Contents

1 Introduction

DBMS stands for *database management system*. It may be a simple function library or even a larger system composed by several programs and processes running separately and in parallel, whose main function is to manage one or several databases hosted in a server. It has the responsibility to manipulate and to keep the consistency of data, allowing the software developers to focus on functionalities. Thus, practically any modern system that manages data utilizes some kind of DBMS, regardless of the amount of stored information.

OmniDB's creators, Rafael Thofehrn Castro and William Ivanski, worked in a company where they needed to deal with several different databases from customers on a daily basis. These databases were from different DBMS technologies, and so they needed to keep switching between database management tools (typically one for each DBMS). As they were not keen of the existing unified database management tools (that could manage different DBMS), they came up with OmniDB's main idea.

OmniDB's first version was presented as an undergrad final project in the Computer Science Course from the Federal University of Paraná, in Brazil. The objective was to trace a common line between popular DBMS, and to study deeply their *metadata*. The result was a tool written in ASP.NET/C# capable of connecting and identifying the main structures (tables, keys, indexes and constraints), in a generic way, from several DBMS:

- Firebird
- MariaDB / MySQL
- Oracle
- PostgreSQL
- SQLite
- Microsoft SQL Server

OmniDB's first version also allowed the conversion between all DBMSs supported by the tool. This feature was developed to be user friendly, requiring just a few steps: the user needs to select a source connection, the structures that will be converted (just tables and all their structures, along with their data) and the target connection.

Since early development, OmniDB was designed as an web app. Consequently, it runs in any browser, from any operational system. It can be accessed by several computers and multiple users, each one of them with his/her own group of connections. It also can be hosted in any operational system, without the need of install any dependencies. We will see further details on installation in the next chapters.

OmniDB's main objective is to offer an unified workspace with all functionalities needed to manipulate different DMBS. DBMS specific tools aren't required: in OmniDB, the context switch between different DBMS is done with a simple connection switch, without leaving the same page. The end-user's sensation is that there is no difference when he/she manipulates different DBMS, it just feels like different connections.

Despite this, OmniDB is built with simplicity in mind, designed to be a fast and lightweight web application. OmniDB is also powered by the WebSocket technology, allowing the user to execute multiple queries and procedures in multiple databases in multiple hosts in background.

OmniDB is also secure. All OmniDB user data are stored encrypted, and no database password is stored at all. When the user first connects to a database, OmniDB asks for the password. This password is encrypted and stored in memory for a specific amount of time. When this time expires, OmniDB asks the password again. This ensures maximum security for the database OmniDB is connecting to.

2 Migration from .NET / Mono to Python

OmniDB was rewritten to Python using the Django framework. Starting from version `2.0`, OmniDB Python version will receive new features and will be actively maintained.

The source code for the ASP.NET/C# version is in the branch `csharp`. The next release of OmniDB C# version is 1.7, and it will only receive bug fixes.

OmniDB source code is hosted on GitHub and there are 3 main branches:

- **master**: Contains the current beta release of OmniDB Python version
- **dev**: Contains the current development release of OmniDB Python version
- **csharp**: Contains the .NET / Mono version of OmniDB

Besides being written in Python, initial version of OmniDB 2.0 contains the following main differences from the C# version:

- Support to HTTPS;
- It allows query execution in background and cancellation through the use of websockets;
- There is a new Snippet feature;
- Log capabilities and a test suite are almost finished;
- Initially, only an improved support of PostgreSQL is implemented. More RDBMS support coming soon;
- Database conversion feature was disabled by now, but it will be re-enabled soon;
- You don't need to install dependencies and web servers any more. Everything OmniDB needs is now bundled together.

3 Installation

OmniDB provides 2 kinds of packages to fit every user needs:

- **OmniDB Application**: Runs a web server on a random port behind, and provides a simplified web server window to use OmniDB interface without any additional setup. Just feels like a desktop application.
- **OmniDB Server**: Runs a web server on a random port. User needs to connect to it through a web browser. Provides user management, ideal to be hosted on a server on users' networks.

Both application and server can be installed on the same machine.

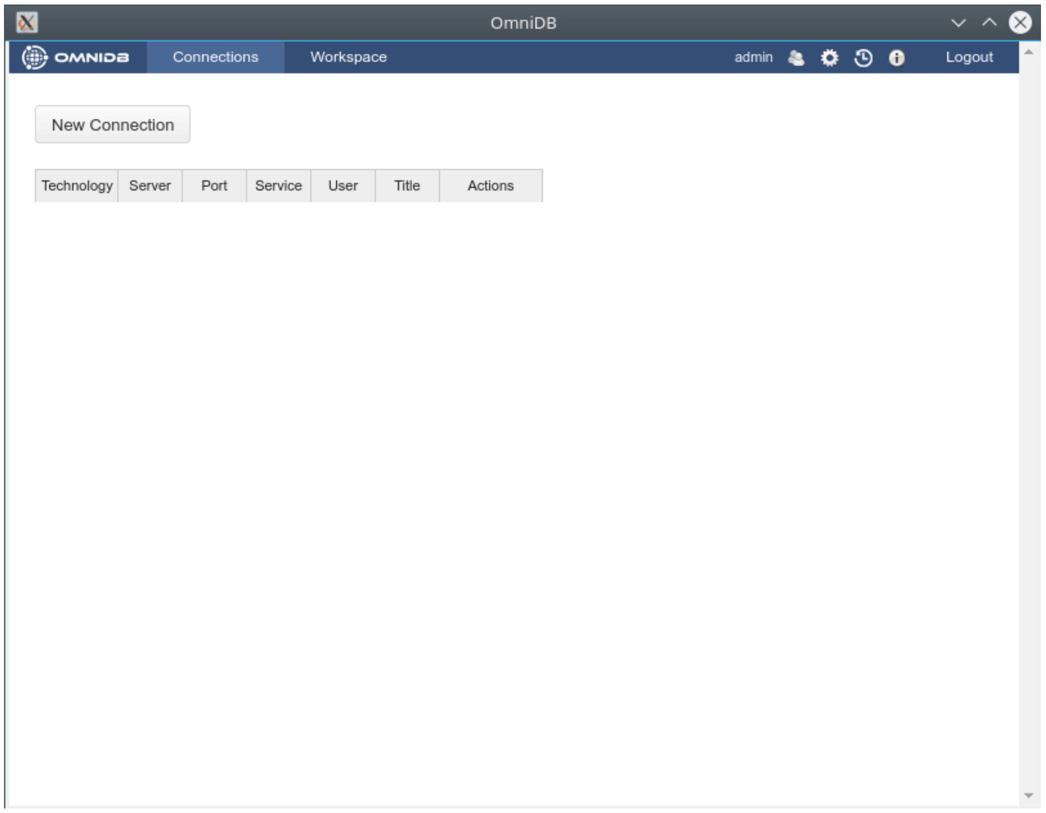
3.1 OmniDB Application

In order to run OmniDB app, you don't need to install any additional piece of software. Just head to omnidb.org and download the latest package for your specific operating system and architecture:

- Linux 32 bits / 64 bits
  - DEB installer
  - RPM installer
  - Tarball
- Windows 32 bits / 64 bits
  - EXE installer
  - ZIP package
- Mac OSX
  - DMG installer
  - ZIP package

If you choose tarball or zip packages, just extract it somewhere in your computer. Get inside the folder it creates and run the `omnidb-app` executable. It will open OmniDB inside its own window.

With the installer you can install OmniDB in your system, and it will be available through your desktop environment application menu. When you launch it, OmniDB will open its own window.



Technology	Server	Port	Service	User	Title	Actions

3.2 OmniDB Server

Like OmniDB app, OmniDB server doesn't require any additional piece of software and the same options for operating system and architecture are provided.

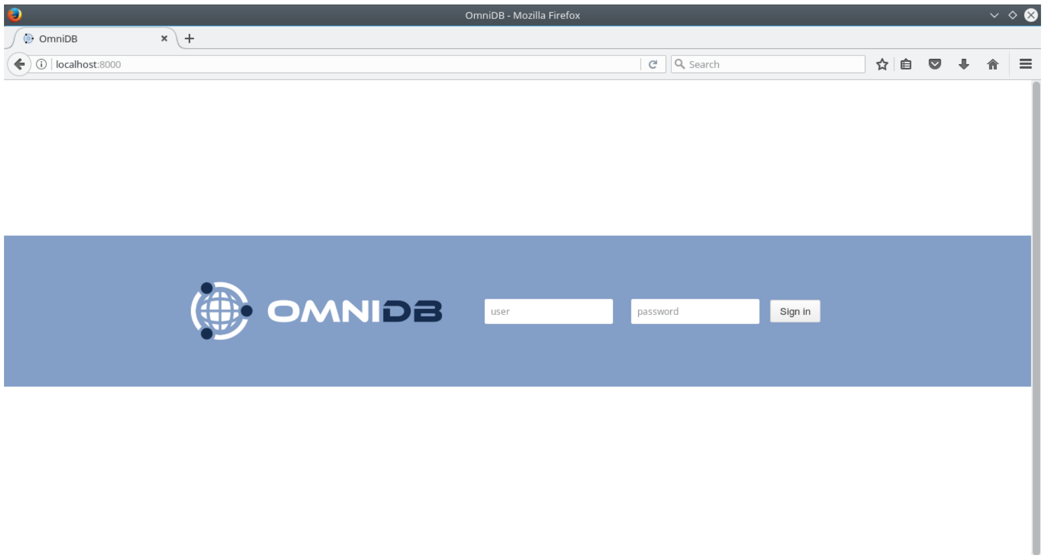
If you choose the tarball or zip package, extract it somewhere in your computer. Get inside the uncompressed folder and run the `omnidb-server` executable.

```
user@machine:~$ cd omnidb-server
user@machine:~/omnidb-server$ ./omnidb
Starting OmniDB 2.0.2 at http://localhost:8000
Open OmniDB in your favorite browser
Press Ctrl+C to exit
```

If you installed OmniDB server with some installer option, you will need administrator privileges:

```
user@machine:~$ sudo omnidb-server
Starting OmniDB 2.0.2 at http://localhost:8000
Open OmniDB in your favorite browser
Press Ctrl+C to exit
```

Now that the web server is running, you may access OmniDB web app on your favorite browser. Type in address bar: `localhost:8000` and hit `Enter`. If everything went fine, you shall see a page like this:

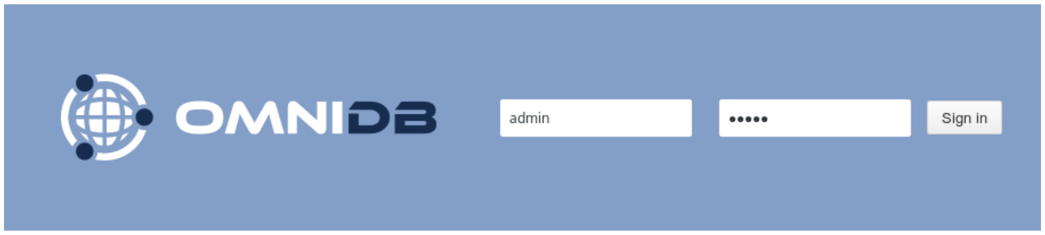


Now you know that OmniDB is running correctly. In the next chapters, we will see how to login for the first time, how to create an user and to utilize OmniDB.

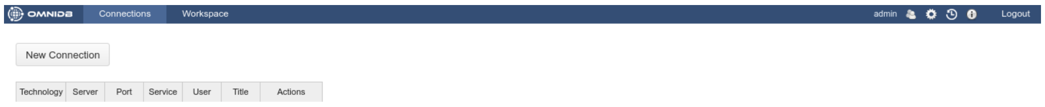
4 Creating Users and Connections

4.1 Logging in as user *admin*

OmniDB comes only with the user *admin*. If you are using the server version, the first thing to do is sign in as *admin*, the default password is *admin*. You don't need to login in the app version.



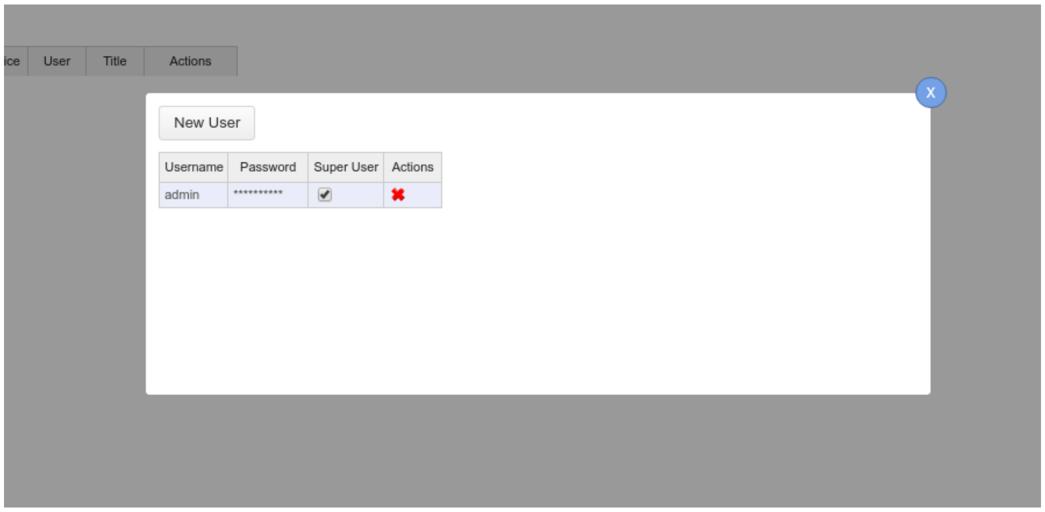
The next window is the **Connections** window. We will talk about it later.



Technology	Server	Port	Service	User	Title	Actions

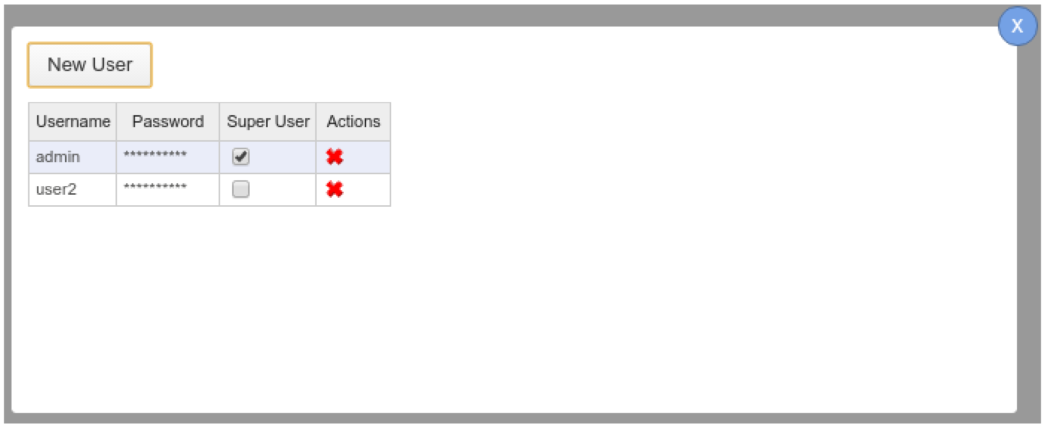
4.2 Creating another user

Click on the *Users* icon on the upper right corner. It will open a popup that allows the current OmniDB super user to create a new OmniDB user.



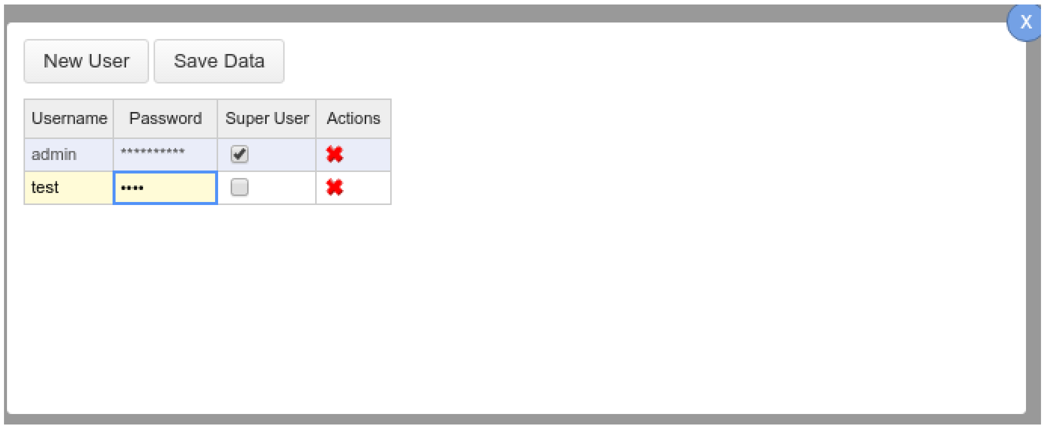
Username	Password	Super User	Actions
admin	**********	✓	✖

After clicking on the *Users* icon the tool inserts a new user called *user2* (if that is the first user after *admin*).



Username	Password	Super User	Actions
admin	**********	✓	✖
user2	**********	☐	✖

You will have to change the *username* and *password*. Check if you want this new user to be a *super user*. This user management window is only seem by super users. When you are done, click on the *Save Data* button inside the popup.

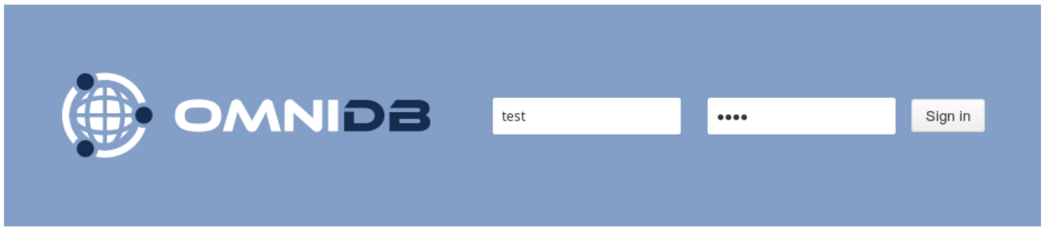


Username	Password	Super User	Actions
admin	**********	✓	✖
test	••••	☐	✖

You can create as many users as you want, edit existing users and also delete users by clicking on the red cross at the actions column. Now you can logout.

4.3 Signing in as the new user

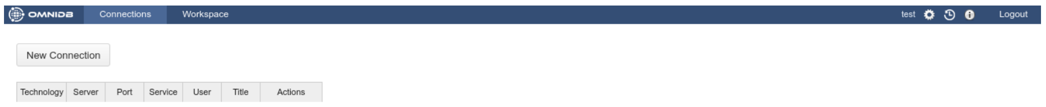
Let us sign in as the user we just created.



test

••••

And we can see the **Connections** window again. Note that now there is no *Users* icon, because the *test* user is not a super user.

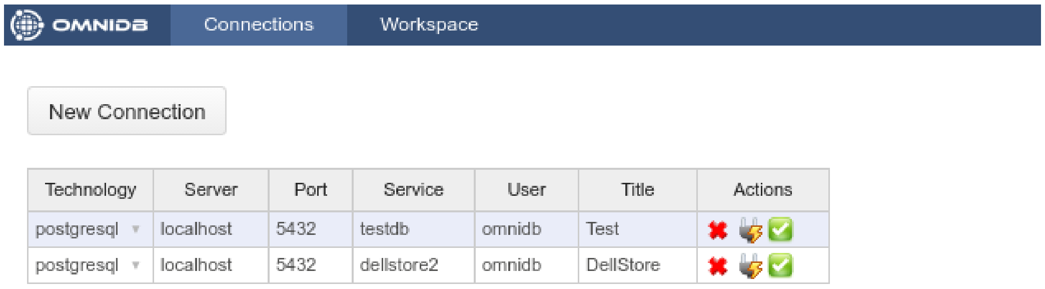


Technology	Server	Port	Service	User	Title	Actions

4.4 Creating connections

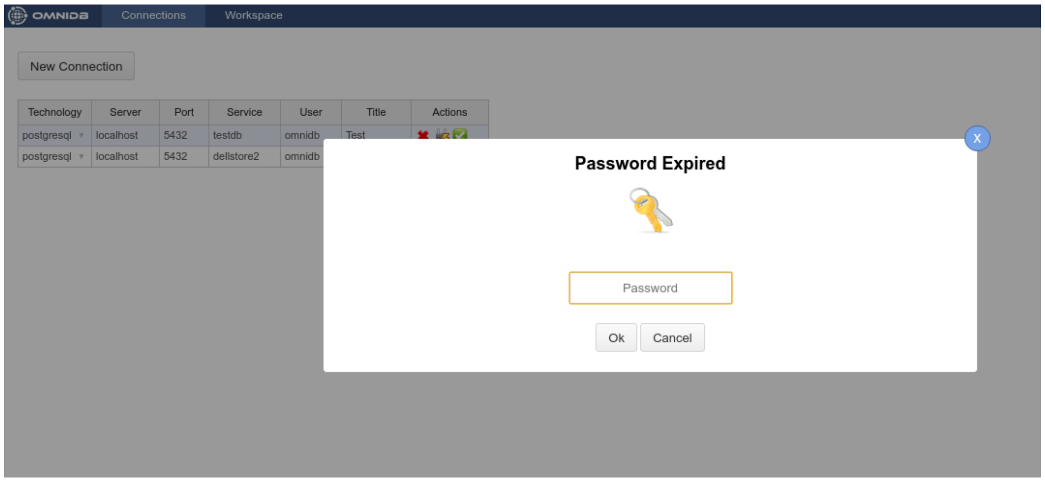
OmniDB C# version supported several DBMS. At the moment, OmniDB Python version, or `OmniDB 2.0`, supports only PostgreSQL. More DBMS support is being added as you read this.

We will now create two connections to PostgreSQL databases. To create the connections you have to click on the button *New Connection* and then choose the connection and fill the other fields. After filling all the fields for both connections, click on the *Save Data* button.



