

- [1 Introduction](#)
 - [1.1 History](#)
- [2 Installation](#)
 - [2.1 OmniDB Application](#)
 - [2.2 OmniDB Server](#)
 - [2.2.1 OmniDB User Database](#)
 - [2.2.2 OmniDB configuration file](#)
 - [2.2.3 OmniDB in the browser](#)
- [3 Creating Users and Connections](#)
 - [3.1 Logging in as user *admin*](#)
 - [3.2 Creating another user](#)
 - [3.3 Signing in as the new user](#)
 - [3.4 Creating connections](#)
- [4 Managing Databases](#)
 - [4.1 Sections of the *Workspace* window](#)
 - [4.2 Connection Outer Tab](#)
 - [4.3 Working with databases](#)
 - [4.4 Working with multiple tabs inside the same connection](#)
- [5 Creating, Changing and Removing Tables](#)
 - [5.1 Creating tables](#)
 - [5.2 Editing tables](#)
 - [5.3 Removing tables](#)
- [6 Managing Table Data](#)
- [7 Writing SQL Queries](#)
- [8 Visualizing Query Plans](#)
 - [8.1 Textual visualization](#)
 - [8.2 Tree visualization](#)
- [9 Visualizing Data](#)
 - [9.1 Simple graph](#)
 - [9.2 Complete graph](#)
- [10 Managing other PostgreSQL Elements](#)
- [11 Additional Features](#)
 - [11.1 SQL History](#)
 - [11.2 User Settings](#)
 - [11.3 Contextual Help](#)
- [12 OmniDB Config Tool](#)
 - [12.1 Create super user](#)
 - [12.2 Vacuum](#)
 - [12.3 Reset database](#)

1 Introduction

OmniDB is an open source browser-based app designed to access and manage many different Database Management systems, e.g. PostgreSQL, Oracle and MySQL. OmniDB can run either as an App or via Browser, combining the flexibility needed for various access paths with a design that puts security first. OmniDB is actively developed, automatically tested on a variety of databases and browsers and comes with full documentation.

Since early development, OmniDB was designed as an browser-based 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 browser-based 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.

1.1 History

OmniDB's creators, Rafael Thofehn 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.

2 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 browser 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.

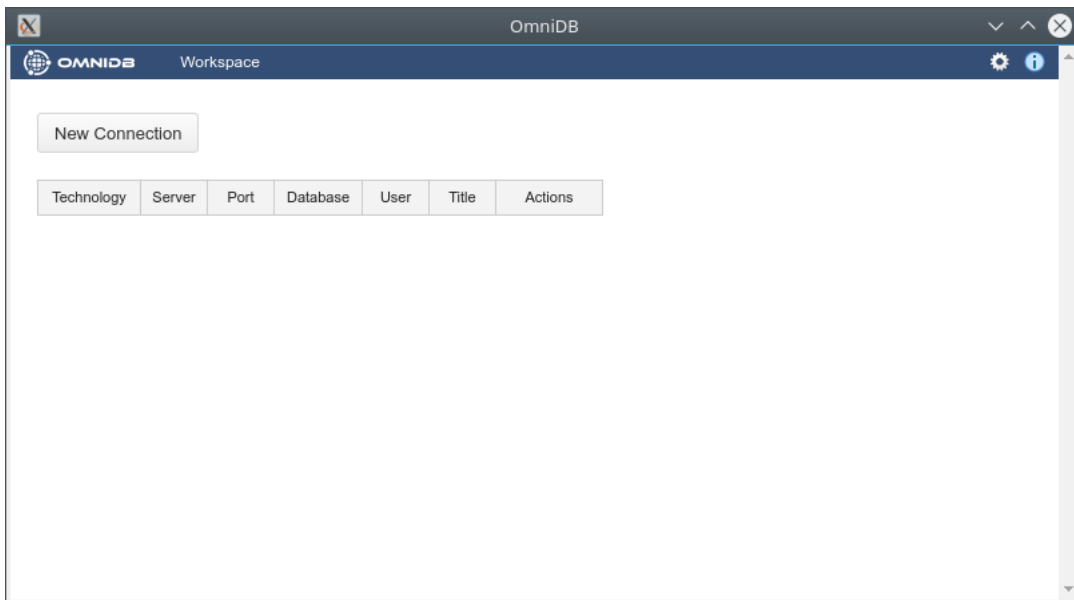
Both application and server can be installed on the same machine.

2.1 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
- Windows 32 bits / 64 bits
 - EXE installer
- Mac OSX
 - DMG installer

Use the specific installer for your Operational System and it will be available through your desktop environment application menu or via command line with `omnidb-app`.



2.2 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.

Use the specific installer for your Operational System and it will be available through command line with `omnidb-server`.

