel for better or closer view.

A sample network diagram is shown below.



Go to top

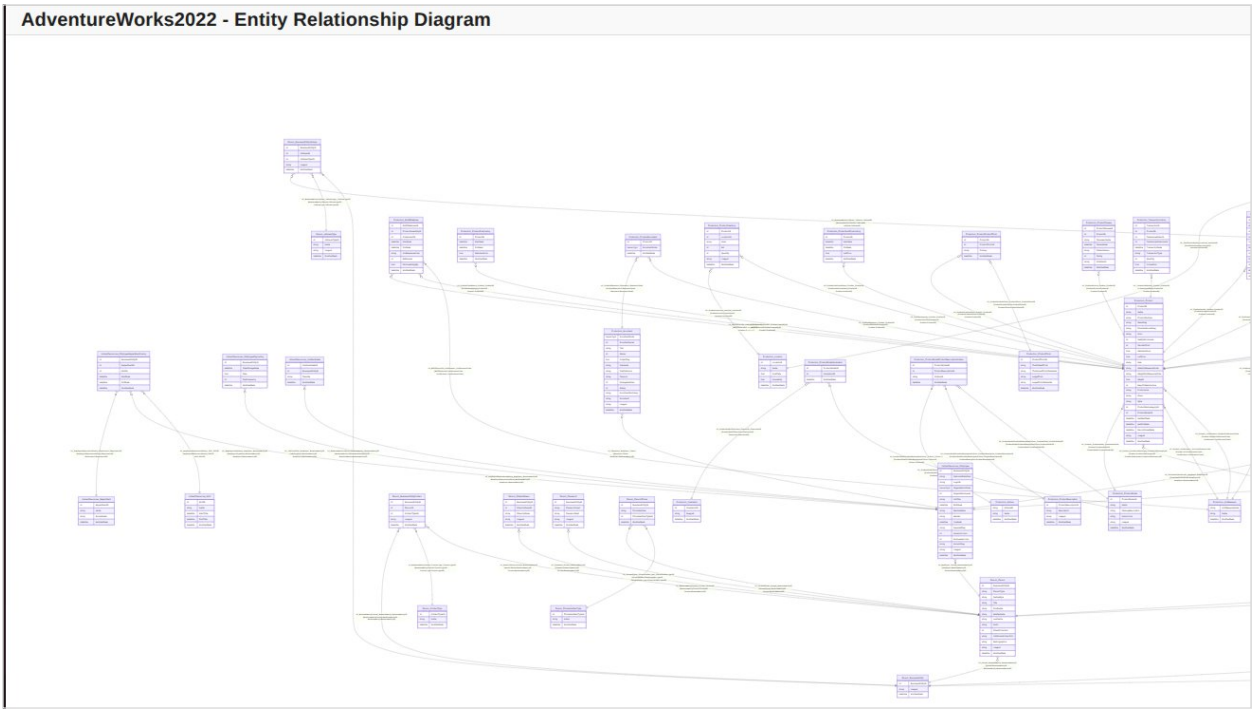
Usage 4: Entity-Relationship (ER) Diagram of the database metadata

Click on the **ER DIAGRAM** button in dbdoc.

The ER diagram of the database metadata will be opened in a separate browser tab.

You can click and drag to reposition the ER diagram. Also you can zoom in or zoom out the diagram by using the mouse wheel for better or closer view of any part of the diagram.

A sample ER diagram is shown below.



Usage 5: Download the database metadata in MS Word document

Click on the **MS WORD** button in dbdoc.

The metadata of the database will be downloaded in the MS Word format.

Please open the document in Microsoft Word or ONLYOFFICE and update the Table of Contents for correct result.

Go to top