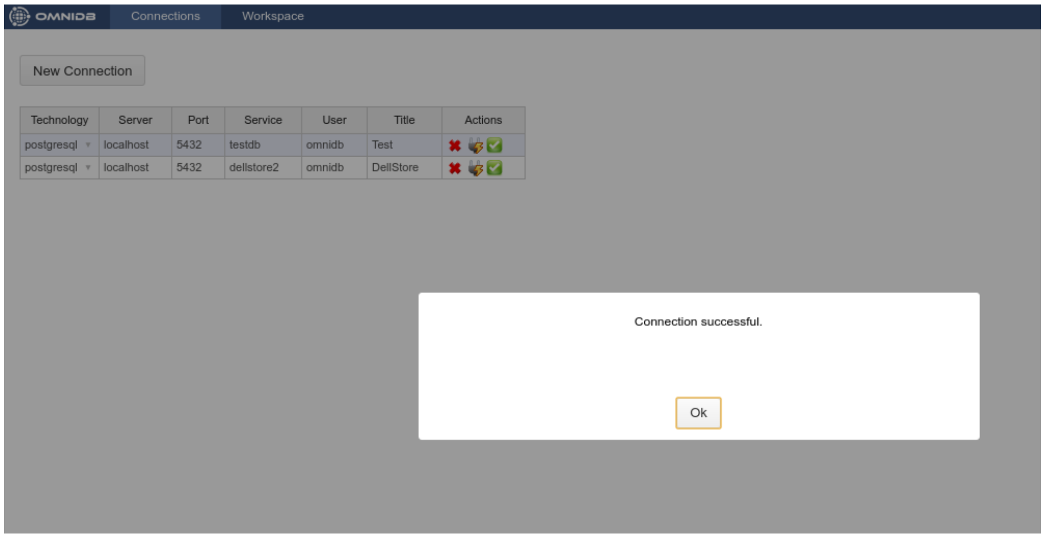
Technology	Server	Port	Service	User	Title	Actions
postgresql	localhost	5432	testdb	omnidb	Test	✖ ✓
postgresql	localhost	5432	dellstore2	omnidb	DellStore	✖ ✓

For each connection there is an *Actions* column where you can delete, test and select them. Go ahead and test one of the connections.



Notice the *Password Expired* pop-up. This is happening because OmniDB does not store the database user password on disk. When the user types a password in this popup, the password is encrypted and stored in memory.

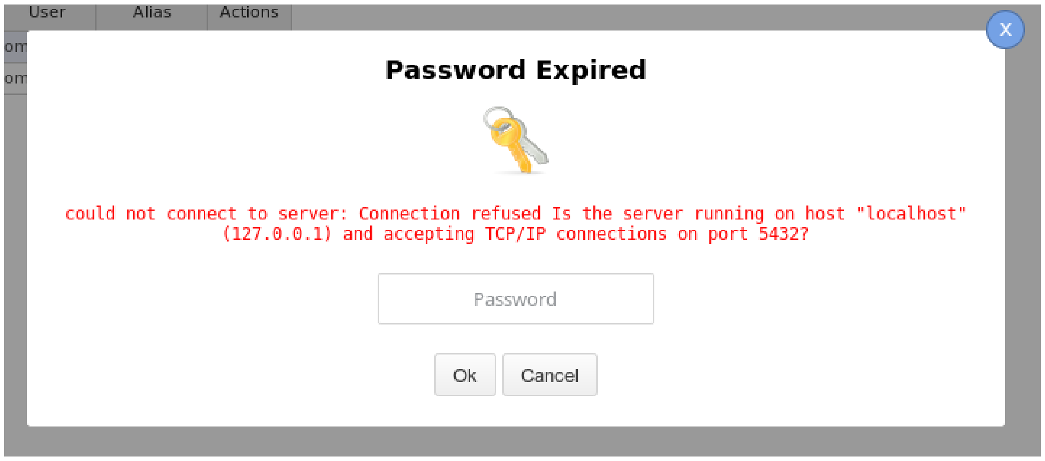
After you type the password and hit *Enter*, if the connection to the database is successful you will see a confirmation pop-up.



Technology	Server	Port	Service	User	Title	Actions
postgresql	localhost	5432	testdb	omnidb	Test	
postgresql	localhost	5432	dellstore2	omnidb	DellStore	

Connection successful.

But, if you have trouble of any kind connecting to your PostgreSQL database, the *Password Expired* popup will remain showing the error OmniDB got.

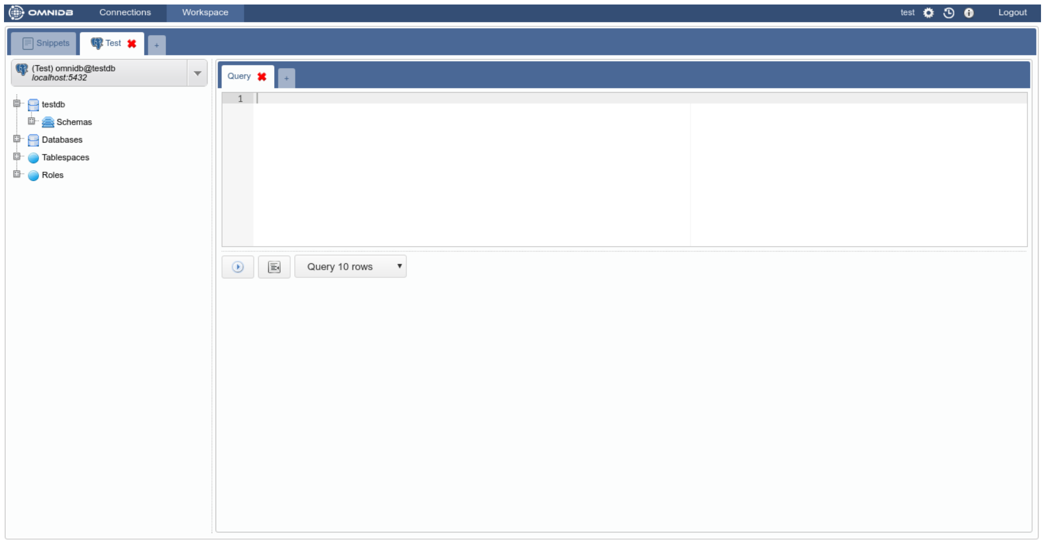


```
could not connect to server: Connection refused Is the server running on host "localhost"
(127.0.0.1) and accepting TCP/IP connections on port 5432?
```

Also, in the connections grid, if you click on the *Select Connection* action, OmniDB will open it in the **Workspace Window**.

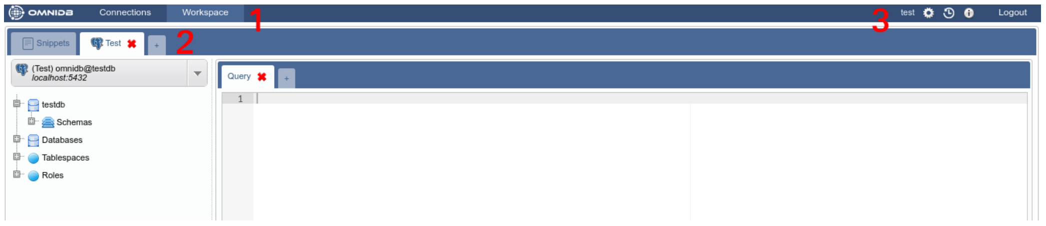
5 Workspace

After creating at least one connection the user can enter the *Workspace*, either by clicking the *Workspace* tab or by clicking in the *Select Connection* action in the connections grid.



5.1 Sections of the *Workspace* window

This interface has several elements:

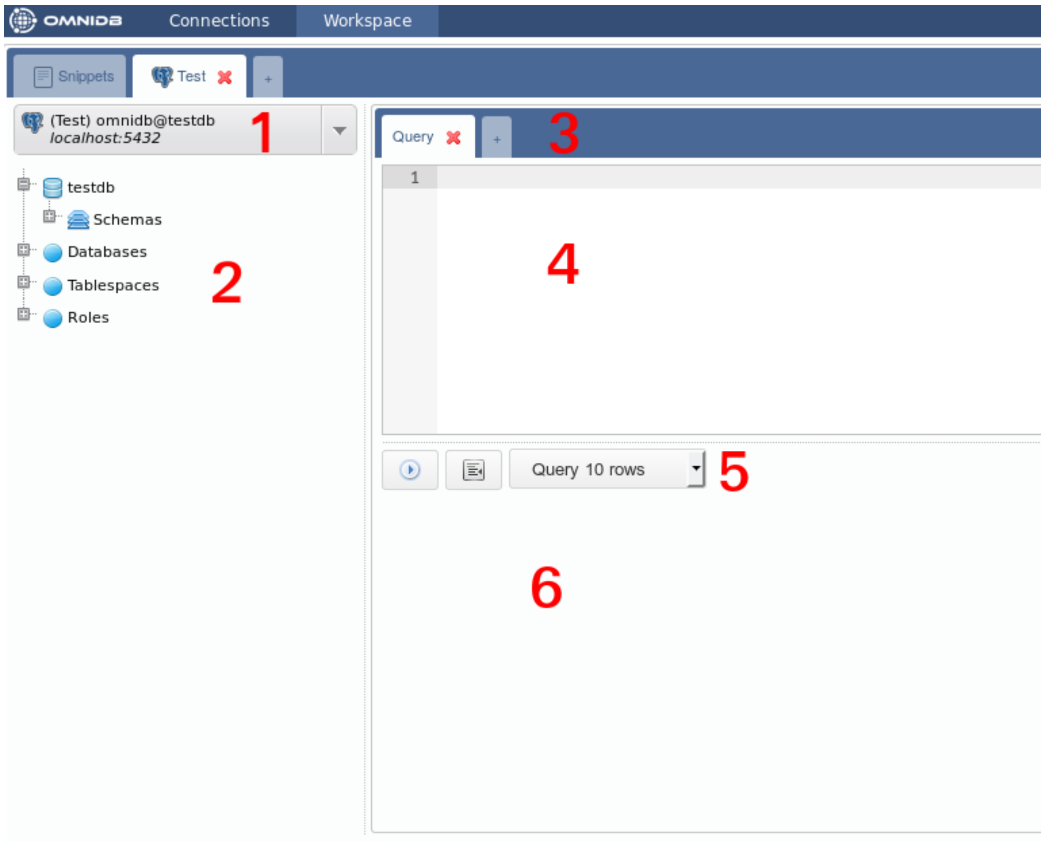


- **1) Links**: Enables the user to navigate between OmniDB windows
- **2) Outer Tabs**: OmniDB lets you work with several databases at the same time. Each database will be accessible through an *outer tab*. Outer tabs also can host miscellaneous features, like the *Snippets* feature
- **3) Options**: Shows the current user logged in, and also links for *user settings*, *query history*, *information* and *logout*.

5.2 Connection Outer Tab

So, the outer table named *Test* has this name because of the alias we put in the connection to the `testdb`. This tab is a *Connection Outer Tab*. Notice the little tab with a cross besides the *Test* outer tab. This allows you to create a new outer tab that will automatically be a *Connection Outer Tab*. However, the *Snippet Outer Tab* is fixed and will always be the first.

A new *Connection Outer Tab* will always automatically point to the first connection on your list of database connections. Or, if you clicked on the *Select Connection* action, it will point to the selected connection. Observe the elements inside of this tab:



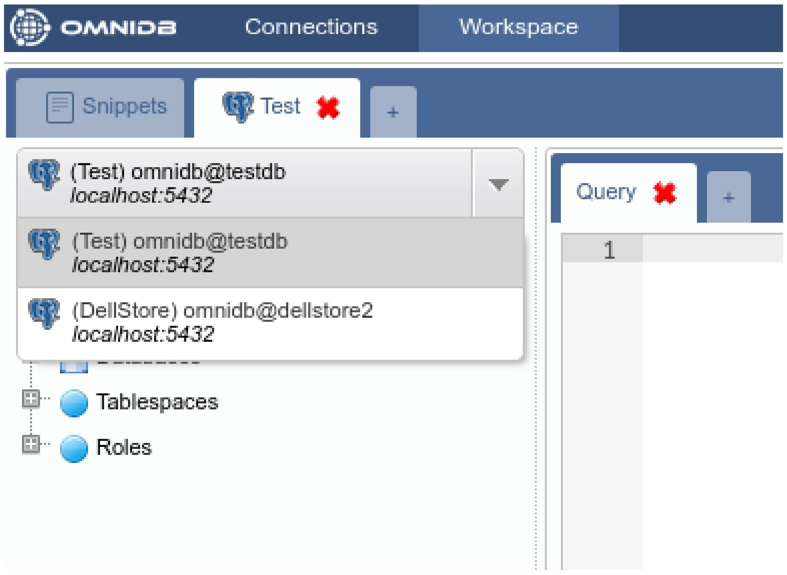
- **1) Connection Selector**: Shows all connections and lets the user select the current one
- **2) Tree of Structures**: Displays a hierarchical tree where you can

navigate through the database elements

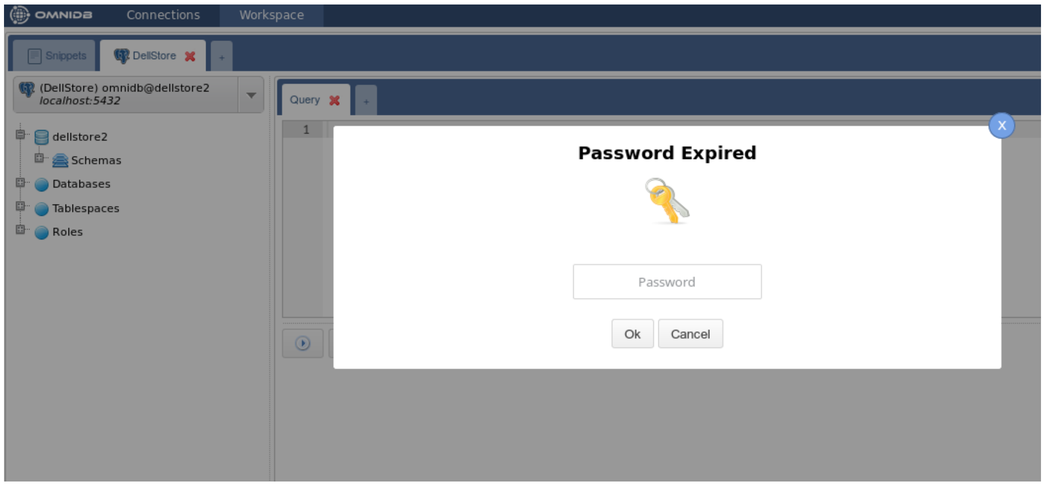
- **3) Inner Tabs**: Allows the user to execute actions in the current database. There are several kinds of inner tabs for the current database. By clicking on the last small tab with a cross, you can add a new tab. A new tab always will be a *Query Tab*, where you can write any kind of SQL statement
- **4) Inner Tab Content**: Can vary depending on the kind of inner tab. The figure shows a *Query Tab* and in this case the content will be an *SQL Editor*, with syntax highlight and autocomplete
- **5) Inner Tab Actions**: Can vary depending on the kind of inner tab. For a *Query Tab*, they are *Execute Button*, *Format Button* and *Editor Mode* (script, execute or query)
- **6) Inner Tab Results**: A *Query Tab* in query mode, after you click in the *Execute Button* or type the execute shortcut (`Alt-Q`), will show a grid with the query results. All modes will show error messages, if any.

5.3 Working with databases

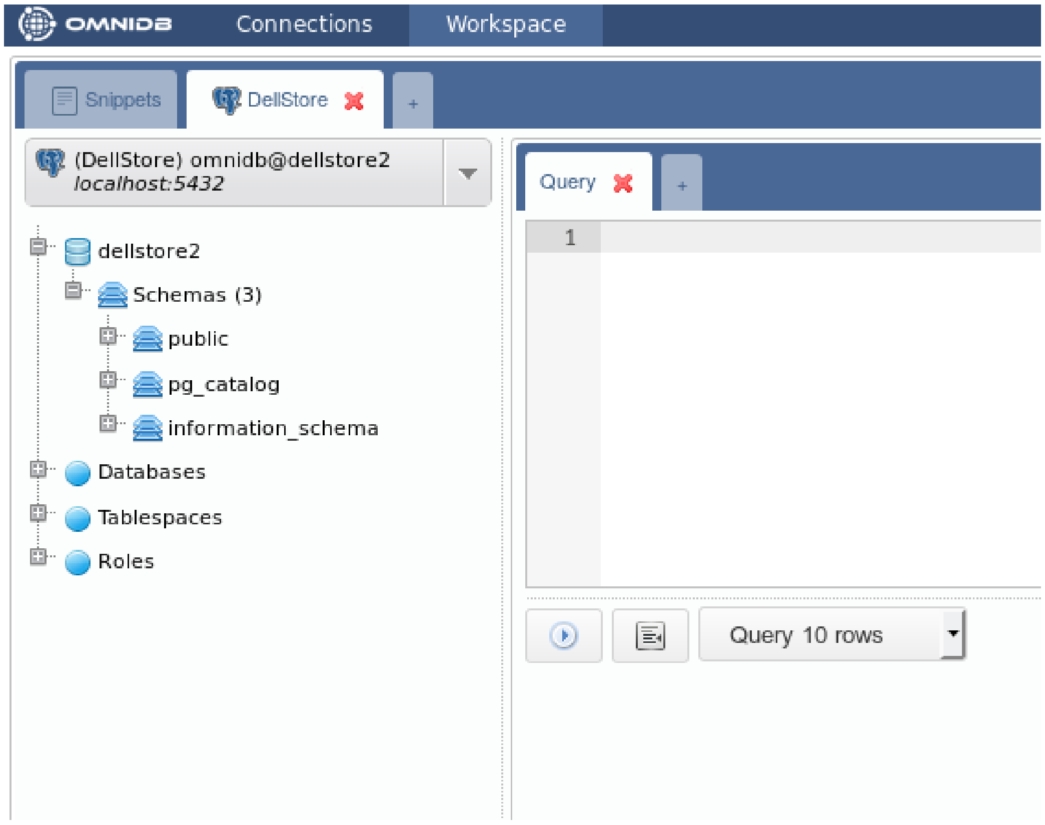
Take a look at your connections selector. OmniDB always points to the first available connection but you can change it by clicking on the selector.



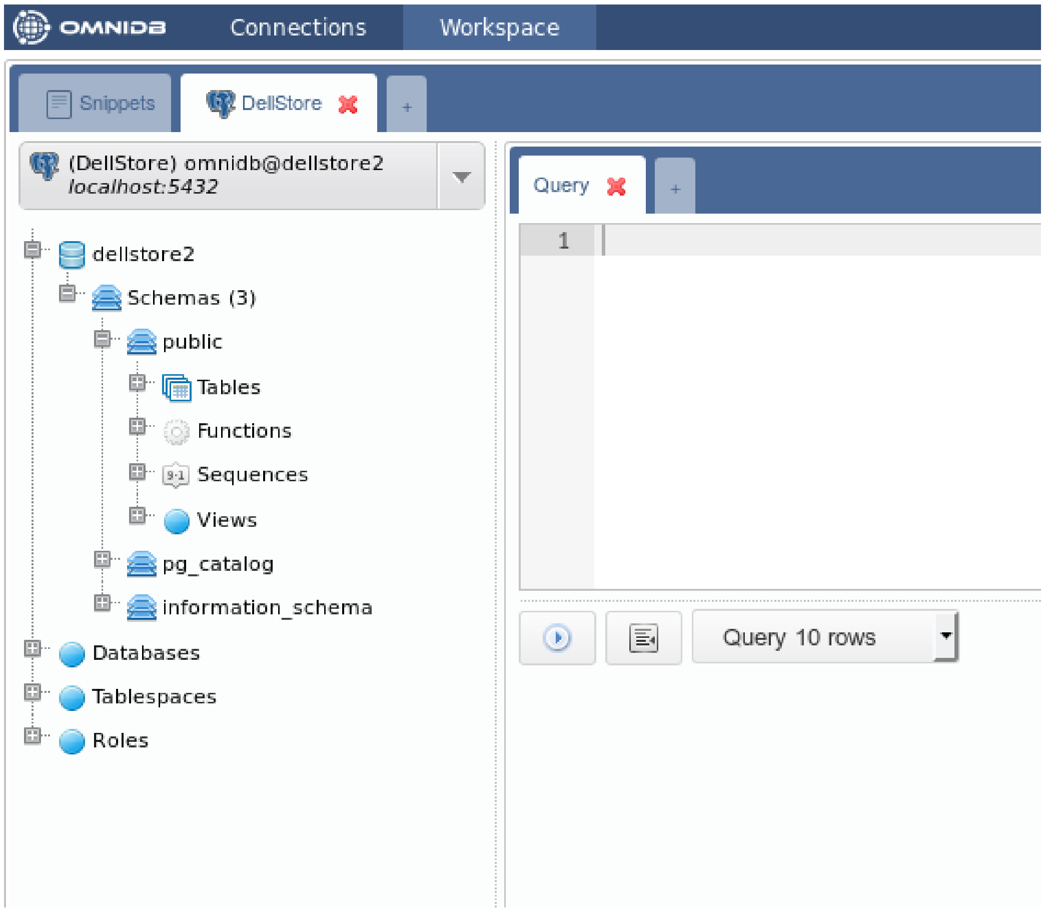
Select the *DellStore* connection. Now go to the tree right below the selector and click to expand the node *Schemas*.