You will need administrator privileges:

```
user@machine:~$ sudo omnidb-server
Starting OmniDB websocket...
Checking port availability...
Starting websocket server at port 25482.
OmniDB successfully migrated user database from version 0.0.0 to version 2.4.0
Starting OmniDB server...
Checking port availability...
Starting server OmniDB 2.4.0 at 0.0.0.0:8000.
Open OmniDB in your favorite browser
Press Ctrl+C to exit
```

Note how OmniDB starts a *websocket server* in a random port (in the above example it started in port 25482) and a *web server* in port 8000. You can also specify websocket server port, web server port and listening address:

```
user@machine:~$ sudo omnidb-server -p 8080 -w 2000 -H 127.0.0.1
Starting OmniDB websocket...
Checking port availability...
Starting websocket server at port 2000.
OmniDB successfully migrated user database from version 0.0.0 to version 2.4.0
Starting OmniDB server...
Checking port availability...
Starting server OmniDB 2.4.0 at 127.0.0.1:8080.
Open OmniDB in your favorite browser
```

2.2.1 OmniDB User Database

Since version 2.4.0, upon initialization both server and app will create a file `~/ .omnidb/omnidb-app/omnidb.db` (for OmniDB app) or `~/ .omnidb/omnidb-server/omnidb.db` (for OmniDB server) in the user home directory, if it does not exist. That can be confirmed by the message *OmniDB successfully migrated user database from version 0.0.0 to version 2.4.0* you saw above. This file is also called **user database** and contains user data. If it already exists, then OmniDB will check whether the version of the server matches the version of the user database:

```
user@machine:~$ sudo omnidb-server
Starting OmniDB websocket...
Checking port availability...
Starting websocket server at port 25482.
User database version 2.4.0 is already matching server version.
Starting OmniDB server...
Checking port availability...
Starting server OmniDB 2.4.0 at 0.0.0.0:8000.
Open OmniDB in your favorite browser
Press Ctrl+C to exit
```

Future releases of OmniDB will contain the **user database migration** SQL commands required to upgrade the user database, if necessary. This way user data is not lost by upgrading OmniDB. Imagine the following scenario: you use OmniDB 2.4.0 now and you decide to upgrade it to newest release 2.5.0, for example. After the upgrade, when you start OmniDB server, it will apply the changes version 2.5.0 requires. So you will see something like that:

```
user@machine:~$ sudo omnidb-server
Starting OmniDB websocket...
Checking port availability...
Starting websocket server at port 25482.
OmniDB successfully migrated user database from version 2.4.0 to version 2.5.0
Starting OmniDB server...
Checking port availability...
Starting server OmniDB 2.5.0 at 0.0.0.0:8000.
Open OmniDB in your favorite browser
Press Ctrl+C to exit
```