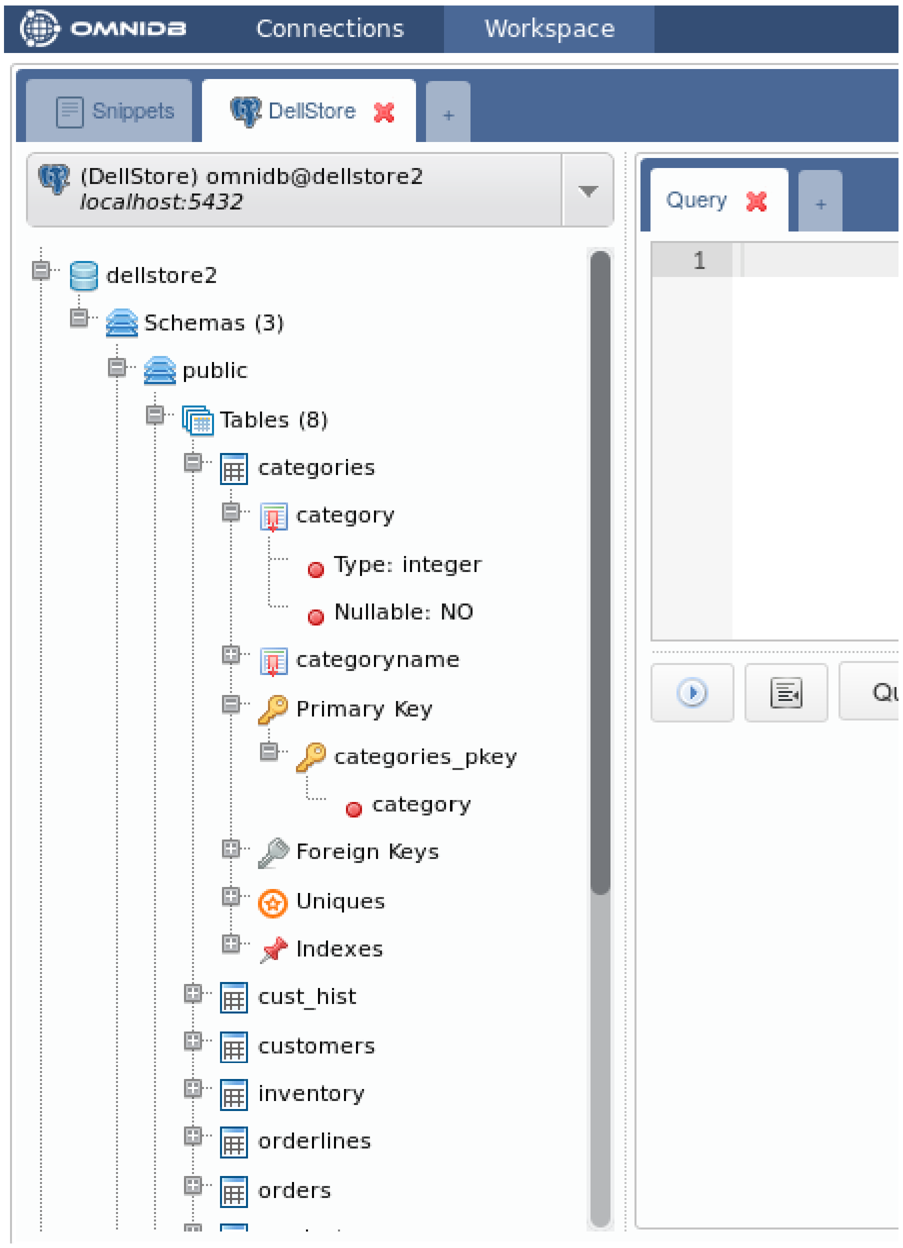
Bear in mind that every 10 minutes you keep without performing actions on the database, will trigger a *Password Expired* popup. As explained before, this is important for your database security. After you type the correct password, you will see all schemas in your database (in case of PostgreSQL, `TOAST` and temp schemas are not shown).



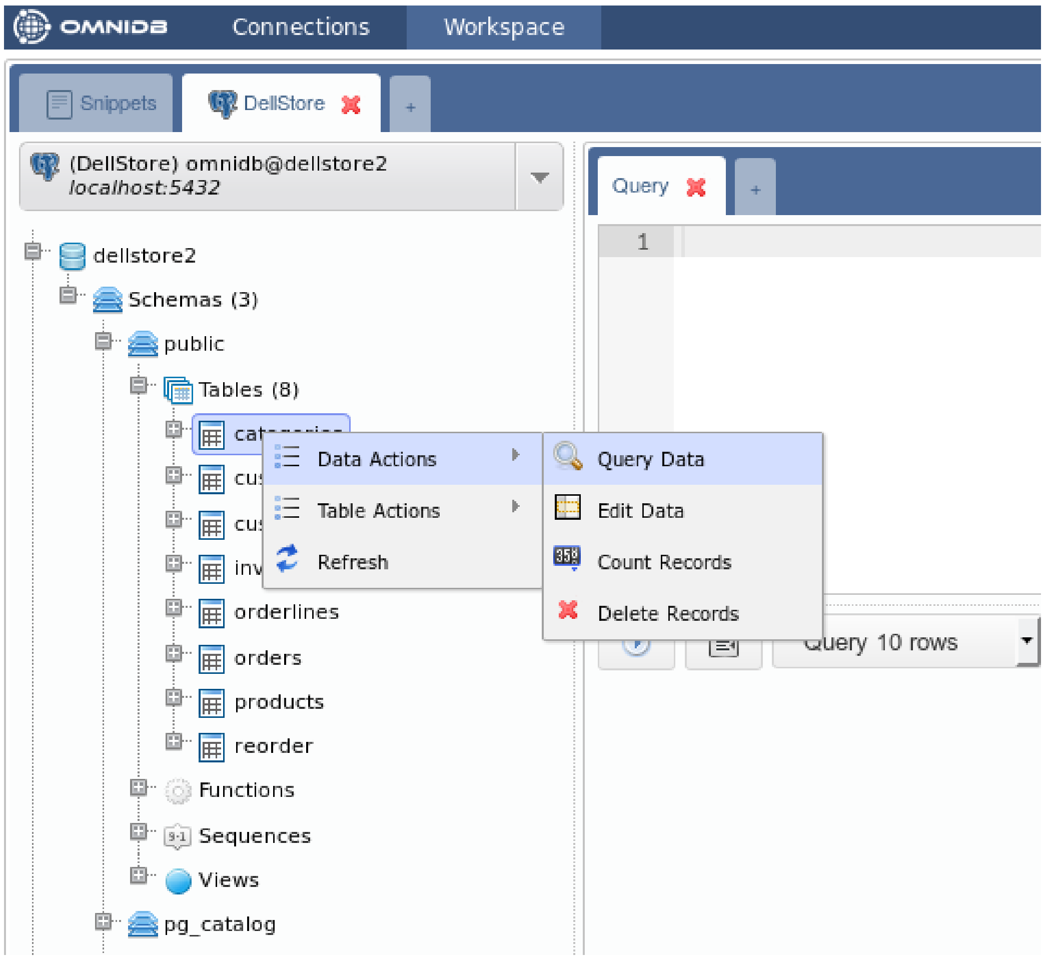
Now click to expand the schema `public`. You will see different kinds of elements contained in this schema.



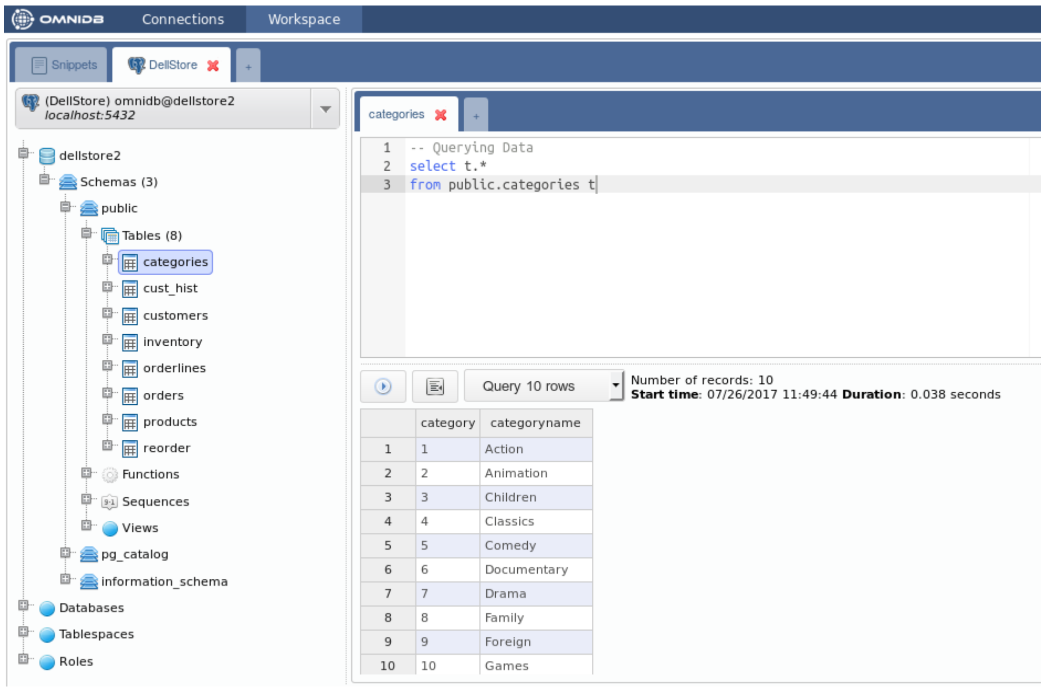
Now click to expand the node *Tables*, and you will see all tables contained in the schema `public`. Expand any table and you will see its columns, primary key, foreign keys, unique constraints and indexes. Each column is also expansible, displaying data type and nullable constraint.



In order to view records inside a table, right click it and choose *Data Actions > Query Data*.



Notice that OmniDB fills the current SQL editor with a simple query to list table records. The records are displayed in a grid right below the editor. This grid can be controlled with keyboard as if you were using a spreadsheet manager. You can also copy data from single cells or block of cells (that can be selected with the keyboard or mouse) and paste on any spreadsheet manager.



```
-- Querying Data
select t.*
from public.categories t
```

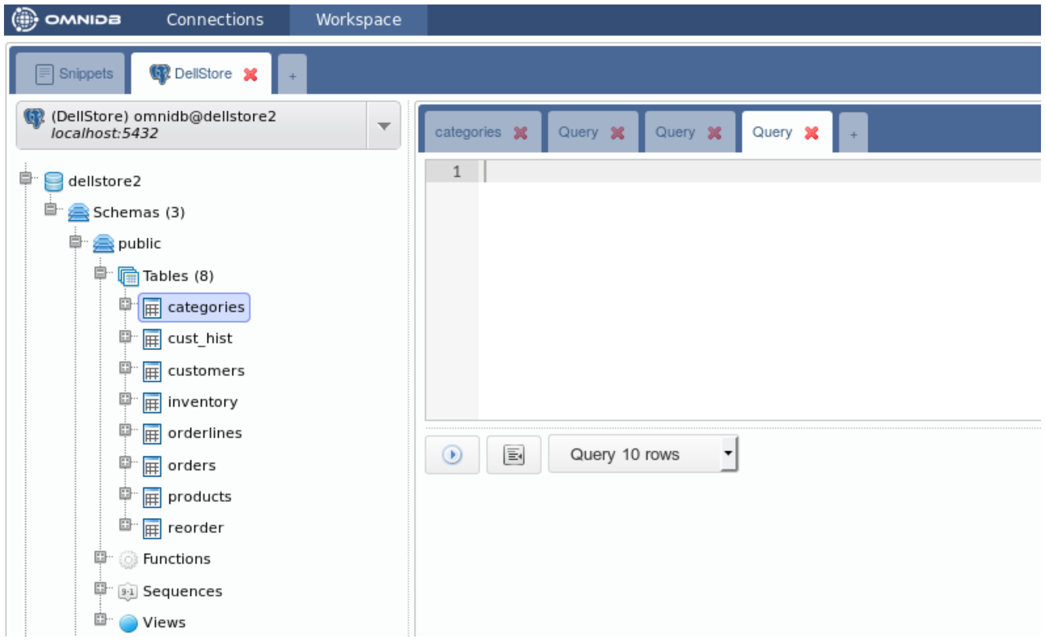
Number of records: 10
Start time: 07/26/2017 11:49:44 **Duration**: 0.038 seconds

	category	categoryname
1	1	Action
2	2	Animation
3	3	Children
4	4	Classics
5	5	Comedy
6	6	Documentary
7	7	Drama
8	8	Family
9	9	Foreign
10	10	Games

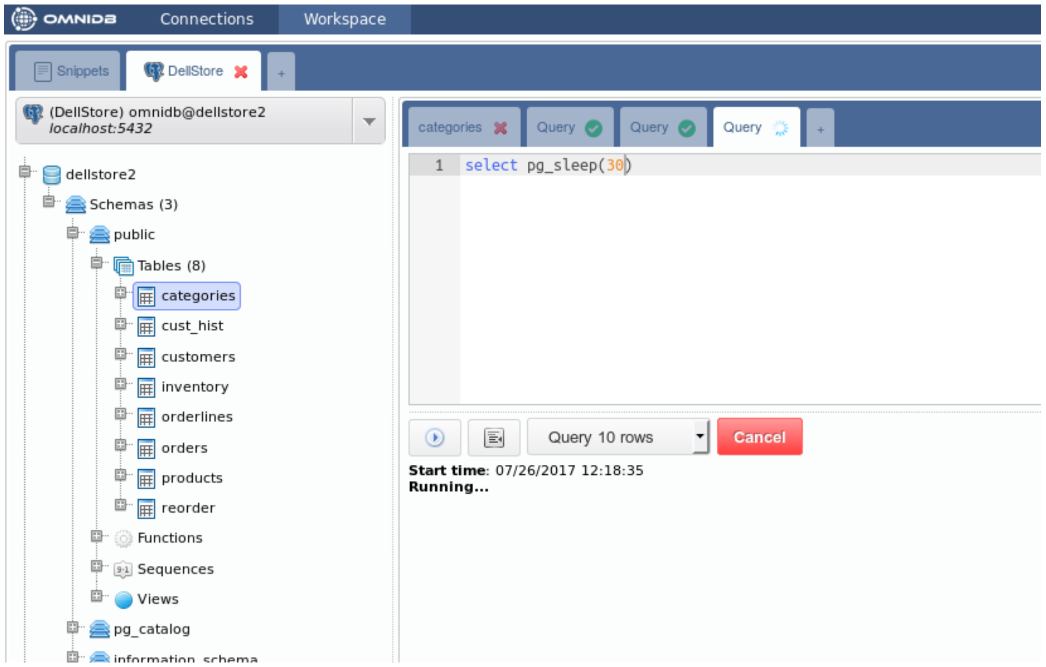
You can edit the query on the SQL editor, writing simple or more complex queries and clicking on the action button. You can control how many records should be displayed (10, 100, 1000 or all rows). More details in the next chapters.

5.4 Working with multiple tabs inside the same connection

Inside a single connection, you can create several inner tabs by clicking on the last little tab with a cross. Each new inner tab will be a *Query Tab*.



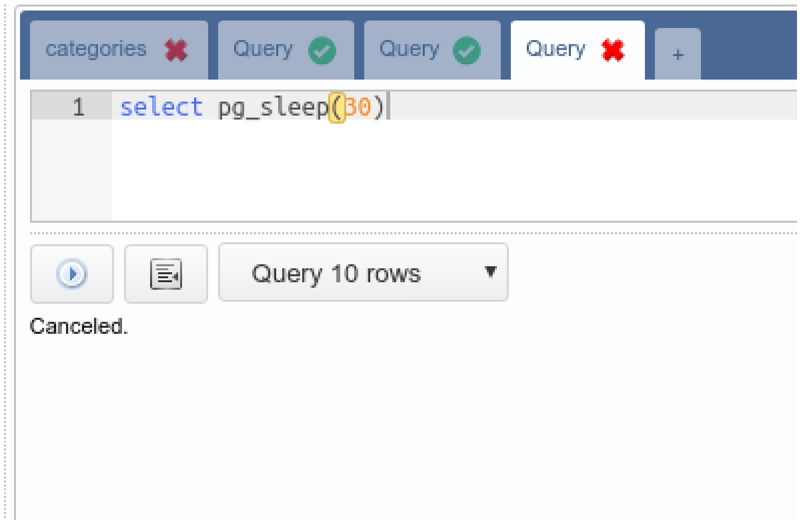
On OmniDB, you can execute several SQL statements and procedures in parallel. When it is executing, an icon will be shown in the tab to indicate its current state. If some process is finished and it is not in the current tab, that tab will show a green icon indicating the routine being executed there is now finished.



```
select pg_sleep(30)
```

Start time: 07/26/2017 12:18:35
Running...

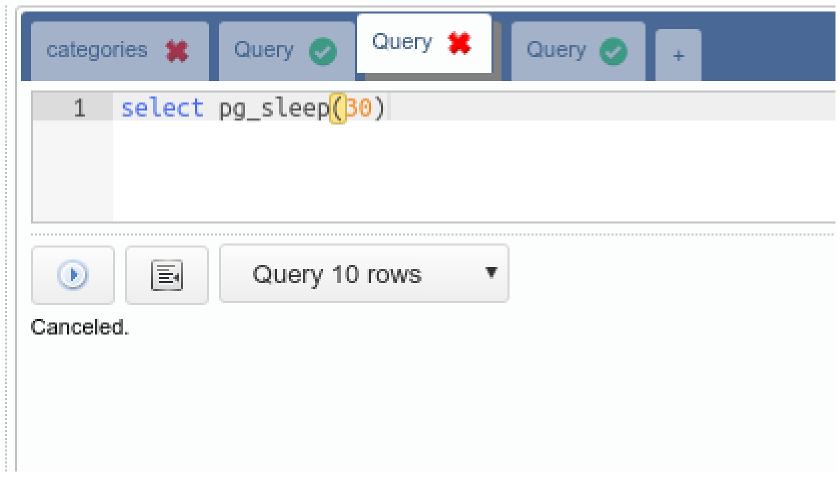
By clicking in the *Cancel button*, you can cancel a process running inside the database.



```
select pg_sleep(30)
```

Canceled.

You can also drag and drop a tab to change its order. This works with both inner and outer tabs.



```
select pg_sleep(30)
```

Canceled.

Additionally, you can use keyboard shortcuts to manage inner tabs (SQL Query) and outer tabs (Connection):

- **Ctrl-Insert**: Insert a new inner tab
- **Ctrl-Delete**: Removes an inner tab
- **Ctrl-<**: Change focus to inner tab at left
- **Ctrl->**: Change focus to inner tab at right
- **Ctrl-Shift-Insert**: Insert a new outer tab
- **Ctrl-Shift-Delete**: Removes an outer tab
- **Ctrl-Shift-<**: Change focus to outer tab at left
- **Ctrl-Shift->**: Change focus to outer tab at right

6 Table Management

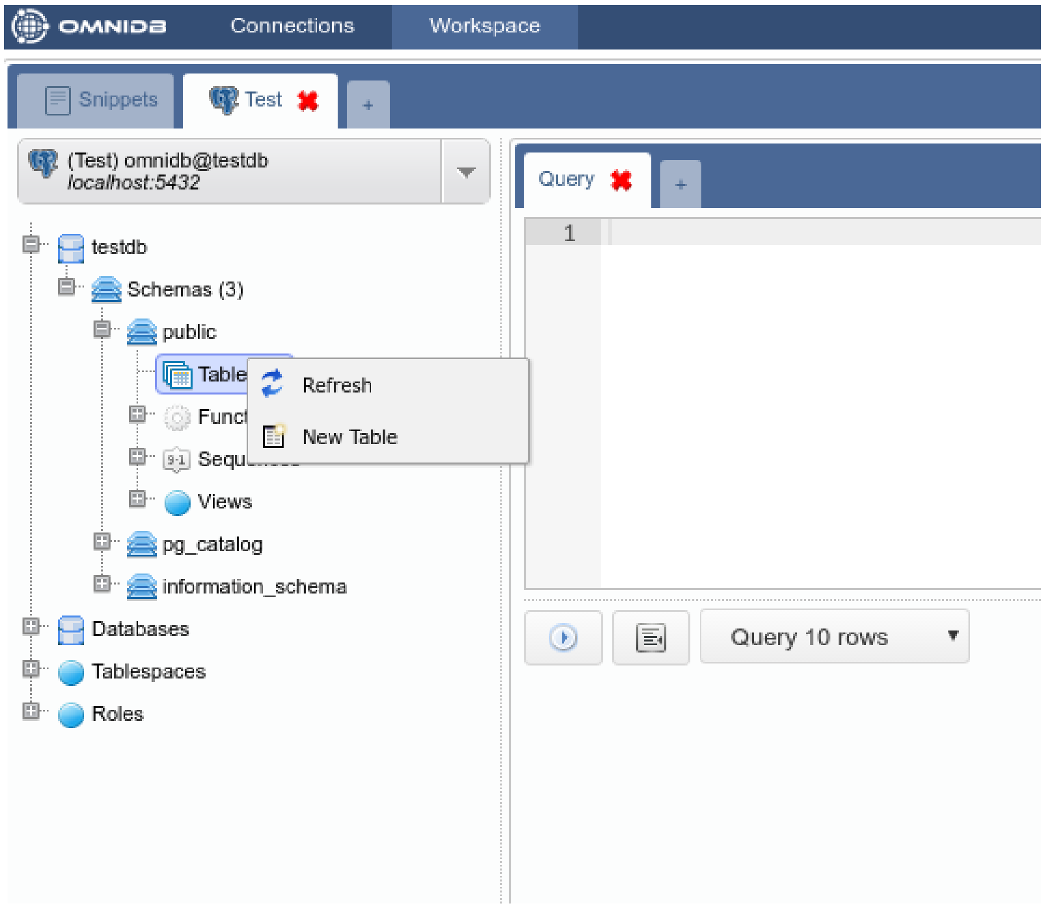
6.1 Creating tables

OmniDB has a table creation interface that lets you configure columns, constraints and indexes. A couple of observations should be mentioned:

- Most DBMS automatically create indexes when primary keys and unique constraints are created. Because of that, the indexes tab is only available after creating the table.
- Each DBMS has its unique characteristics and limitations regarding table creation and the OmniDB interface reflects these limitations. For

instance, SQLite does not allow us to change existing columns and constraints. Because of that, the interface lets us change only table name and add new columns when dealing with SQLite databases (it is still not the case in OmniDB Python version, as it currently supports only PostgreSQL databases).