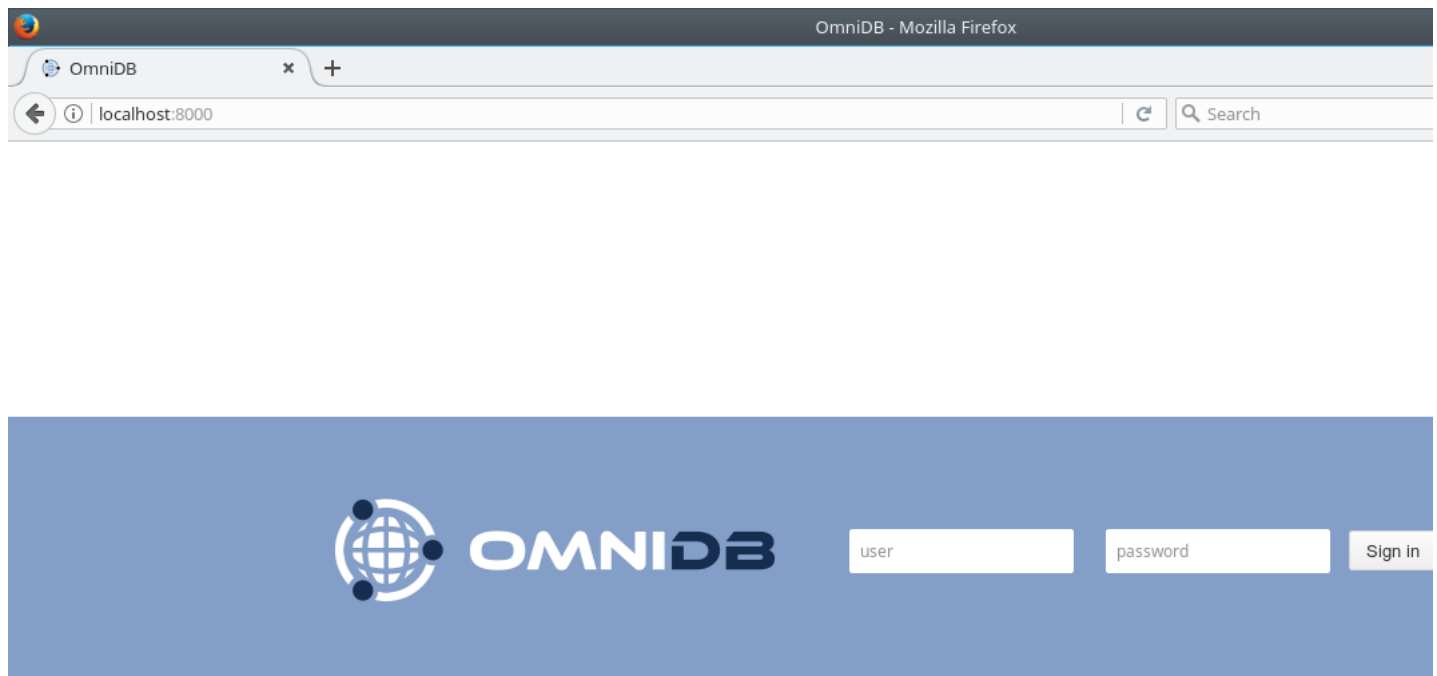
2.2.2 OmniDB configuration file

Starting on version 2.1.0, OmniDB server comes with a configuration file `omnidb.conf` that enables the user to specify parameters such as port and listening address. Also, 2.1.0 enables us to start the server with SSL, this requires a certificate and is configured in the same configuration file.

Starting on version 2.4.0, this file is located in `/ .omnidb/omnidb-app/omnidb.db` (for OmniDB app) or `~/ .omnidb/omnidb-server/omnidb.db` (for OmniDB server) in the user home directory.

2.2.3 OmniDB in the browser

Now that the web server is running, you may access OmniDB browser-based 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:

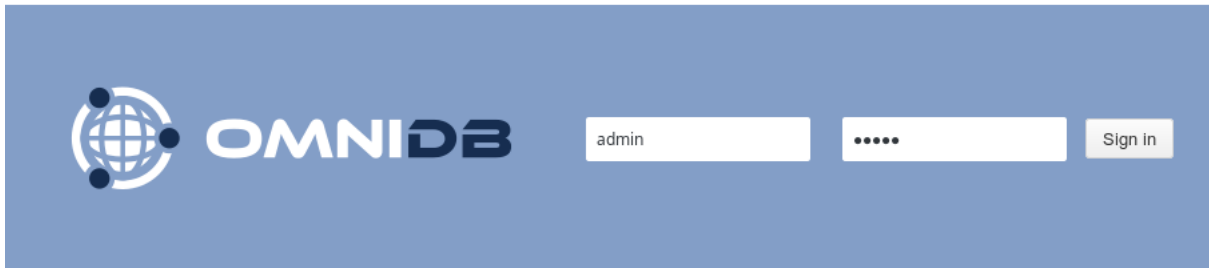


Now you know that OmniDB is running correctly. In the next chapters, we will see how to login for the first time, how to create an user and to utilize OmniDB.

3 Creating Users and Connections

3.1 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.

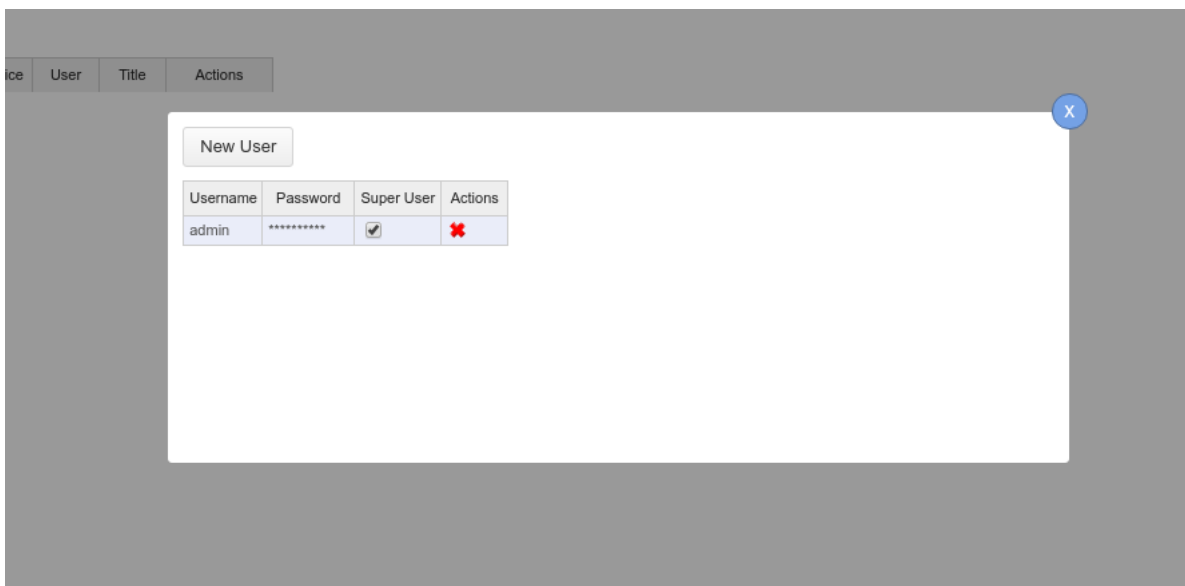


The next window is the **Connections** window. We will talk about it later.