We will create example tables (*Customer* and *Address*) in the `testdb` database we connected to earlier. Right click on the **Tables** node and select the **New Table** action:



We will create the table *Customer* with a primary key that will be referenced by the table *Address*:

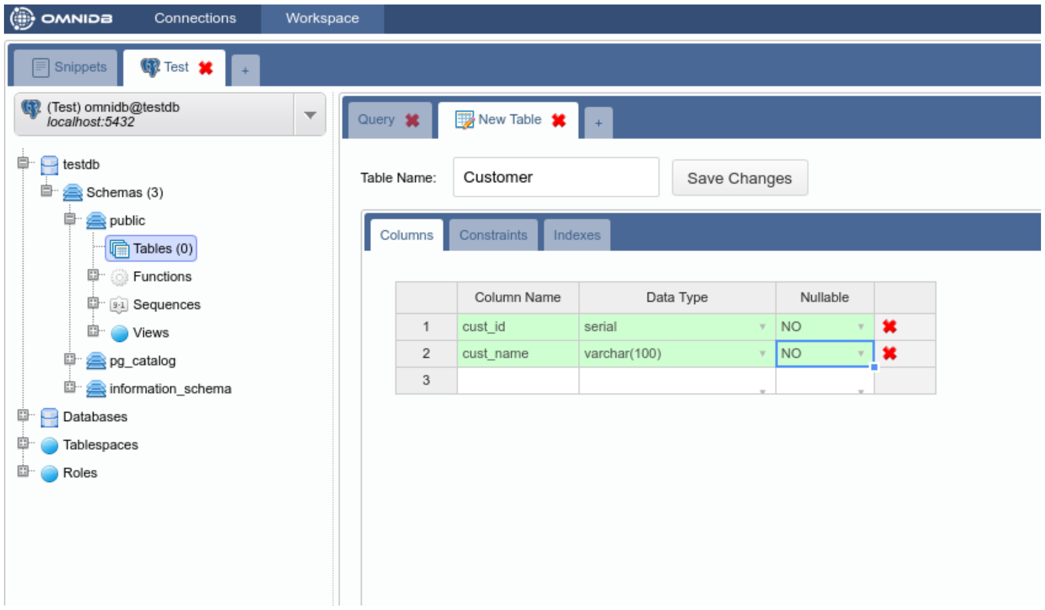


Table Name: Customer

	Column Name	Data Type	Nullable	
1	cust_id	serial	NO	✖
2	cust_name	varchar(100)	NO	✖
3				

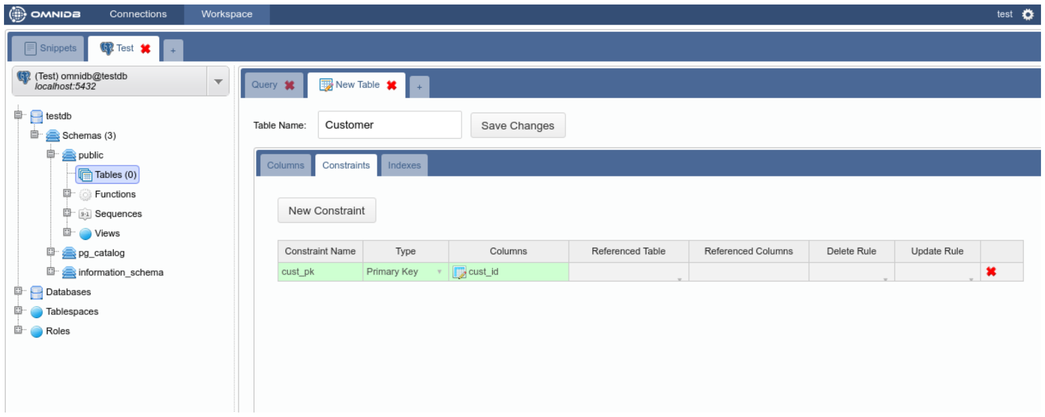


Table Name: Customer

Constraint Name	Type	Columns	Referenced Table	Referenced Columns	Delete Rule	Update Rule	
cust_pk	Primary Key	cust_id					✖

Note how the table appers in the *Tables* tree node:

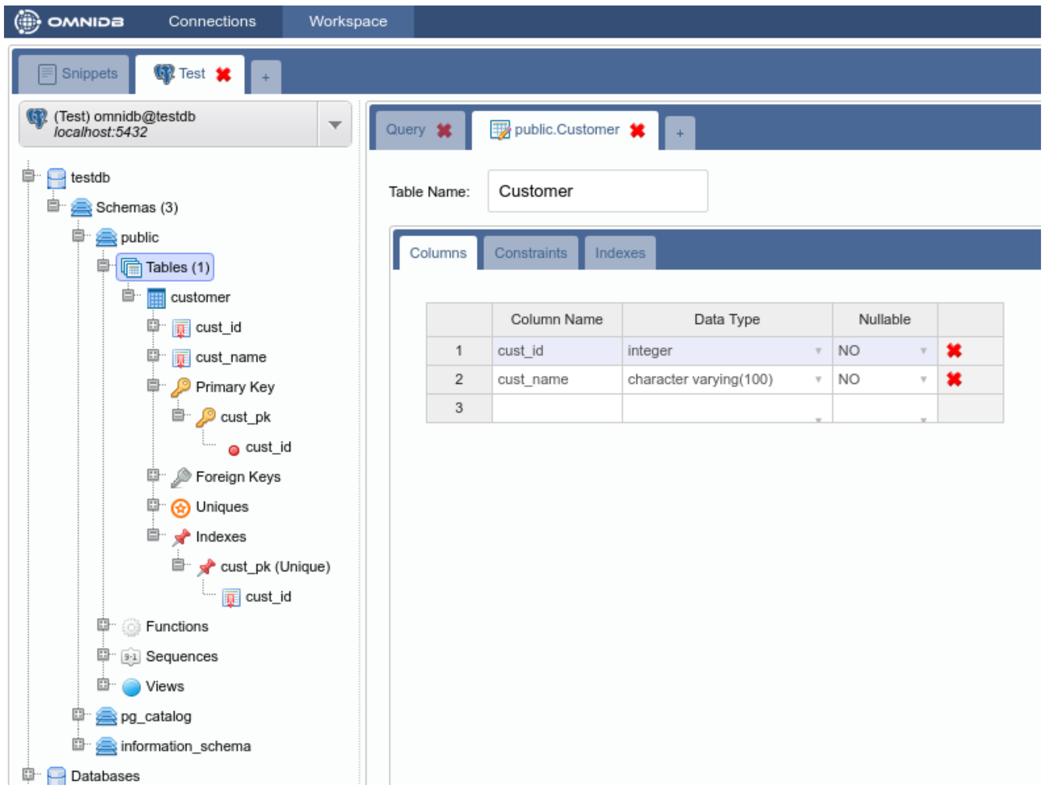


Table Name: Customer

	Column Name	Data Type	Nullable	
1	cust_id	integer	NO	✖
2	cust_name	character varying(100)	NO	✖
3				

Now create the table *Address* with a primary key and a foreign key:

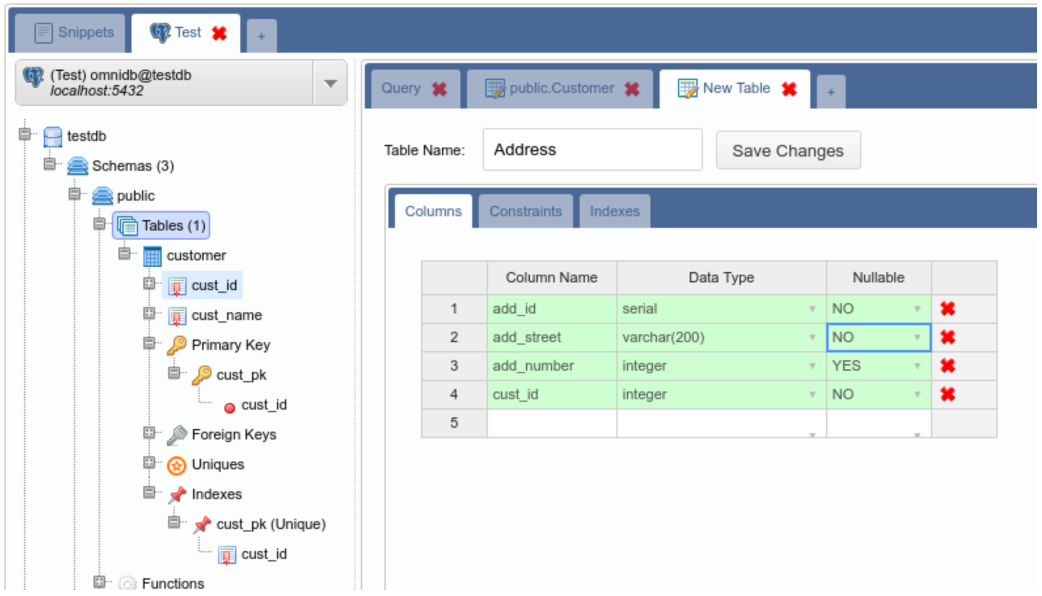


Table Name: Address

	Column Name	Data Type	Nullable	
1	add_id	serial	NO	✖
2	add_street	varchar(200)	NO	✖
3	add_number	integer	YES	✖
4	cust_id	integer	NO	✖
5				

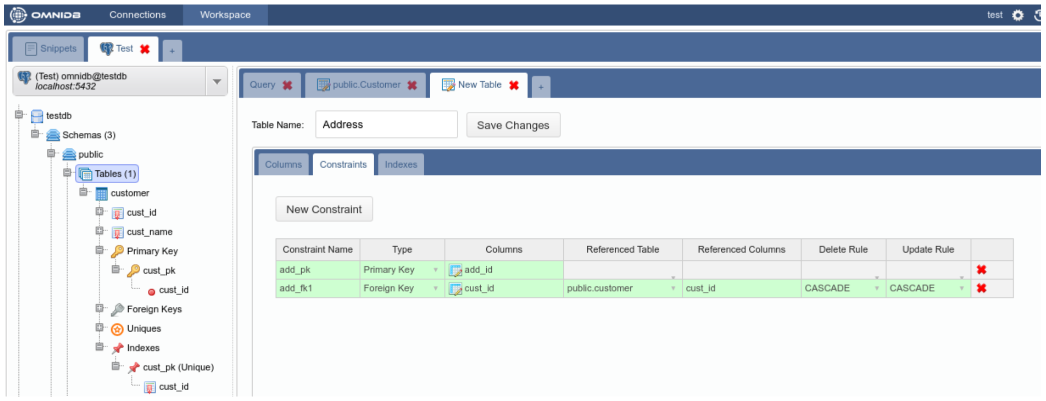
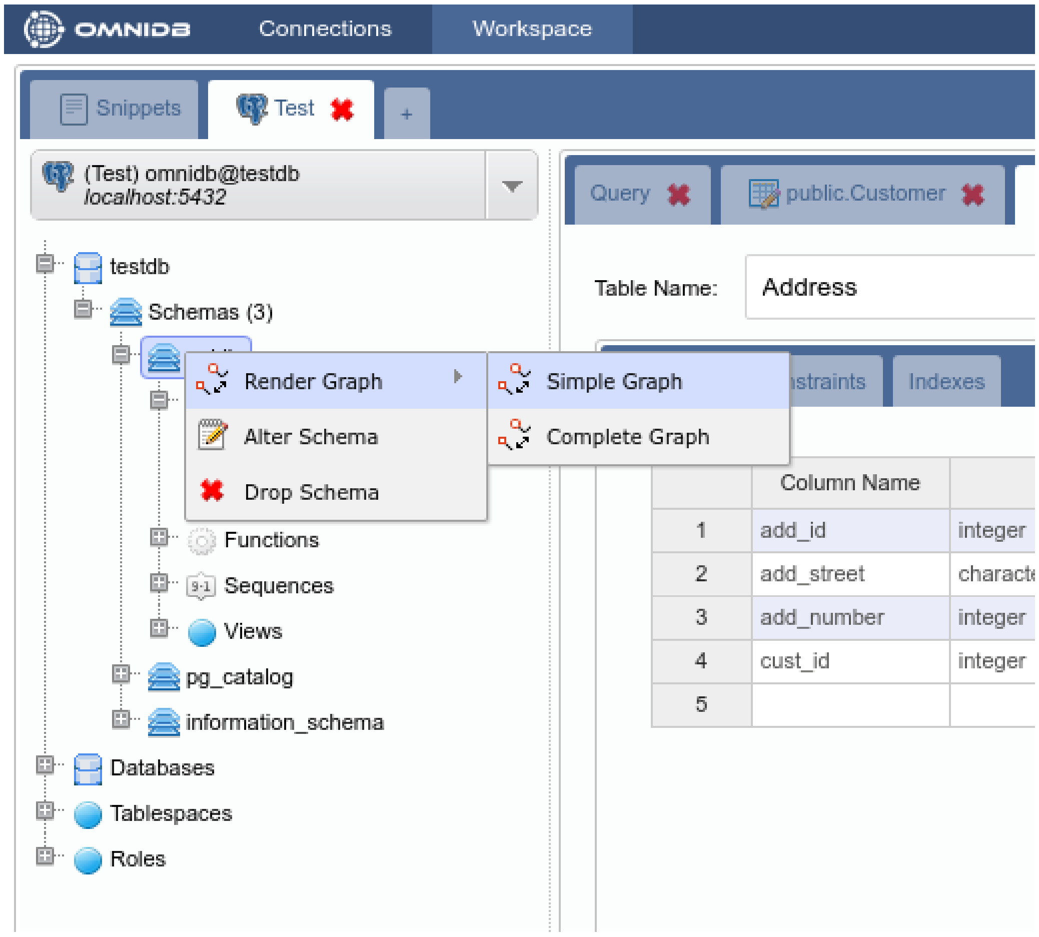


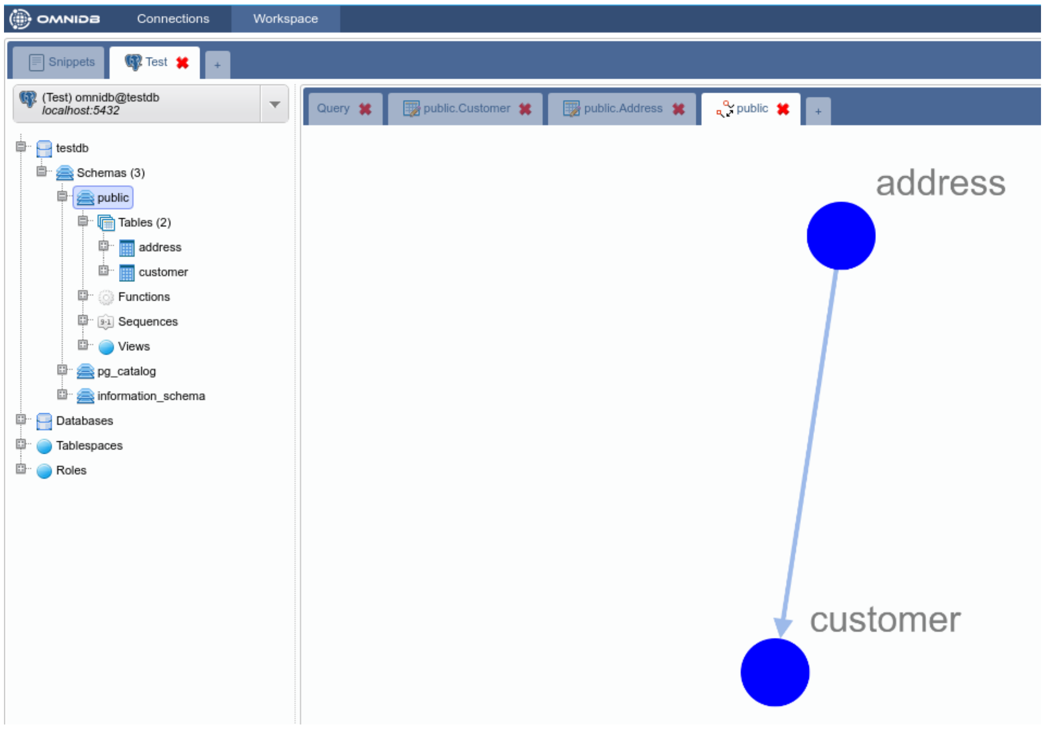
Table Name: Address

Constraint Name	Type	Columns	Referenced Table	Referenced Columns	Delete Rule	Update Rule	
add_pk	Primary Key	add_id					✖
add_fk1	Foreign Key	cust_id	public.customer	cust_id	CASCADE	CASCADE	✖

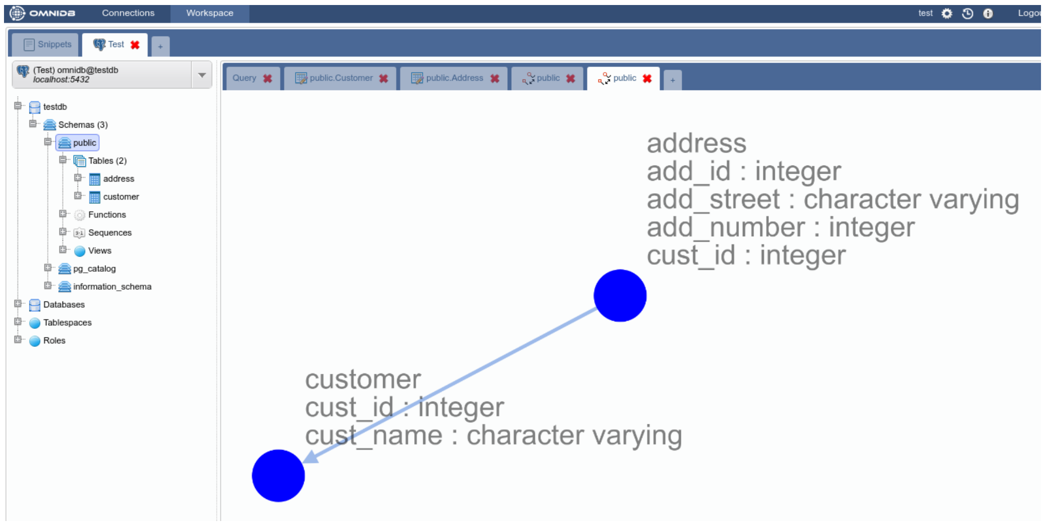
At this point we have two tables in schema `public`. The schema structure can be seen with the graph feature by right clicking on the schema `public` node of the tree and selecting *Render Graph > Simple Graph*:



	Column Name	
1	add_id	integer
2	add_street	characte
3	add_number	integer
4	cust_id	integer
5		

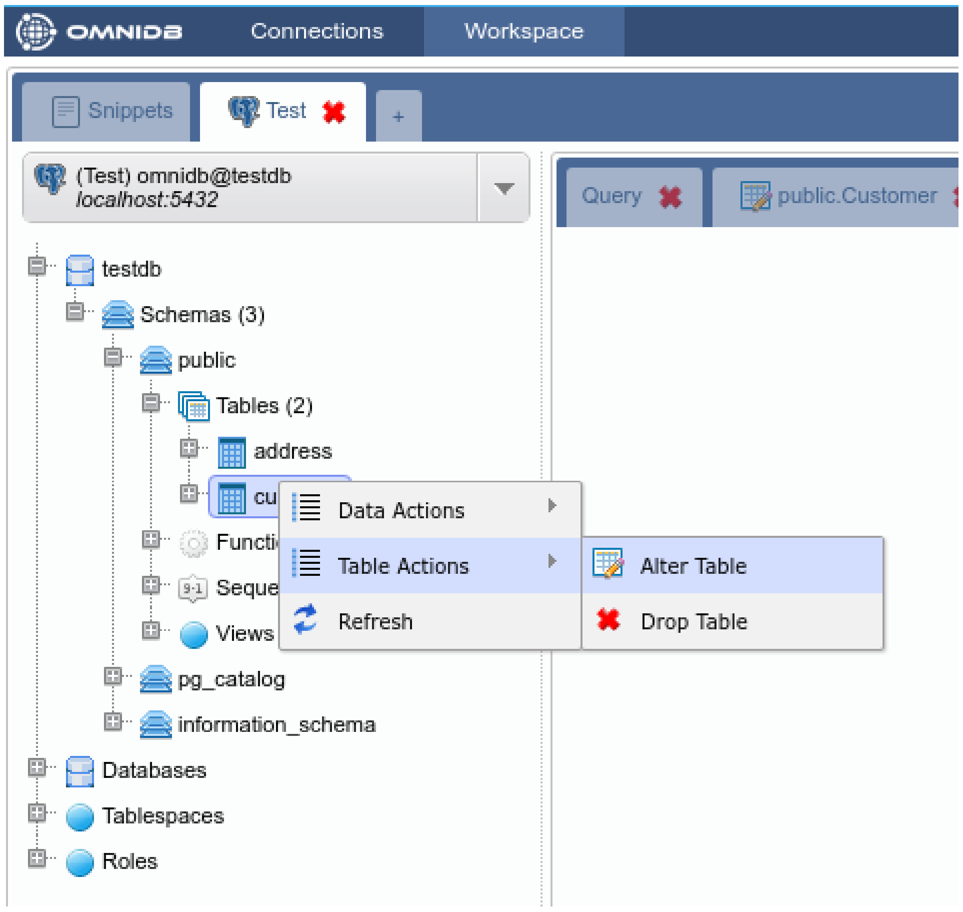


And this is what the *Complete Graph* looks like:



6.2 Editing tables

OmniDB also lets you edit existing tables (always following DBMS limitations). To test this feature we will add a new column to the table *Customer*. To access the alter table interface just right click the table node and select the action *Table Actions > Alter Table*:



Add the column *cust_age* and save:

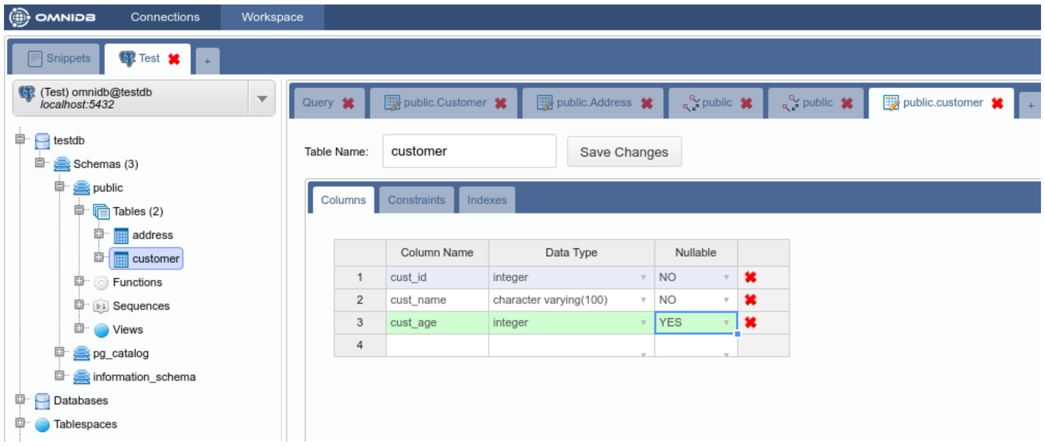
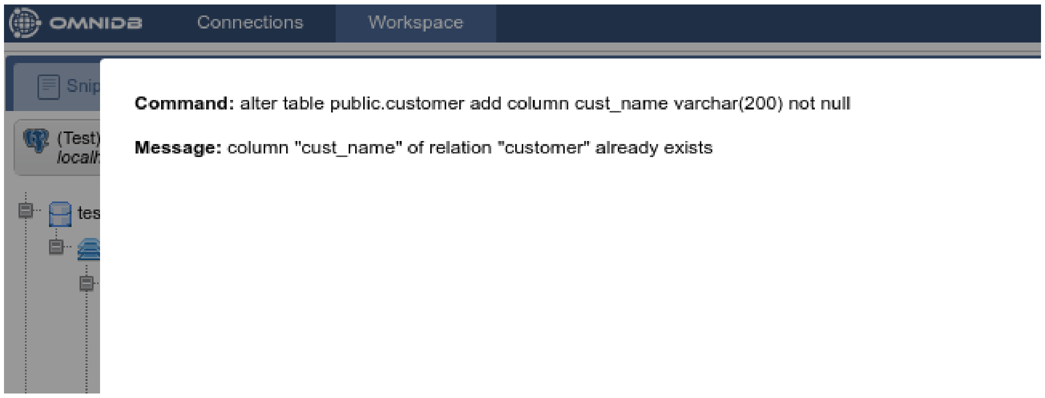


Table Name: customer

	Column Name	Data Type	Nullable
1	cust_id	integer	NO
2	cust_name	character varying(100)	NO
3	cust_age	integer	YES
4			

The interface is capable of detecting errors that may occur during alter table operations, showing the command and the error that occurred. To demonstrate it we will try to add the column *cust_name*, which already belongs to this table:

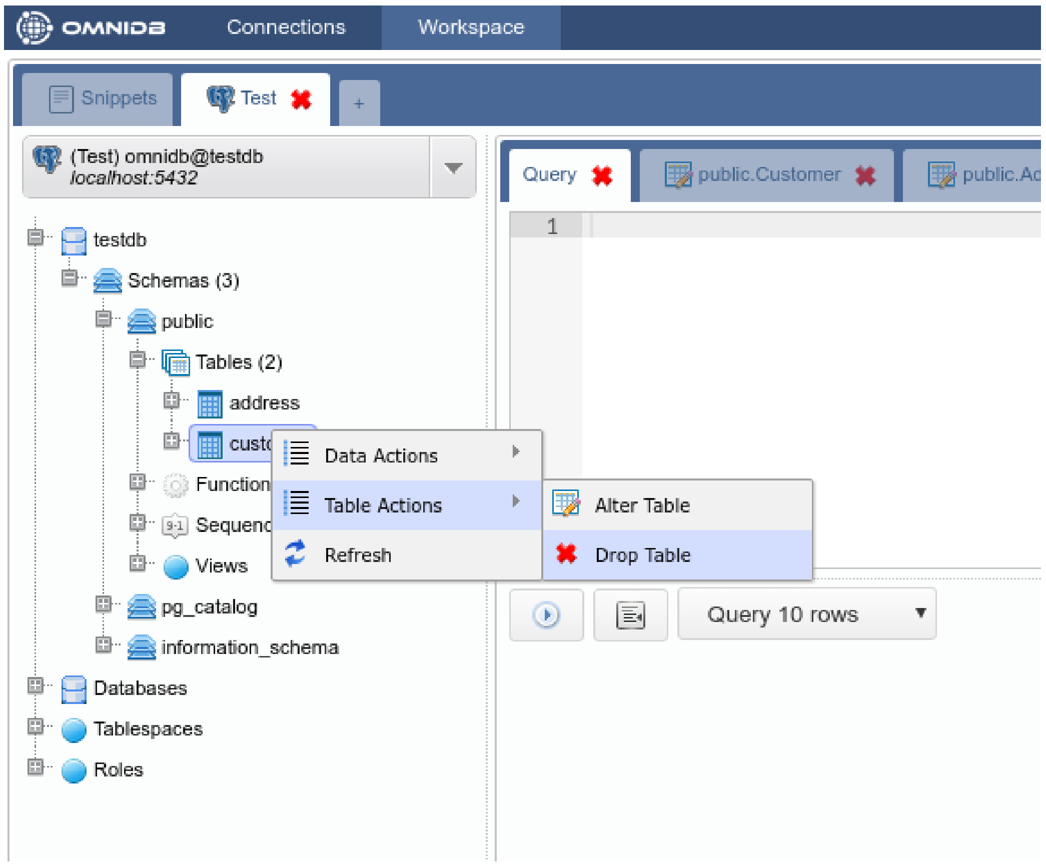


Command: alter table public.customer add column cust_name varchar(200) not null

Message: column "cust_name" of relation "customer" already exists

6.3 Removing tables

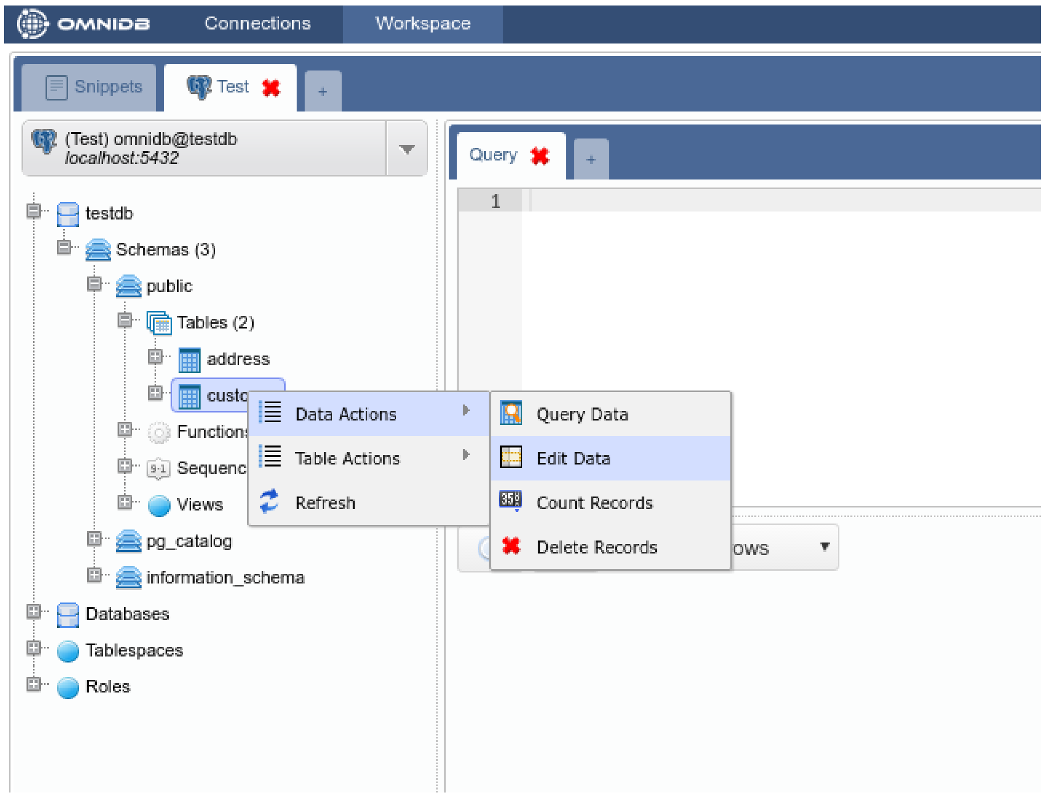
In order to remove a table just right click the table node and select the action *Table Actions > Drop Table*:

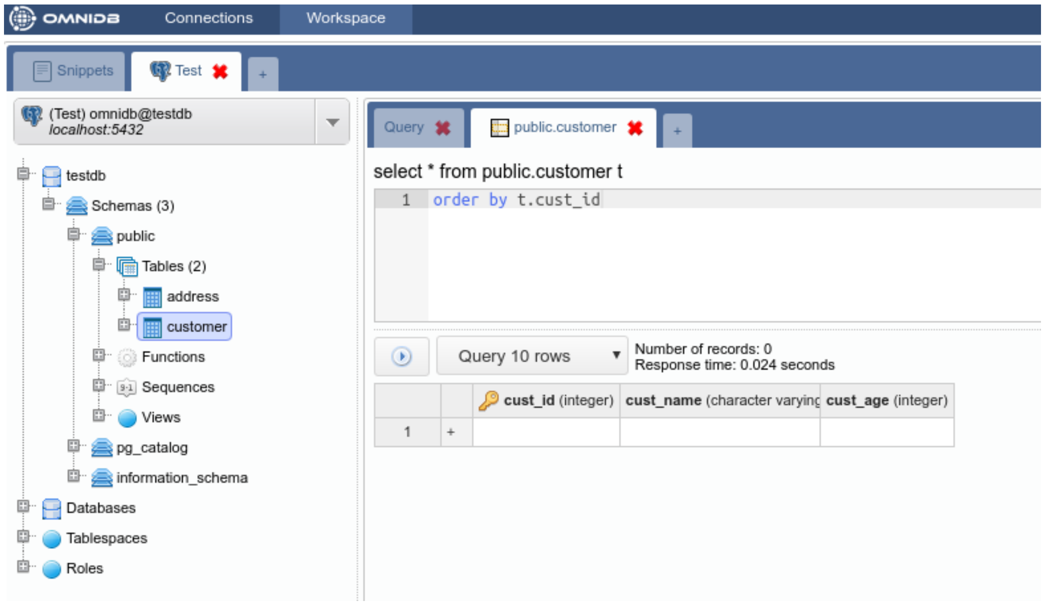


7 Data Management

The tool allow us to edit records contained in tables through a very simple and intuitive interface. Given that only a few DBMS have unique identifiers for table records, we opted to allow data editing and removal only for tables that have primary keys. Tables that do not have it can only receive new records.

To access the record editing interface, right click the table node and select the action *Data Actions* > *Edit Data*:





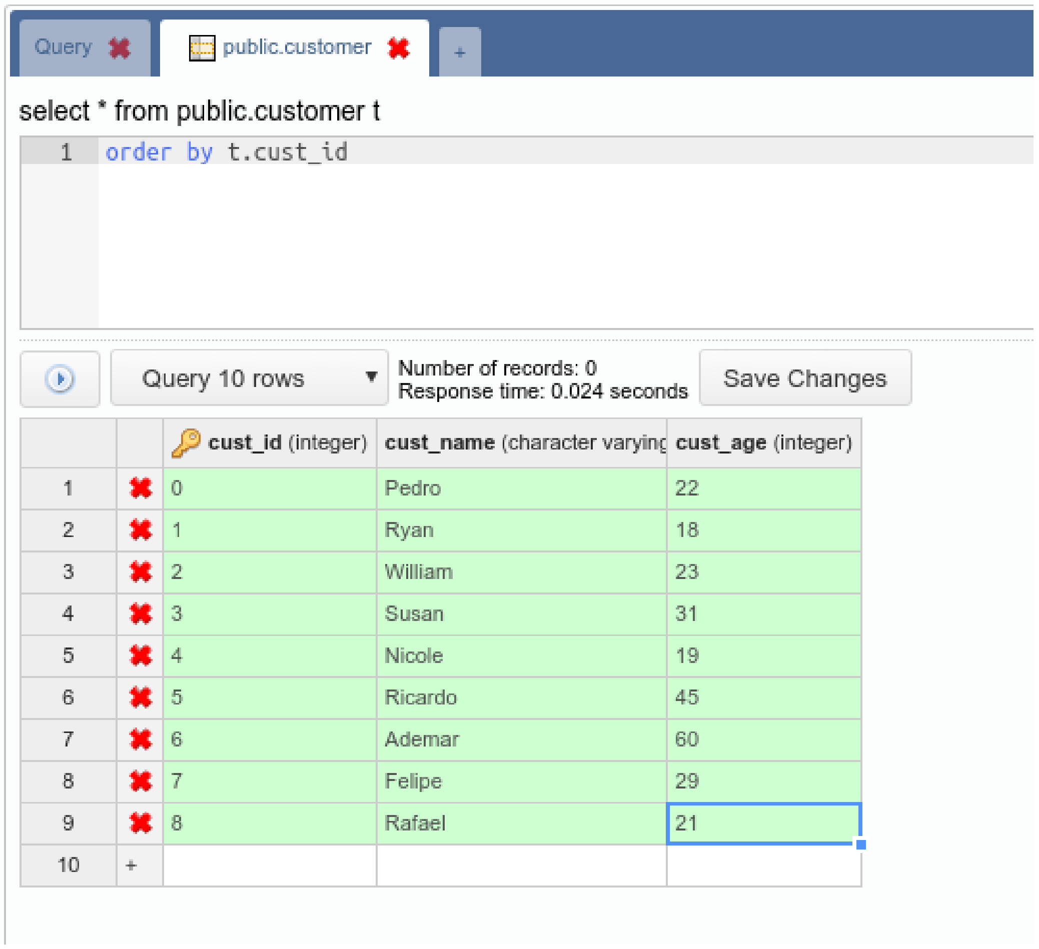
```
select * from public.customer t
order by t.cust_id
```

```
Number of records: 0
Response time: 0.024 seconds
```

		cust_id (integer)	cust_name (character varying	cust_age (integer)
1	+			

The interface has a SQL editor where you can filter and order records. To prevent that the interface requests too many records, there is a field that limits the number of records to be displayed. The records grid has column names and data types. Columns that belong to the primary key have a key icon next to their names.

The row of the grid that have the symbol * is the row to add new records. Let us insert some records in the table `Customer`:

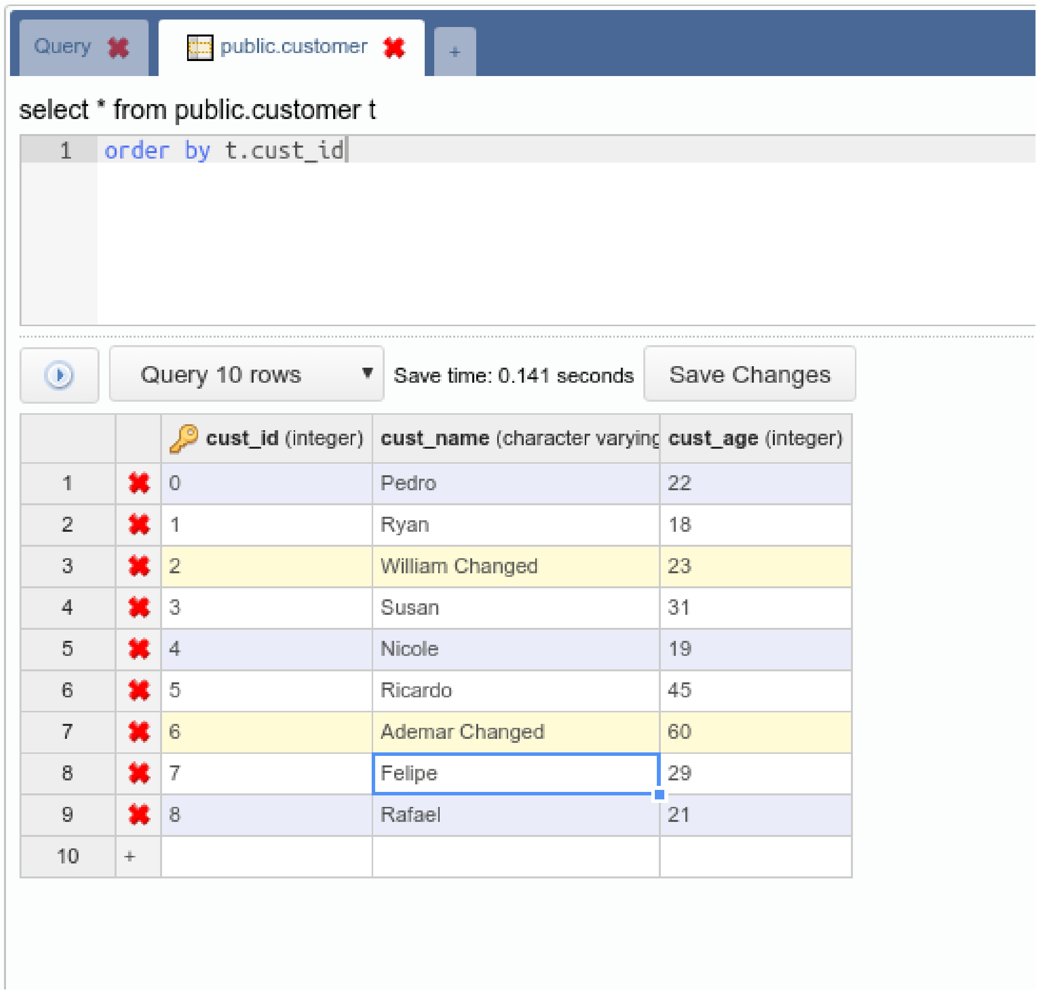


```
select * from public.customer t
order by t.cust_id
```

```
Number of records: 0
Response time: 0.024 seconds
```

		cust_id (integer)	cust_name (character varying	cust_age (integer)
1	✖	0	Pedro	22
2	✖	1	Ryan	18
3	✖	2	William	23
4	✖	3	Susan	31
5	✖	4	Nicole	19
6	✖	5	Ricardo	45
7	✖	6	Ademar	60
8	✖	7	Felipe	29
9	✖	8	Rafael	21
10	+			

After saving, the records will be inserted and can be edited (only because this table has a primary key). Let's change the *cust_name* of some of the existing records:



Query | public.customer | +

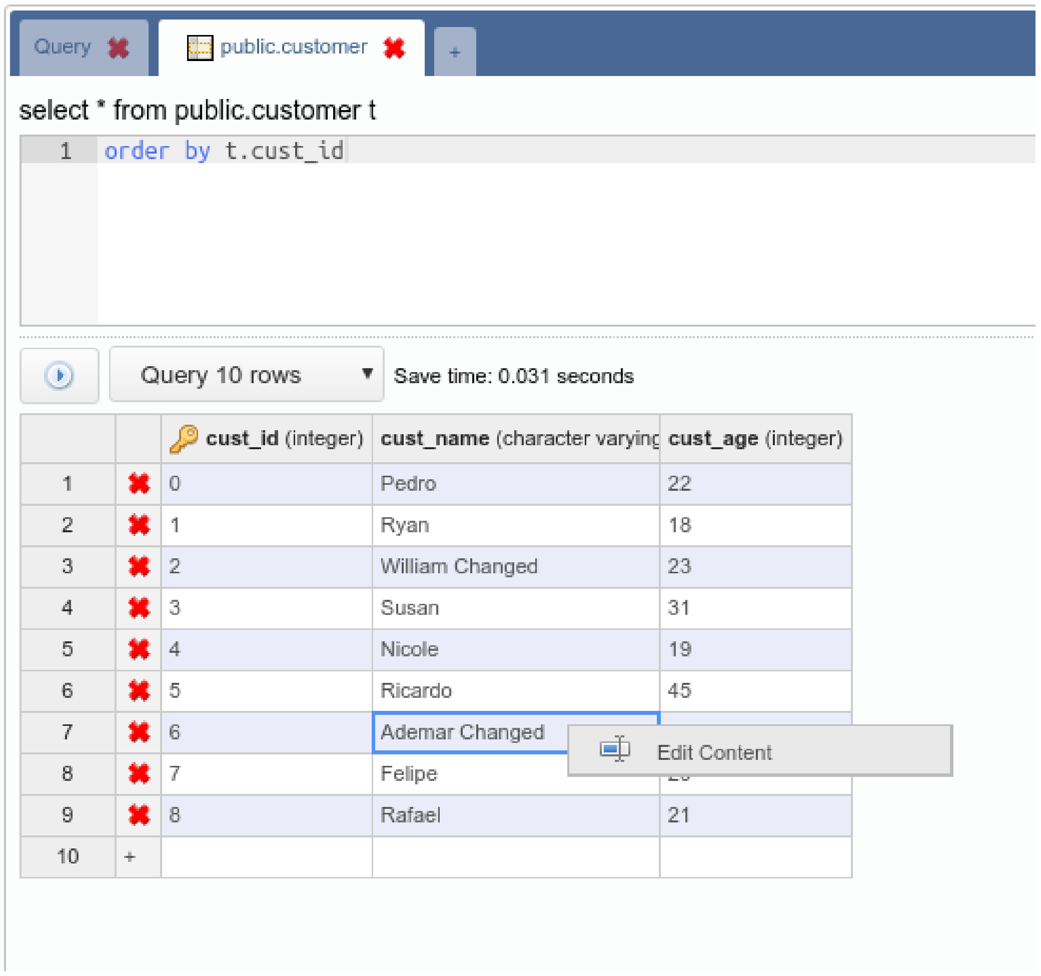
select * from public.customer t

```
order by t.cust_id
```

Query 10 rows | Save time: 0.141 seconds | Save Changes

		cust_id (integer)	cust_name (character varying	cust_age (integer)
1	✖	0	Pedro	22
2	✖	1	Ryan	18
3	✖	2	William Changed	23
4	✖	3	Susan	31
5	✖	4	Nicole	19
6	✖	5	Ricardo	45
7	✖	6	Ademar Changed	60
8	✖	7	Felipe	29
9	✖	8	Rafael	21
10	+			

Tables can have fields with values represented by very long strings. To help edit these fields, OmniDB has an interface that can be accessed by right clicking the specific cell:

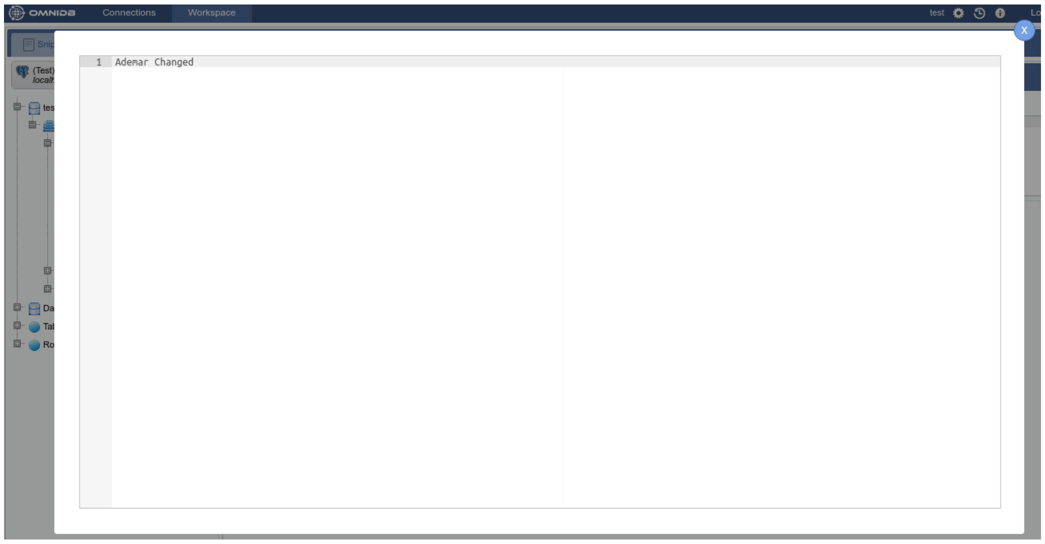


select * from public.customer t

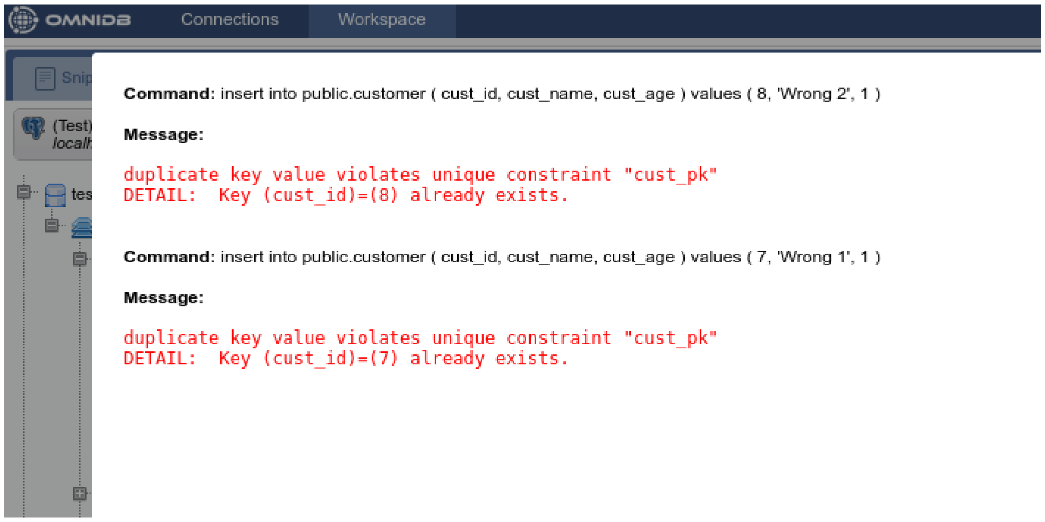
```
order by t.cust_id
```

Save time: 0.031 seconds

		cust_id (integer)	cust_name (character varying	cust_age (integer)
1	✖	0	Pedro	22
2	✖	1	Ryan	18
3	✖	2	William Changed	23
4	✖	3	Susan	31
5	✖	4	Nicole	19
6	✖	5	Ricardo	45
7	✖	6	Ademar Changed	
8	✖	7	Felipe	[illegible]
9	✖	8	Rafael	21
10	+			


```
Adenar Changed
```

The interface detects errors that may occur during operations related to records. To demonstrate, let us insert two records with existing `cust_id` (primary key):


Command: insert into public.customer (cust_id, cust_name, cust_age) values (8, 'Wrong 2', 1)

Message:

```
duplicate key value violates unique constraint "cust_pk"
DETAIL:  Key (cust_id)=(8) already exists.
```

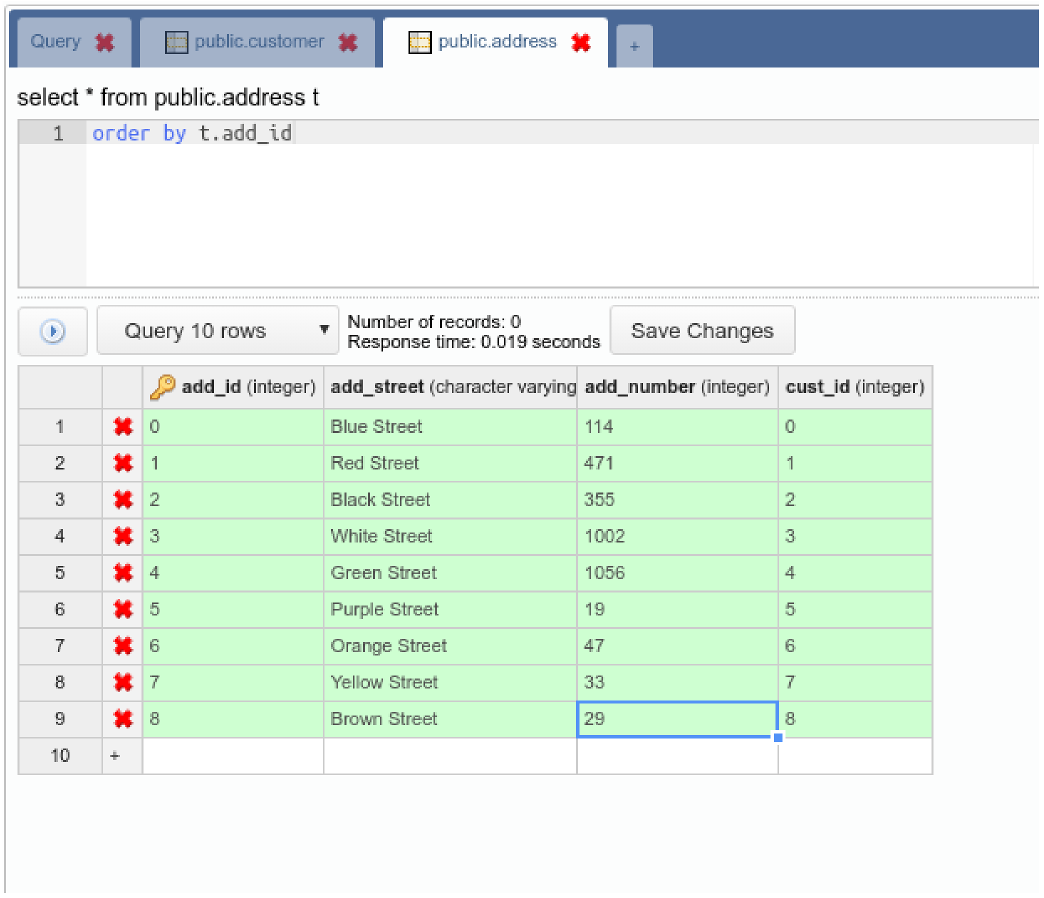
Command: insert into public.customer (cust_id, cust_name, cust_age) values (7, 'Wrong 1', 1)

Message:

```
duplicate key value violates unique constraint "cust_pk"
DETAIL:  Key (cust_id)=(7) already exists.
```

It shows which commands tried to be executed and the respective errors.

To complete this chapter, let's add some records to the *Address* table:



```
select * from public.address t
order by t.add_id
```

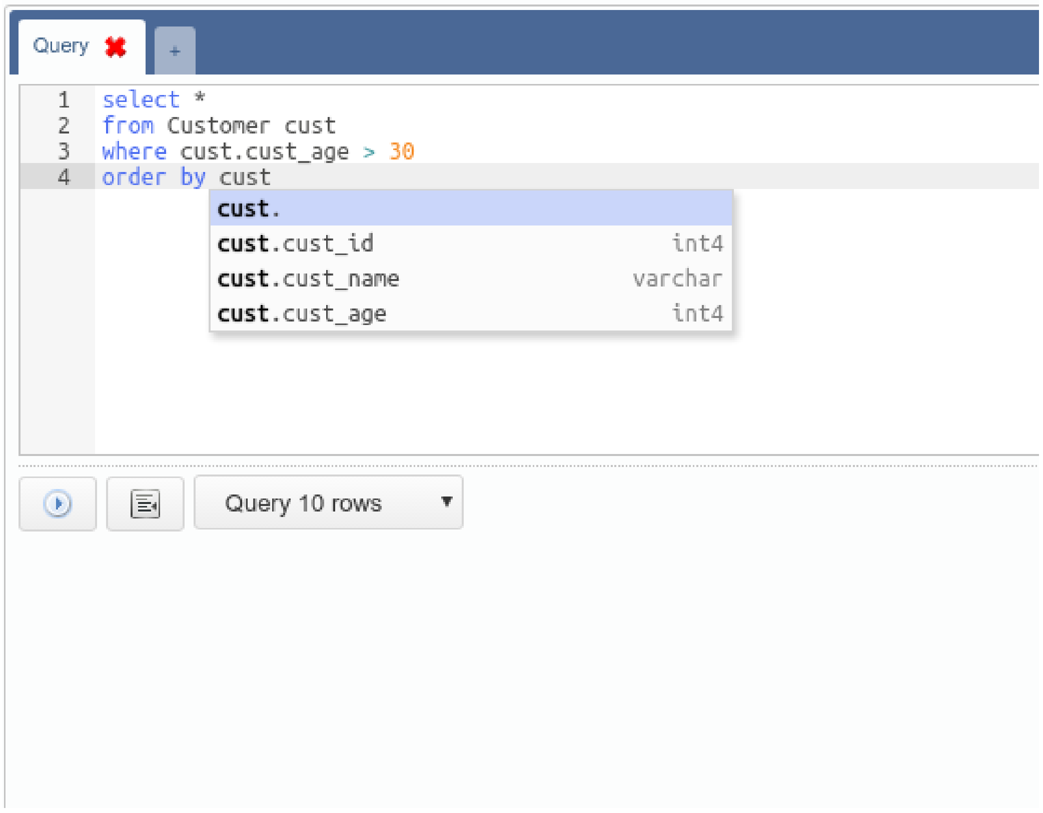
Number of records: 0
Response time: 0.019 seconds

		add_id (integer)	add_street (character varying	add_number (integer)	cust_id (integer)
1	✖	0	Blue Street	114	0
2	✖	1	Red Street	471	1
3	✖	2	Black Street	355	2
4	✖	3	White Street	1002	3
5	✖	4	Green Street	1056	4
6	✖	5	Purple Street	19	5
7	✖	6	Orange Street	47	6
8	✖	7	Yellow Street	33	7
9	✖	8	Brown Street	29	8
10	+				

8 SQL Editor

The tool comes with a tab system where each tab contains a SQL editor, an action button, an indent button, a field to select the type of command and a space to display the result.

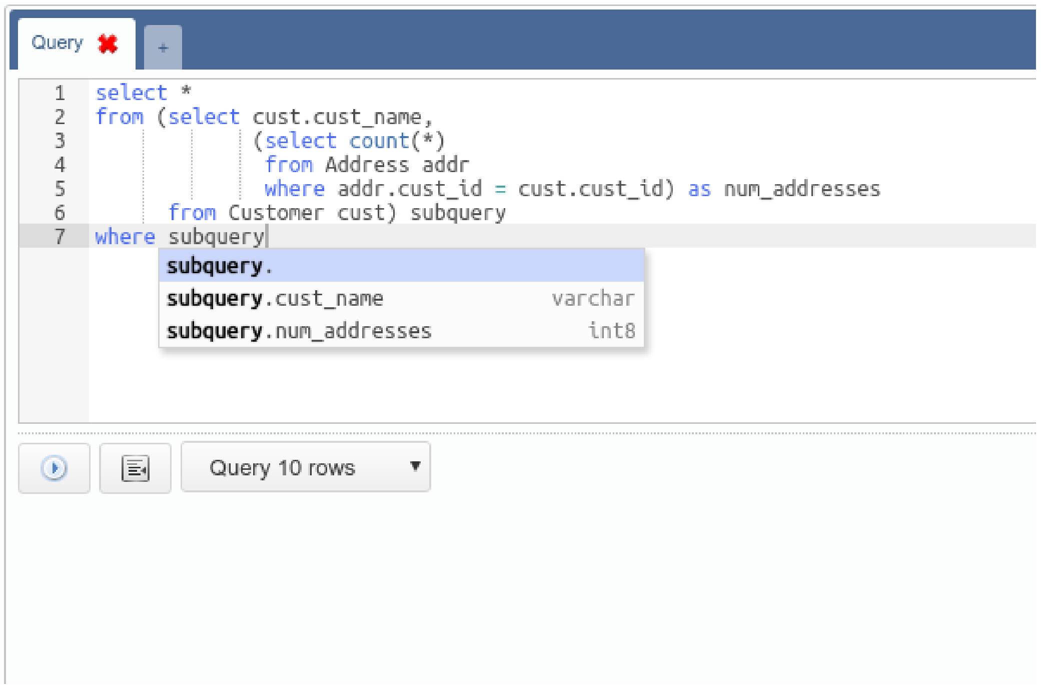
The SQL editor has a feature that helps a lot when creating new queries: SQL code completion. With this feature it is possible to autocomplete columns contained in a table referenced by an alias. To open the autocomplete interface you just have to type the alias and then the dot character:



```
select *
from Customer cust
where cust.cust_age > 30
order by cust
```

```
cust.
cust.cust_id          int4
cust.cust_name        varchar
cust.cust_age         int4
```

Besides autocompleting table columns the editor also searches for columns contained in subqueries:

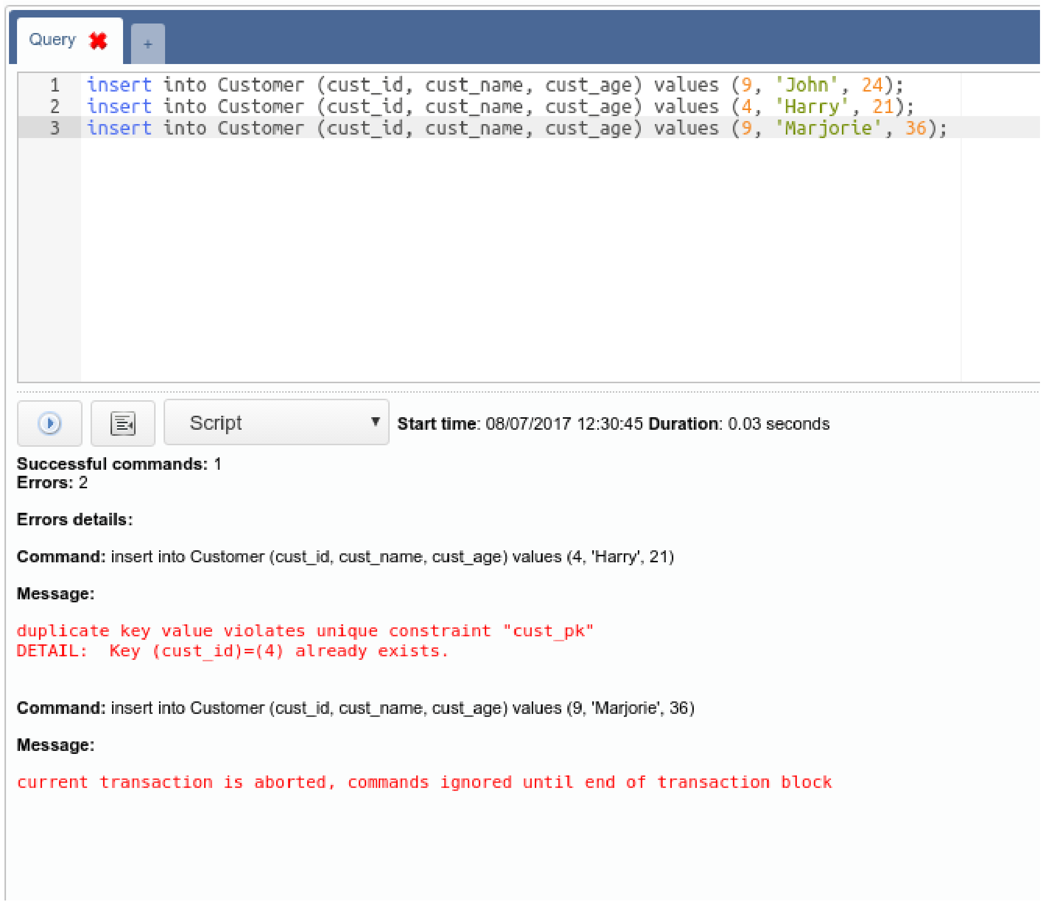


```
select *
from (select cust.cust_name,
             (select count(*)
              from Address addr
              where addr.cust_id = cust.cust_id) as num_addresses
      from Customer cust) subquery
where subquery
```

```
subquery.
subquery.cust_name            varchar
subquery.num_addresses           int8
```

The field to select the type of command has the following options:

- **Script**: script execution, which is a sequence of commands separated by semicolon:



```
insert into Customer (cust_id, cust_name, cust_age) values (9, 'John', 24);
insert into Customer (cust_id, cust_name, cust_age) values (4, 'Harry', 21);
insert into Customer (cust_id, cust_name, cust_age) values (9, 'Marjorie', 36);
```

Start time: 08/07/2017 12:30:45 **Duration**: 0.03 seconds

Successful commands: 1
Errors: 2

Errors details:

Command: insert into Customer (cust_id, cust_name, cust_age) values (4, 'Harry', 21)

Message:

```
duplicate key value violates unique constraint "cust_pk"
DETAIL:  Key (cust_id)=(4) already exists.
```

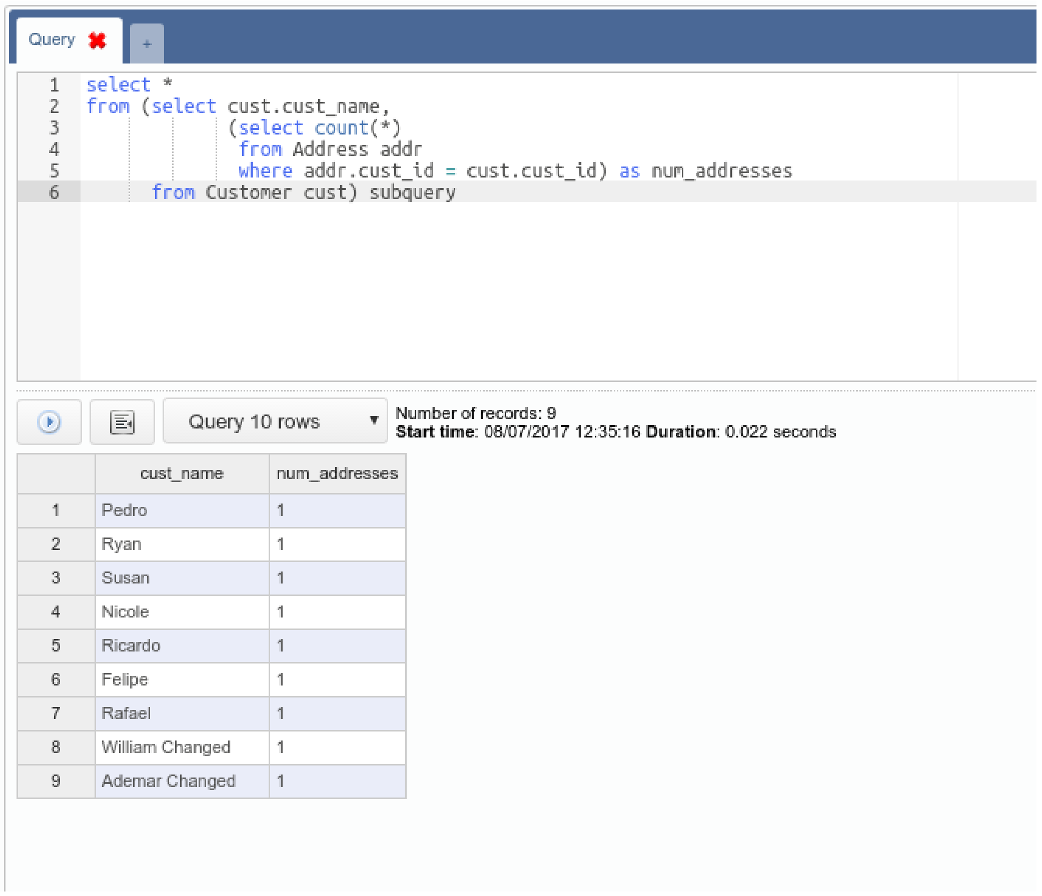
Command: insert into Customer (cust_id, cust_name, cust_age) values (9, 'Marjorie', 36)

Message:

```
current transaction is aborted, commands ignored until end of transaction block
```

The return shows the response time, the number of commands that were successfully executed, the number of commands that generated errors and a list displaying each error.