3.2 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.



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

New User

Username	Password	Super User	Actions
admin	*****	<input checked="" type="checkbox"/>	✖
user2	*****	<input type="checkbox"/>	✖

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 seen by super users. When you are done, click on the *Save Data* button inside the popup.

New User Save Data

Username	Password	Super User	Actions
admin	*****	<input checked="" type="checkbox"/>	✖
test	****	<input type="checkbox"/>	✖

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.

3.3 Signing in as the new user

Let us sign in as the user we just created.



OMNIDB

test

....

Sign In

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.



Workspace

New Connection

Technology	Server	Port	Database	User	Title	Actions
------------	--------	------	----------	------	-------	---------







3.4 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.

OMNiDB Workspace

New Connection


Technology	Server	Port	Database	User	Title	Actions
postgresql	127.0.0.1	5432	testdb	omnidb	Test	  
postgresql	127.0.0.1	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.

OMNiDB Workspace

New Connection

Technology	Server	Port	Database	User	Title	Actions
postgresql	127.0.0.1	5432	testdb	omnidb		
postgresql	127.0.0.1	5432	dellstore2	omnidb		


fe_sendauth: no password supplied







Ok Cancel

Notice a pop-up appears with the message *fe_sendauth: no password supplied*. This is happening because OmniDB does not store the database user password on disk. Not having any password at hand, OmniDB will try to connect without one, thus trying to take advantage of automatic authentication methods that might be in place: trust method, .pgpass file, and so on. As the database server replies with an error not allowing the user to connect, then OmniDB understands a password is required and asks it to the user. 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.

OMNiDB Workspace

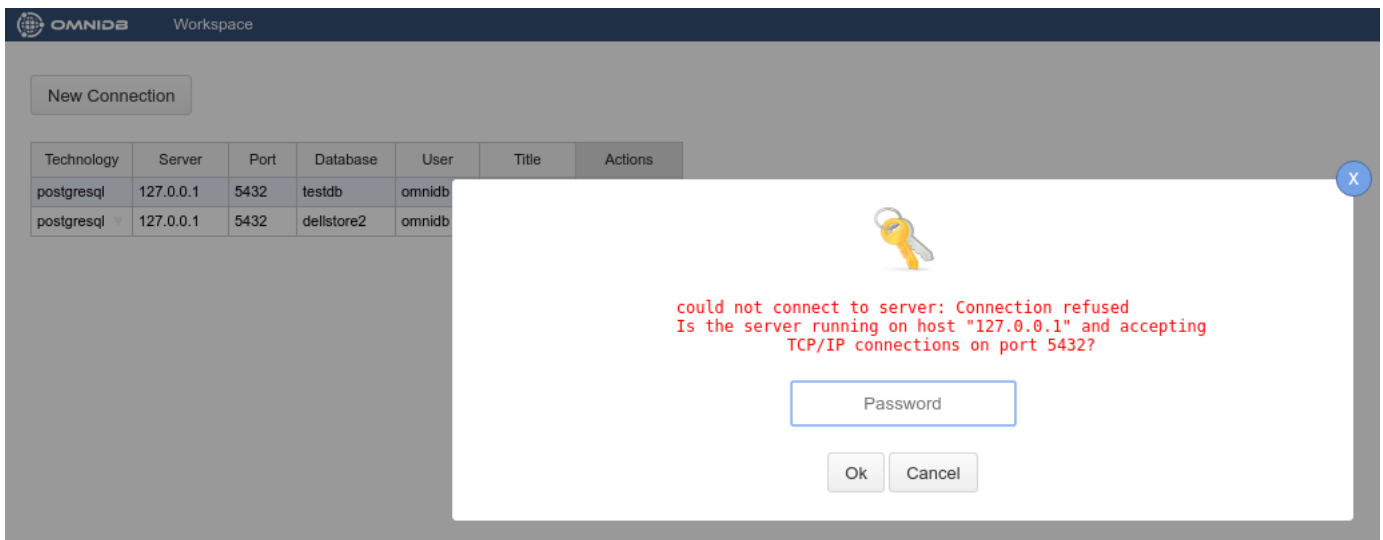
New Connection

Technology	Server	Port	Database	User	Title	Actions
postgresql	127.0.0.1	5432	testdb	omnidb	Test	  
postgresql	127.0.0.1	5432	dellstore2	omnidb	DellStore	  

Connection successful.