- **Execute**: execution of only one command. The return shows the response time or an error.
- **Query (10, 100, 1000, all) rows**: execution of a query that returns a set of records, which are displayed on a grid. Just like in the record editing interface each cell can be visualized separately by right clicking it:



```
select *
from (select cust.cust_name,
            (select count(*)
             from Address addr
             where addr.cust_id = cust.cust_id) as num_addresses
      from Customer cust) subquery
```

Number of records: 9
Start time: 08/07/2017 12:35:16 **Duration**: 0.022 seconds

	cust_name	num_addresses
1	Pedro	1
2	Ryan	1
3	Susan	1
4	Nicole	1
5	Ricardo	1
6	Felipe	1
7	Rafael	1
8	William Changed	1
9	Ademar Changed	1

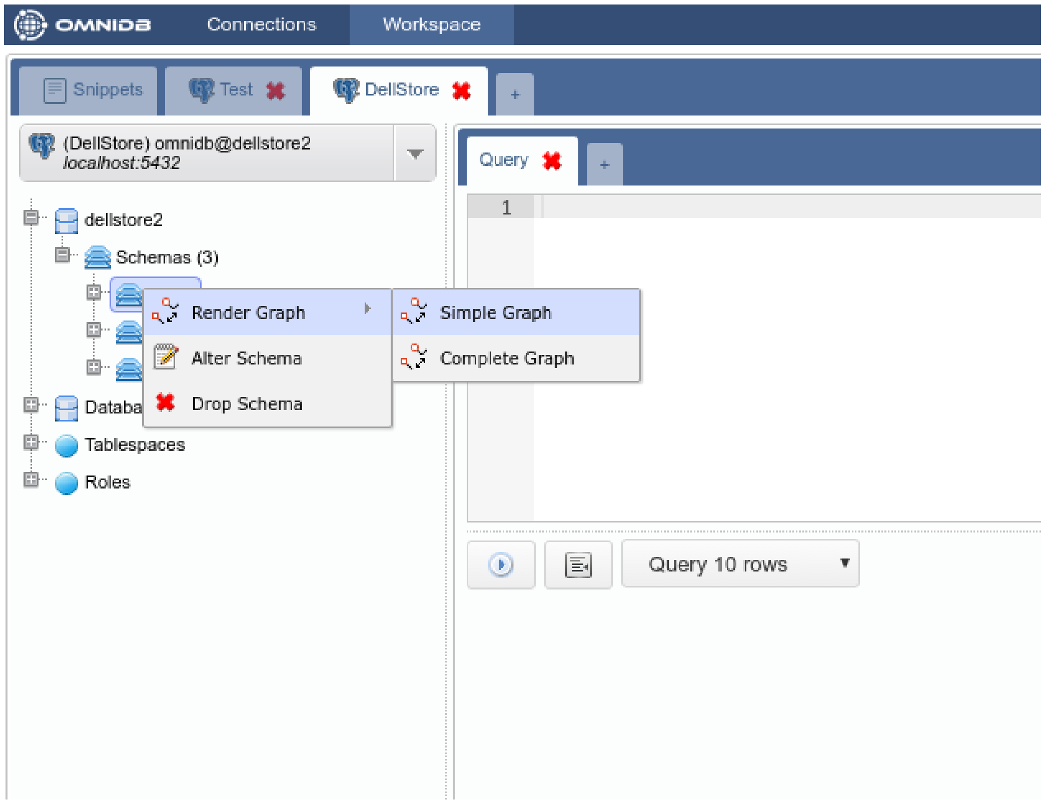
9 Graph with Tables and Relations

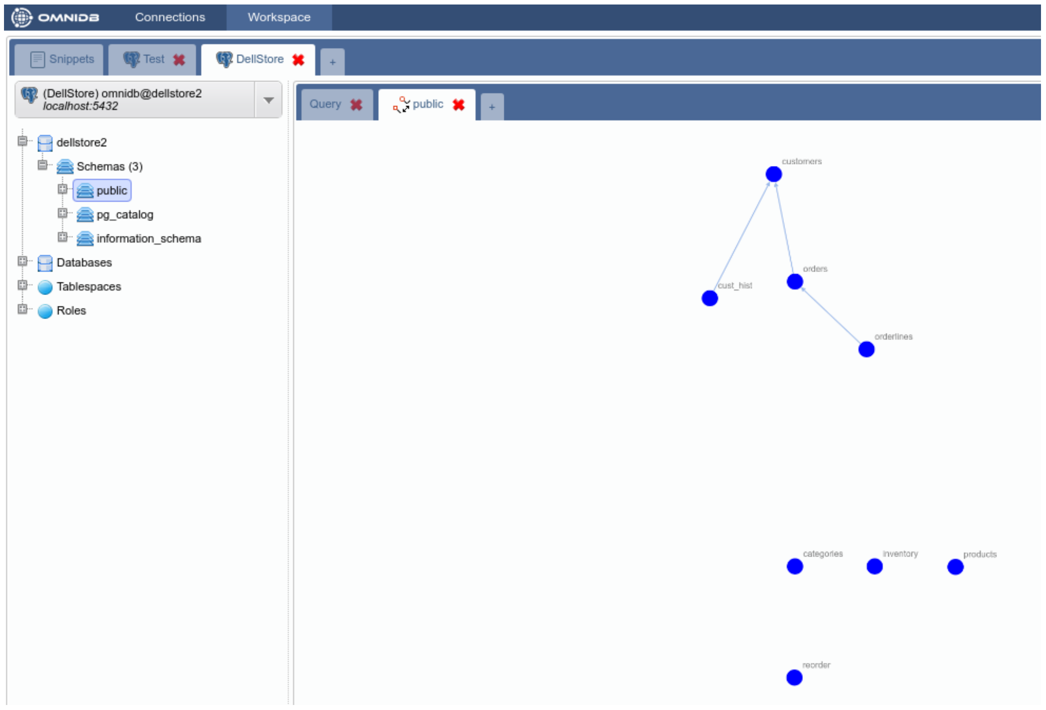
This feature displays a graph with nodes representing tables and edges representing table relationships with foreign keys. Using the mouse, the user is able to zoom in, zoom out, and drag and drop nodes to change its position.

There are two types of graphs: *Simple Graph* and *Complete Graph*.

9.1 Simple graph

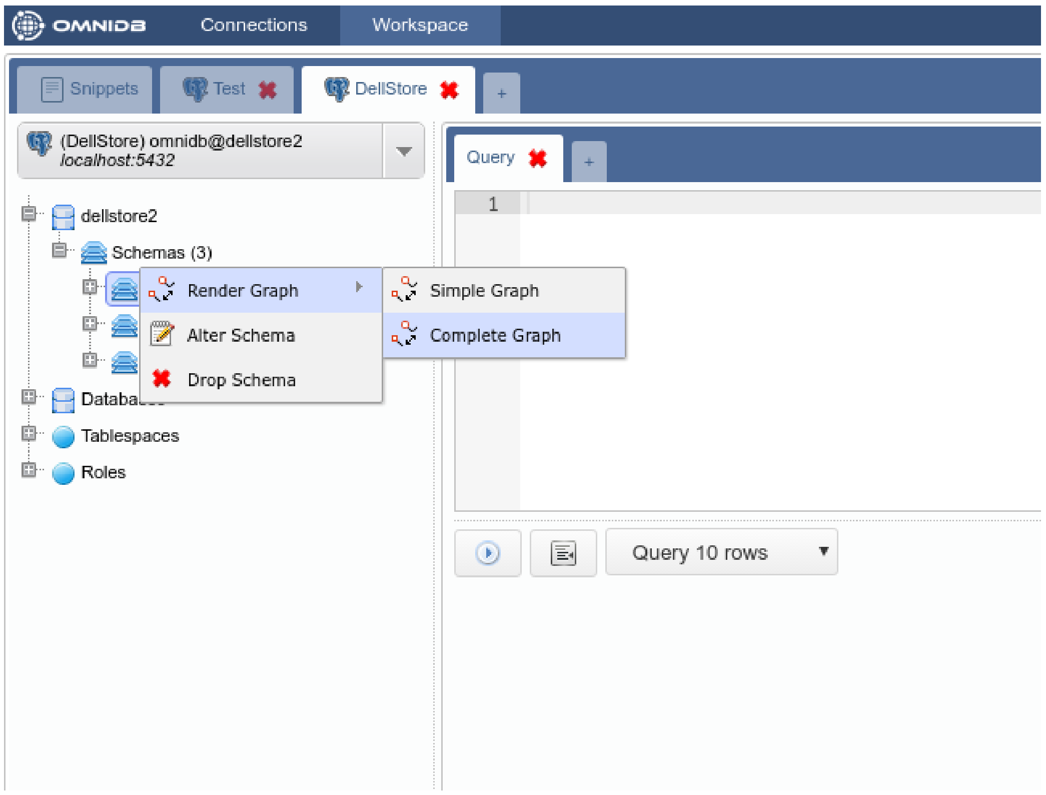
To access it just right click the root node of the tree and then select the action *Render Graph > Simple Graph*:

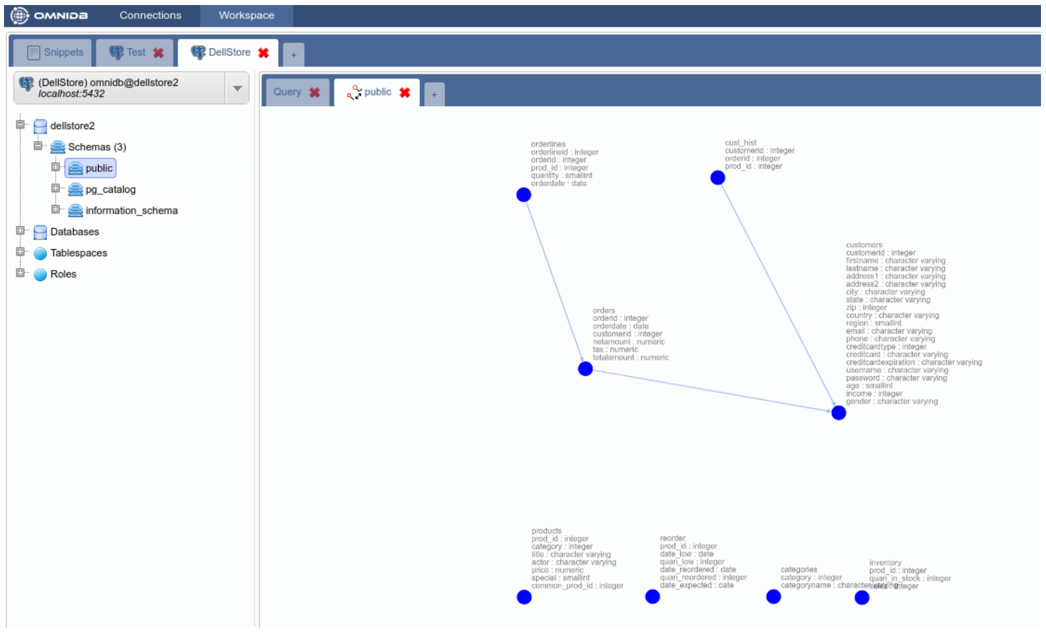




9.2 Complete graph

This graph displays tables with all its columns and respective data types. Additionally, edges now are labeled with information about the specific foreign key. To access it just right click the root node of the tree and then select the action *Render Graph > Complete Graph*:

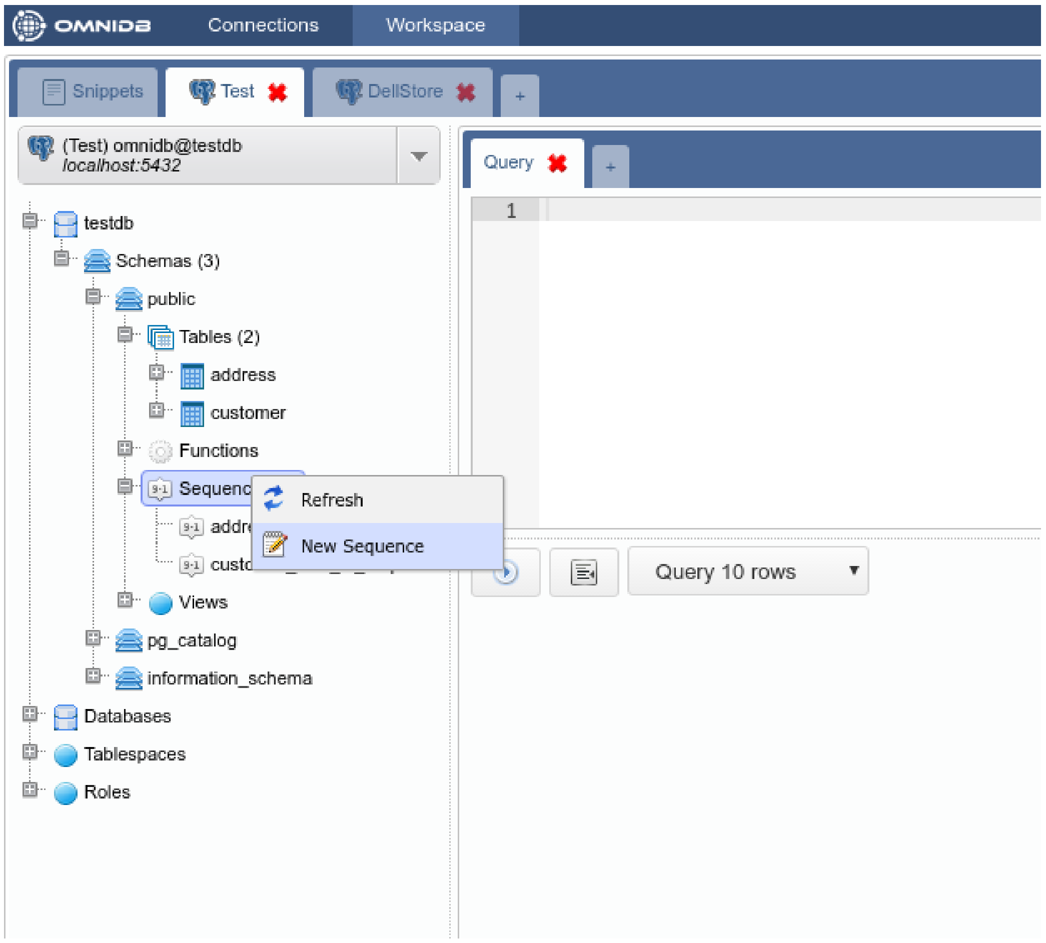

OMNIDB
Connections
Workspace
Snippets
Test
DellStore
(DellStore) omnidb@dellstore2
localhost:5432
Query
dellstore2
Schemas (3)
Render Graph
Alter Schema
Drop Schema
Simple Graph
Complete Graph
Tablespaces
Roles
Query 10 rows

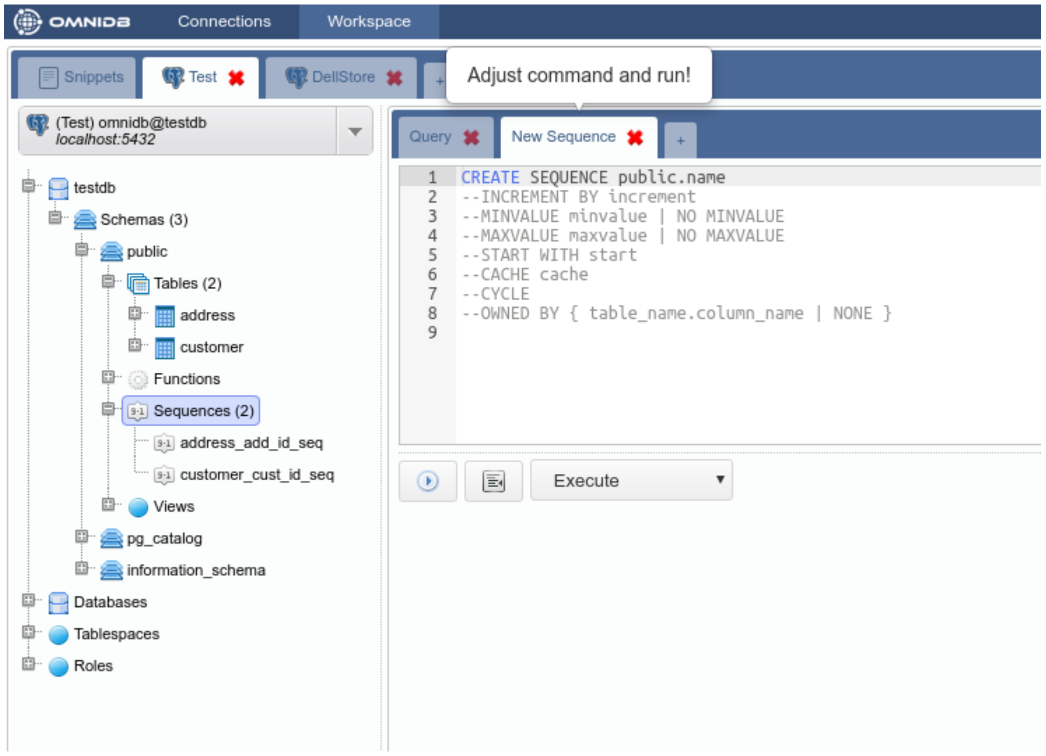

OMNIDB
Connections
Workspace
Snippets
Test
DellStore
(DellStore) omnidb@dellstore2
localhost:5432
Query
public
dellstore2
Schemas (3)
public
pg_catalog
information_schema
Databases
Tablespaces
Roles
orderlines
orderlineid : integer
orderid : integer
prod_id : integer
quantity : smallint
orderdate : date
cust_hist
customerid : integer
orderid : integer
prod_id : integer
customers
customerid : integer
firstname : character varying
lastname : character varying
address1 : character varying
address2 : character varying
city : character varying
state : character varying
zip : integer
country : character varying
region : smallint
email : character varying
phone : character varying
creditcardtype : integer
creditcard : character varying
creditcardexpiration : character varying
username : character varying
password : character varying
age : smallint
income : integer
gender : character varying
orders
orderid : integer
orderdate : date
customerid : integer
netamount : numeric
tax : numeric
totalamount : numeric
products
prod_id : integer
category : integer
title : character varying
actor : character varying
price : numeric
special : smallint
common_prod_id : integer
reorder
prod_id : integer
date_low : date
quan_low : integer
date_reordered : date
quan_reordered : integer
date_expected : date
categories
category : integer
categoryname : character varying
inventory
prod_id : integer
quan_in_stock : integer
sales : integer

10 SQL Templates

With the exception of tables, all PostgreSQL structures are possible to be managed with the use of *SQL templates*. This gives the user more power than using graphical forms to manipulate structures.

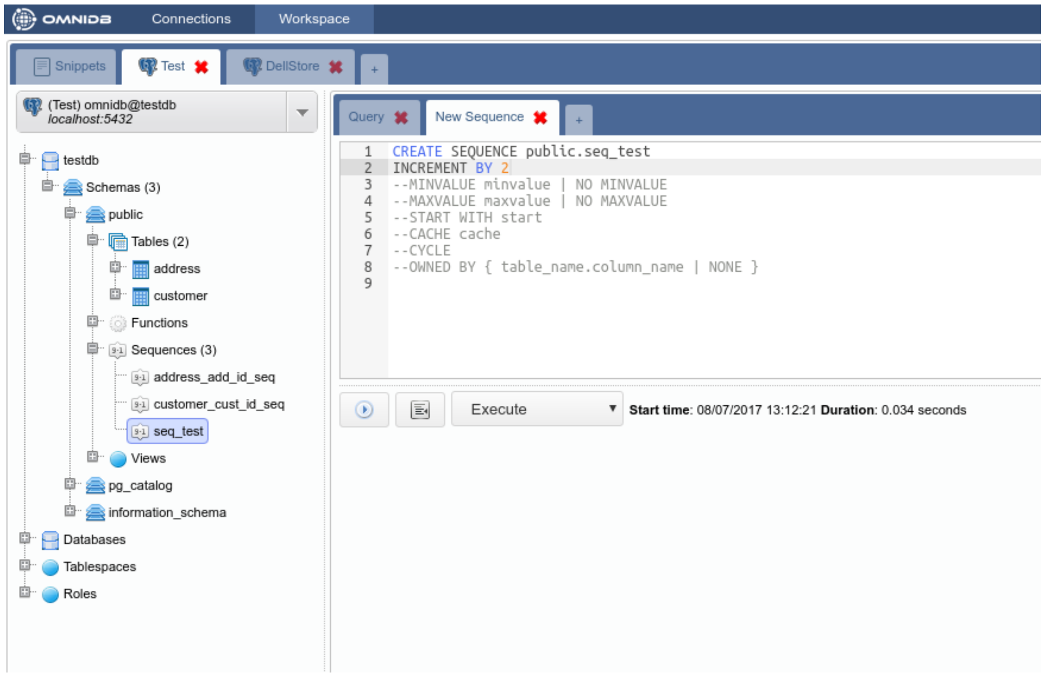
For example, let's consider the sequences inside the schema `public` of the database `testdb`. To create a new sequence, right click on the *Sequences* node, and choose *New Sequence*.





```
CREATE SEQUENCE public.name
--INCREMENT BY increment
--MINVALUE minvalue | NO MINVALUE
--MAXVALUE maxvalue | NO MAXVALUE
--START WITH start
--CACHE cache
--CYCLE
--OWNED BY { table_name.column_name | NONE }
```