Ok

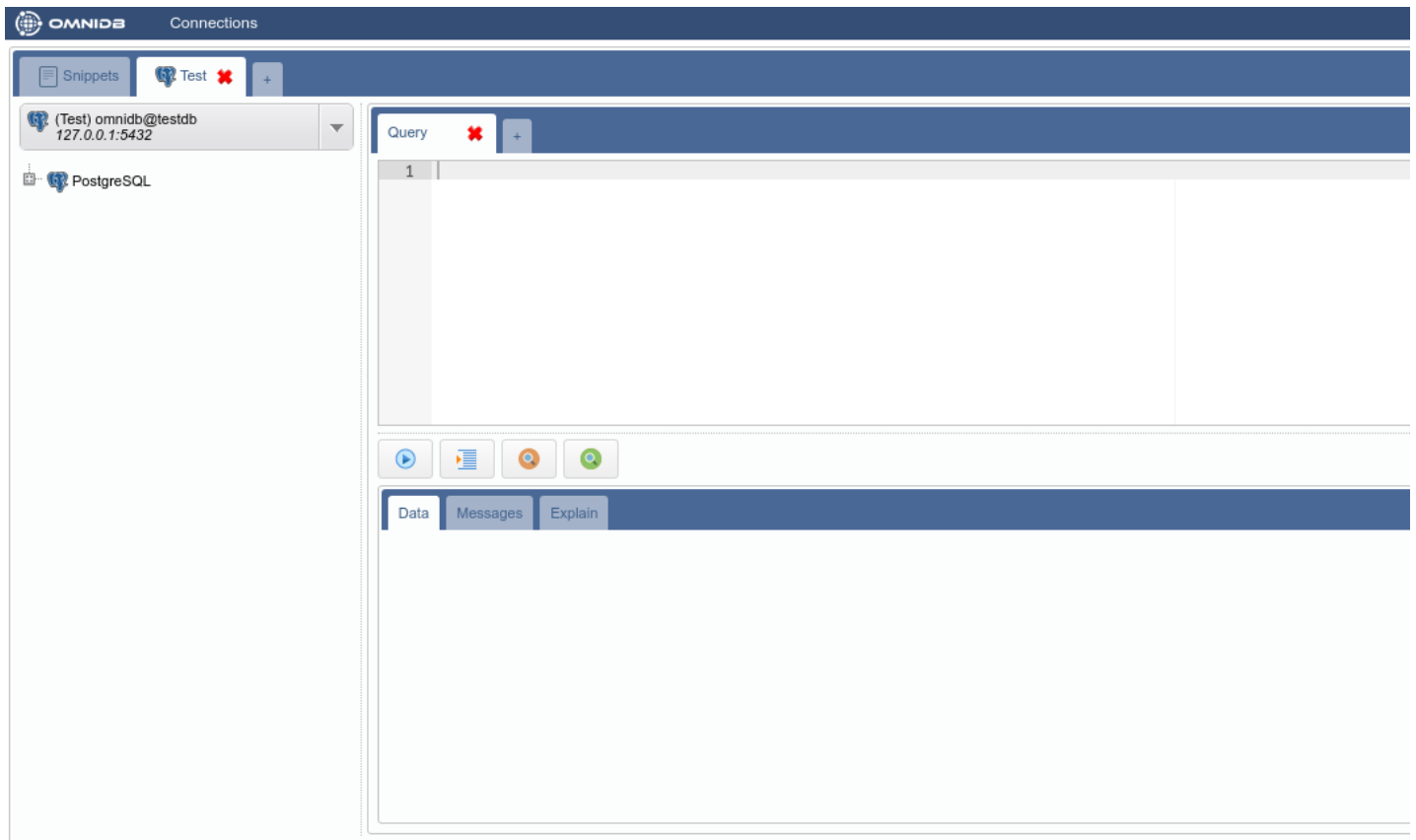
But, if you have trouble of any kind connecting to your PostgreSQL database, the same popup will remain showing the error OmniDB got.



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

4 Managing Databases

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.