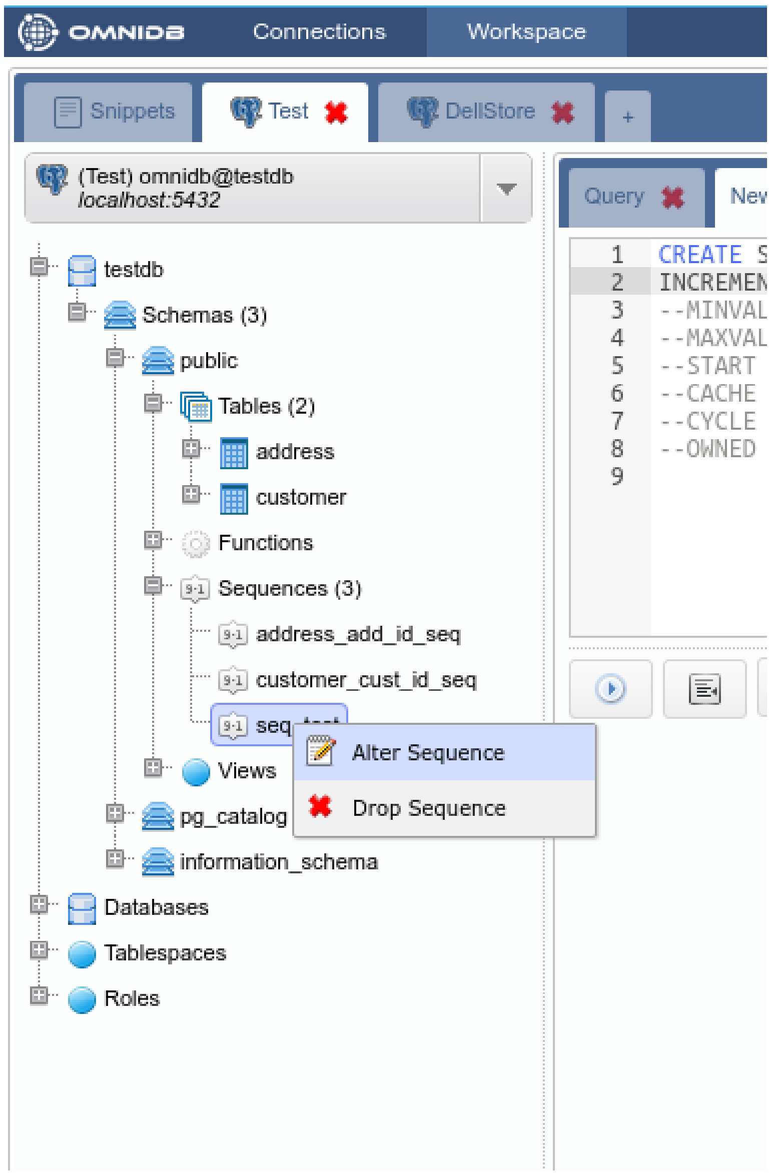
After you change the name of the sequence, you can uncomment other command options and set them accordingly to your needs. When all the command looks fine, you can click on the *Execute button* and the new sequence will be created:



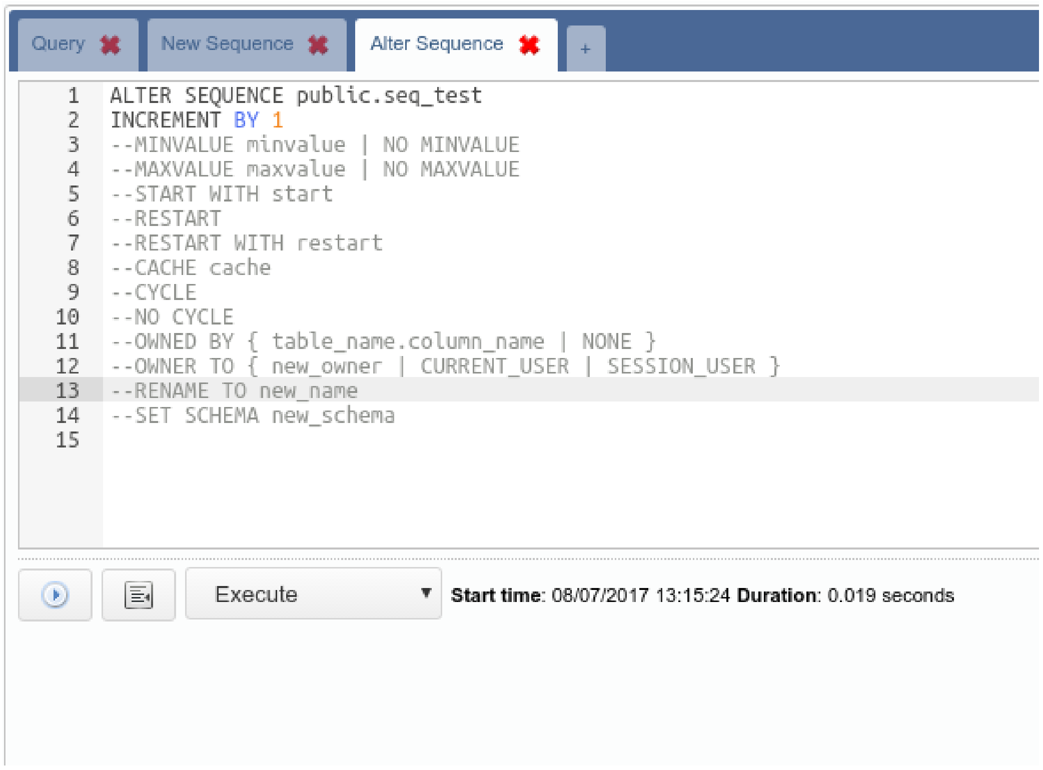
```
CREATE SEQUENCE public.seq_test
INCREMENT BY 2
--MINVALUE minvalue | NO MINVALUE
--MAXVALUE maxvalue | NO MAXVALUE
--START WITH start
--CACHE cache
--CYCLE
--OWNED BY { table_name.column_name | NONE }

```

Start time: 08/07/2017 13:12:21 **Duration**: 0.034 seconds

With right click on an existing sequence, you can alter or drop it. It will work the same way as the creation, by using a SQL template for the user to change.

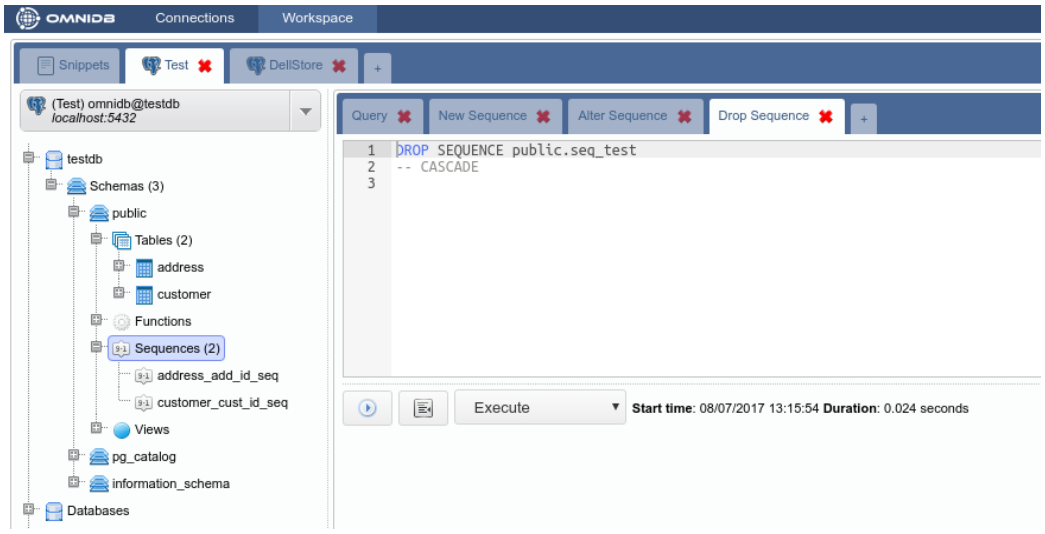


```
CREATE S
INCREMEN
--MINVAL
--MAXVAL
--START
--CACHE
--CYCLE
--OWNED
```



```
ALTER SEQUENCE public.seq_test
INCREMENT BY 1
--MINVALUE minvalue | NO MINVALUE
--MAXVALUE maxvalue | NO MAXVALUE
--START WITH start
--RESTART
--RESTART WITH restart
--CACHE cache
--CYCLE
--NO CYCLE
--OWNED BY { table_name.column_name | NONE }
--OWNER TO { new_owner | CURRENT_USER | SESSION_USER }
--RENAME TO new_name
--SET SCHEMA new_schema

```

Start time: 08/07/2017 13:15:24 **Duration**: 0.019 seconds



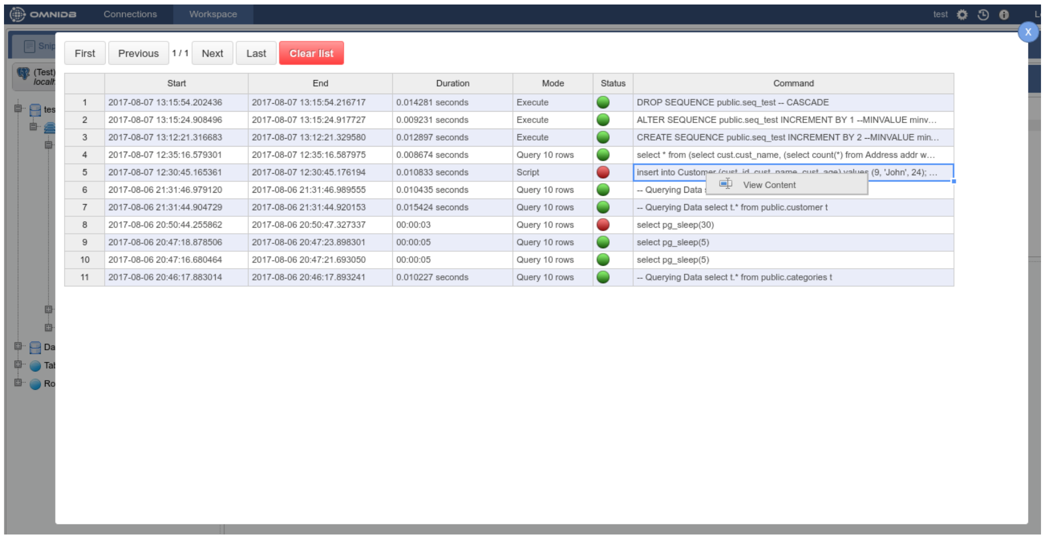
```
DROP SEQUENCE public.seq_test
-- CASCADE

```

Start time: 08/07/2017 13:15:54 **Duration**: 0.024 seconds

11 Additional Features

11.1 SQL History

Every interaction the user does with every database is logged in OmniDB's *SQL History*. To access it you need to click on the clock-like icon on the upper right corner. OmniDB will show a pop-up with all actions in a paginated grid.



	Start	End	Duration	Mode	Status	Command
1	2017-08-07 13:15:54.202436	2017-08-07 13:15:54.216717	0.014281 seconds	Execute	🟢	DROP SEQUENCE public.seq_test -- CASCADE
2	2017-08-07 13:15:24.908496	2017-08-07 13:15:24.917727	0.009231 seconds	Execute	🟢	ALTER SEQUENCE public.seq_test INCREMENT BY 1 --MINVALUE minv...
3	2017-08-07 13:12:21.316683	2017-08-07 13:12:21.329580	0.012897 seconds	Execute	🟢	CREATE SEQUENCE public.seq_test INCREMENT BY 2 --MINVALUE min...
4	2017-08-07 12:35:16.579301	2017-08-07 12:35:16.587975	0.008674 seconds	Query 10 rows	🟢	select * from (select cust.cust_name, (select count(*) from Address addr w...
5	2017-08-07 12:30:45.165361	2017-08-07 12:30:45.176194	0.010833 seconds	Script	🔴	insert into Customer (cust_id, cust_name, cust_age) values (9, 'John', 24); ...
6	2017-08-06 21:31:46.979120	2017-08-06 21:31:46.989555	0.010435 seconds	Query 10 rows	🟢	-- Querying Data s
7	2017-08-06 21:31:44.904729	2017-08-06 21:31:44.920153	0.015424 seconds	Query 10 rows	🟢	-- Querying Data select t.* from public.customer t
8	2017-08-06 20:50:44.255862	2017-08-06 20:50:47.327337	00:00:03	Query 10 rows	🔴	select pg_sleep(30)
9	2017-08-06 20:47:18.878506	2017-08-06 20:47:23.898301	00:00:05	Query 10 rows	🟢	select pg_sleep(5)
10	2017-08-06 20:47:16.680464	2017-08-06 20:47:21.693050	00:00:05	Query 10 rows	🟢	select pg_sleep(5)
11	2017-08-06 20:46:17.883014	2017-08-06 20:46:17.893241	0.010227 seconds	Query 10 rows	🟢	-- Querying Data select t.* from public.categories t

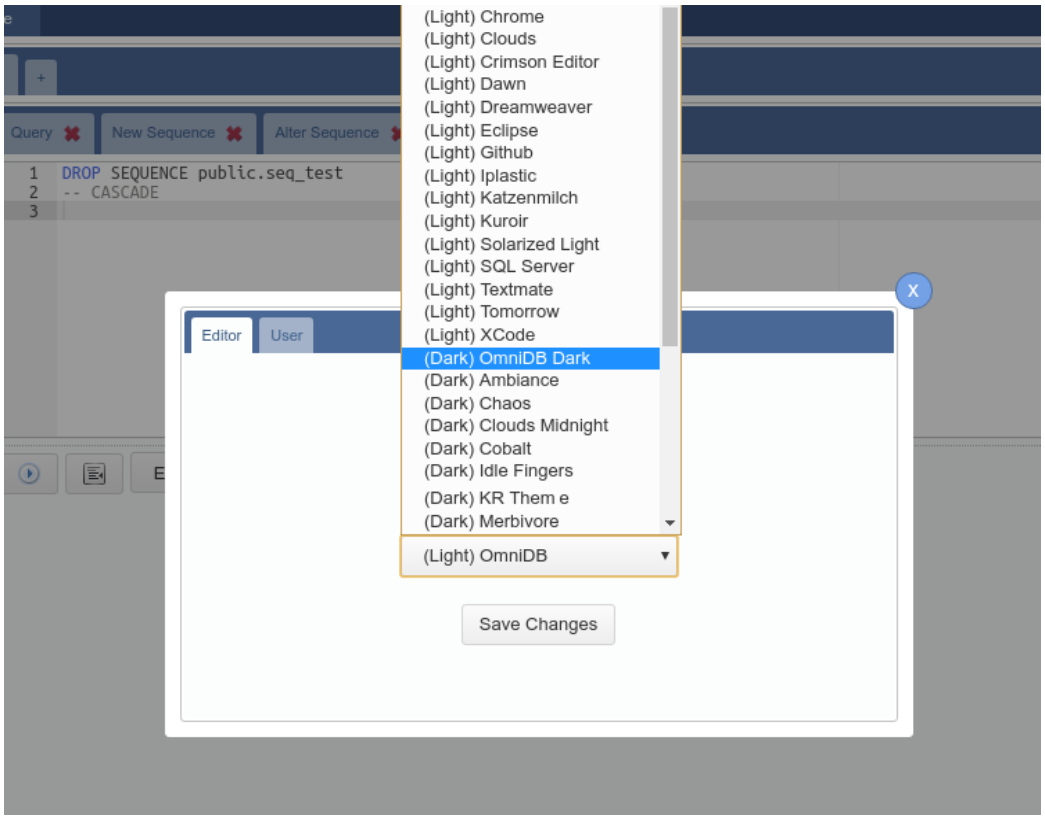
Each action shows date time it started, the time it ended, the duration, the mode, the status and the command. As every grid in OmniDB, you can right click on the command and click *View Content*, where another pop-up will open showing the content in a larger text editor.

11.2 User Settings

Also in the upper right corner, by clicking in the gear-like icon, OmniDB will open the *User Settings* pop-up. It is composed by two tabs:

- **User**: Allows the user to change its password. More user settings will be added in future.
- **Editor**: Allows the user to change the font size of the SQL Editor, and

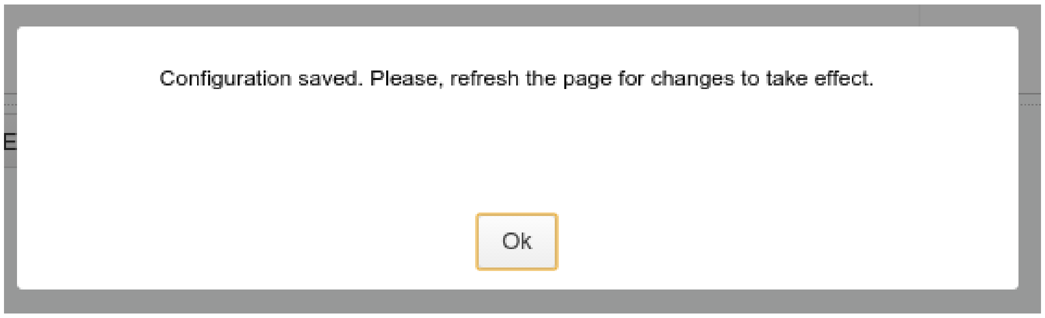
also change the entire OmniDB theme. There are a lot of OmniDB themes, each of them change the syntax highlight color of the editor. They are also categorized in light and dark themes. A light theme is the default; a dark theme will change the entire interface of OmniDB.

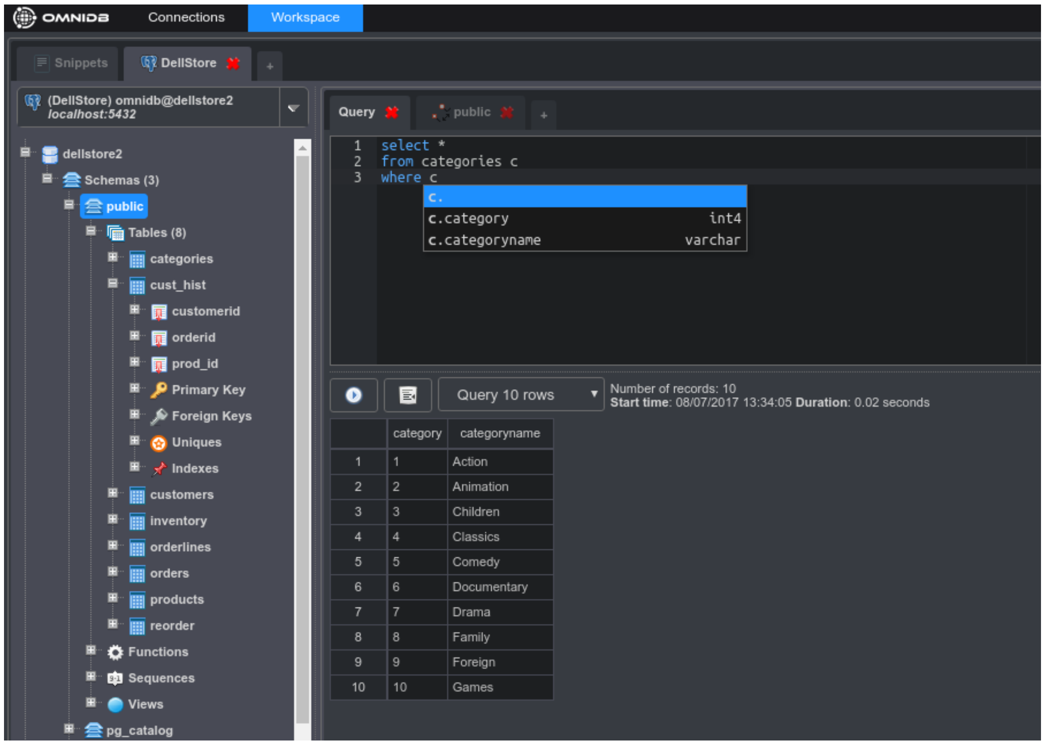


```
DROP SEQUENCE public.seq_test
-- CASCADE
```

Every change in user settings require that you either:

- Refresh the page, if you are using OmniDB server and the interface through a web browser; or
- Open and close OmniDB, if you are using OmniDB app.


Configuration saved. Please, refresh the page for changes to take effect.



```
select *
from categories c
where c
```

Number of records: 10
Start time: 08/07/2017 13:34:05 **Duration**: 0.02 seconds

	category	categoryname
1	1	Action
2	2	Animation
3	3	Children
4	4	Classics
5	5	Comedy
6	6	Documentary
7	7	Drama
8	8	Family
9	9	Foreign
10	10	Games

12 OmniDB Config Tool

Every installation of OmniDB also comes with a small CLI utility called *OmniDB Config*. It will have a different file name, depending on the way you installed OmniDB:

- If you are using a tarball or zip package, it is called **omnidb-config**, for both server and app versions;

- If you used an installer (like the .deb file) of server version, it is called **omnidb-config-server**;
- If you used an installer of app version, it is called **omnidb-config-app**.

Despite having different names, the utility does exactly the same. If you used an installer, it will be put in your `$PATH`.

```
user@machine:~$  omnidb-config-app --help
Usage: omnidb-config-app [options]

Options:
  --version             show program\'s version number and exit
  -h, --help            show this help message and exit
  -c username password, --createsuperuser=username password
                        create super user: -c username password
  -a, --vacuum          databases maintenance
  -r, --resetdatabase   reset databases
```

12.1 Create super user

Option `-c` allows you to create a new super user, without needing to open OmniDB interface.

```
user@machine:~$ omnidb-config-app -c william password
Creating superuser...
Superuser created.
```

12.2 Vacuum

OmniDB has two databases:

- `omnidb.db`: Stores all users and connections, and other OmniDB related stuff;
- *Sessions database*: Stores Django user sessions.

Both databases are SQLite, so it can be useful to vacuum them sometimes to reduce file size. This can be done with the `-a` option.

```
user@machine:~$ omnidb-config-app -a
Vacuuming OmniDB database...
Done.
Vacuuming Sessions database...
Done.
```

12.3 Reset database

If you wish to wipe out all OmniDB information and get a clean database as it was just installed, you can use the `-r` option. Use it with caution!

```
user@machine:~$ omnidb-config-app -r
*** ATENTION *** ALL USERS DATA WILL BE LOST
Would you like to continue? (y/n) y
Cleaning users...
Done.
Cleaning sessions...
Vacuuming OmniDB database...
Done.
Vacuuming Sessions database...
Done.
```