4.1 Sections of the *Workspace* window

This interface has several elements:

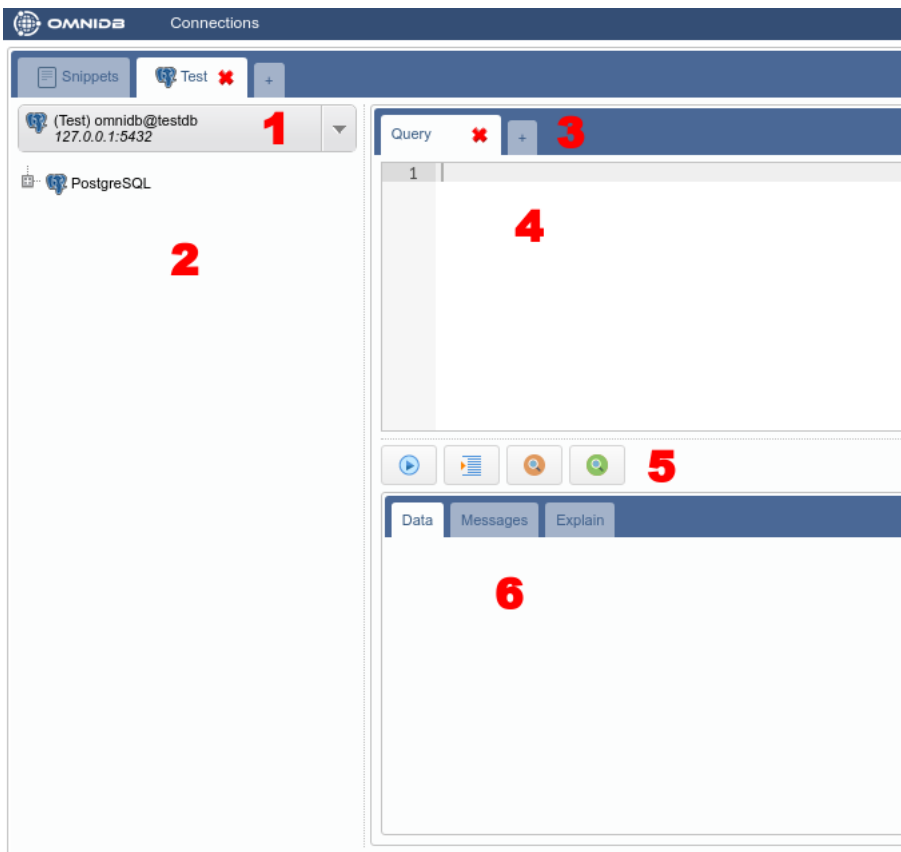


- **1) Links:** Enables the user to navigate to the *Connections* window and back to the *Workspace* window
- **2) Outer Tabs:** OmniDB lets you work with several databases at the same time. Each database will be accessible through an *outer tab*. Outer tabs also can host miscellaneous connection-independent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

4.2 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.

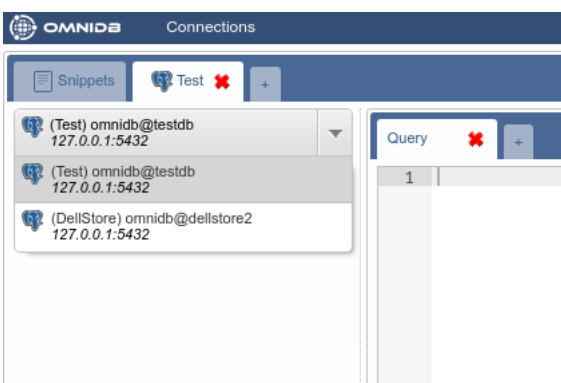
A new *Connection Outer Tab* will always automatically point to the first connection on your list of database connections. Or, if you clicked on the *Select Connection* action, it will point to the selected connection. Observe the elements inside of this tab:



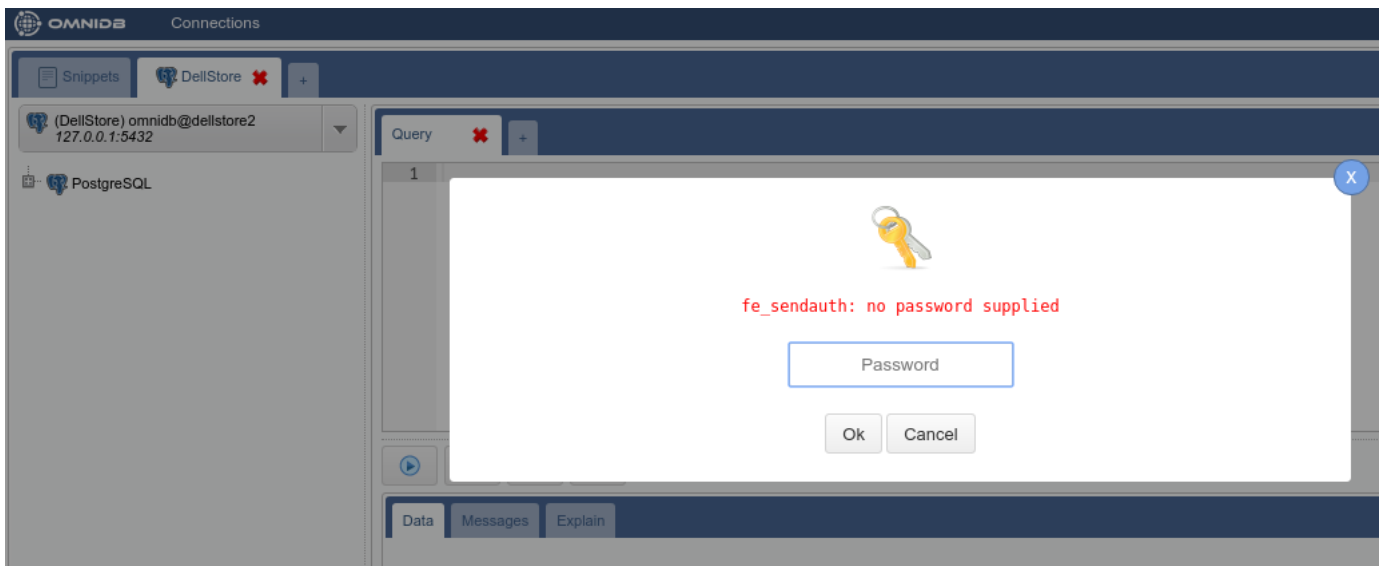
- **1) Connection Selector:** Shows all connections and lets the user select the current one
- **2) Tree of Structures:** Displays a hierarchical tree where you can navigate through the database elements
- **3) Inner Tabs:** Allows the user to execute actions in the current database. There are several 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*, *Format*, *Explain* and *Explain Analyze*
- **6) Inner Tab Results:** A *Query Tab*, after you click in the *Execute Button* or type the execute shortcut (**Alt-Q**), will show a grid with the query results in the *Data* subtab. If the query calls a function that raises messages, those will be shown in the *Messages* subtab. If instead of *Execute* you clicked in *Explain* or *Explain Analyze*, the explain plan for the query will be shown in the *Explain* subtab.

4.3 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.

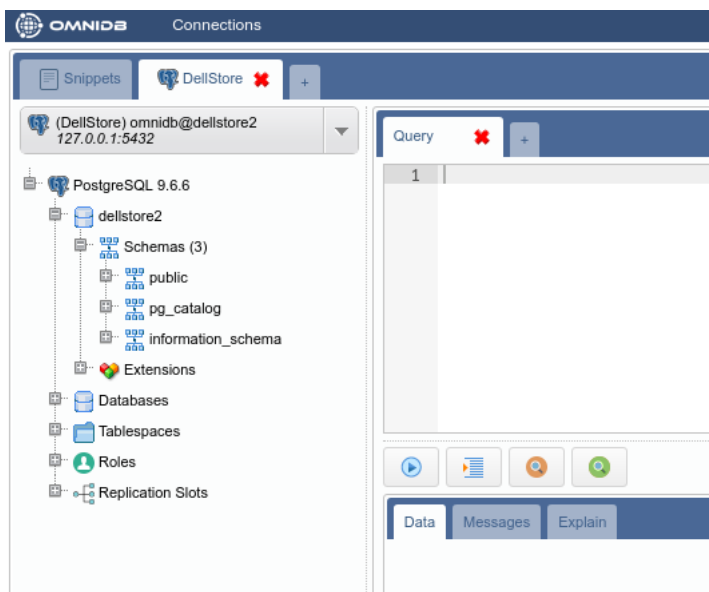


Select the *DellStore* connection. Now go to the tree right below the selector and click to expand the root node *PostgreSQL*.

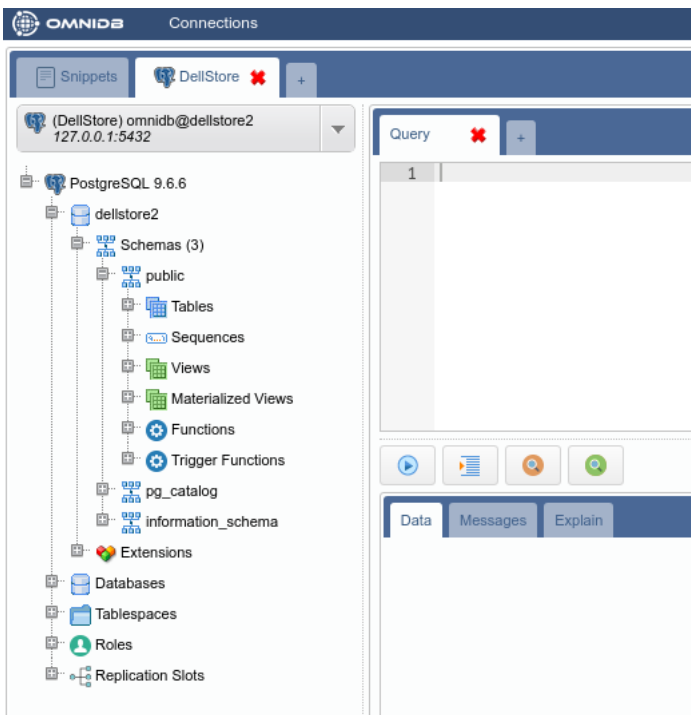


Bear in mind that every 30 minutes you keep without performing actions on the database, will trigger a *Authentication* popup, meaning that the password that OmniDB has encrypted and stored in memory is now expired. As explained before, this is important for your database security. After you type the correct password, you will see the PostgreSQL node now shows the PostgreSQL version and also was expanded, showing the current database connection and also instance wide elements: *Databases*, *Tablespaces*, *Roles* and *Replication Slots*.

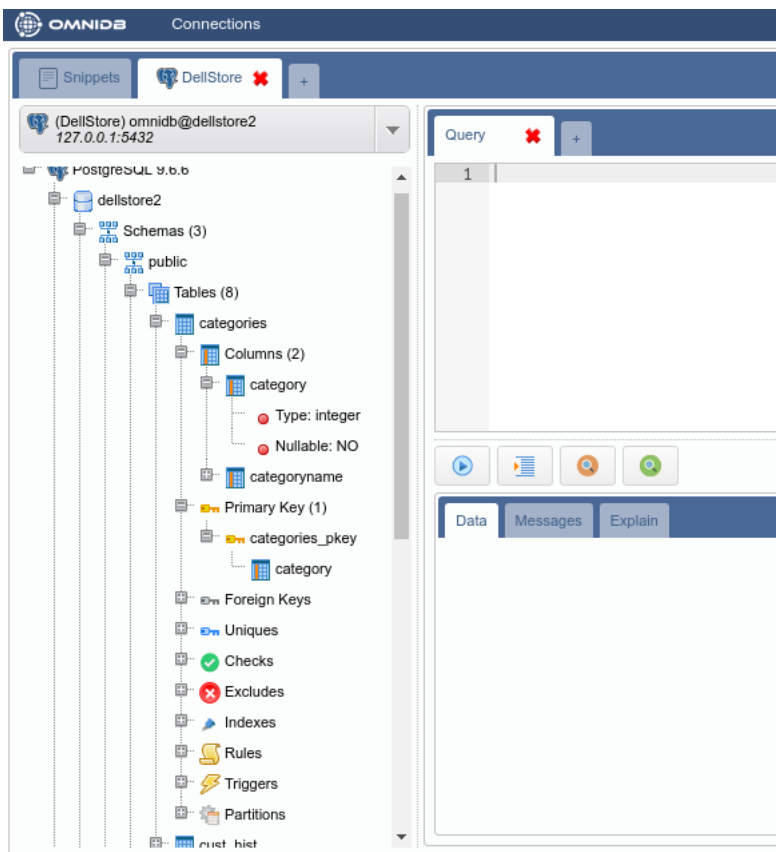
Go ahead and expand the *Schemas* node. You will see all schemas in the current database (in case of PostgreSQL, TOAST and temp schemas are not shown).



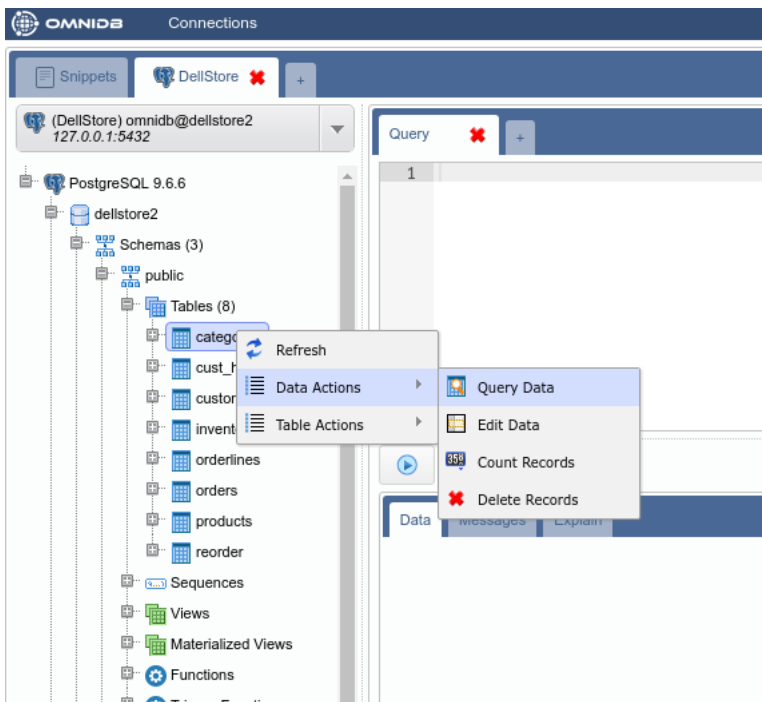
Now click to expand the schema *public*. You will see different kinds of elements contained in this schema.



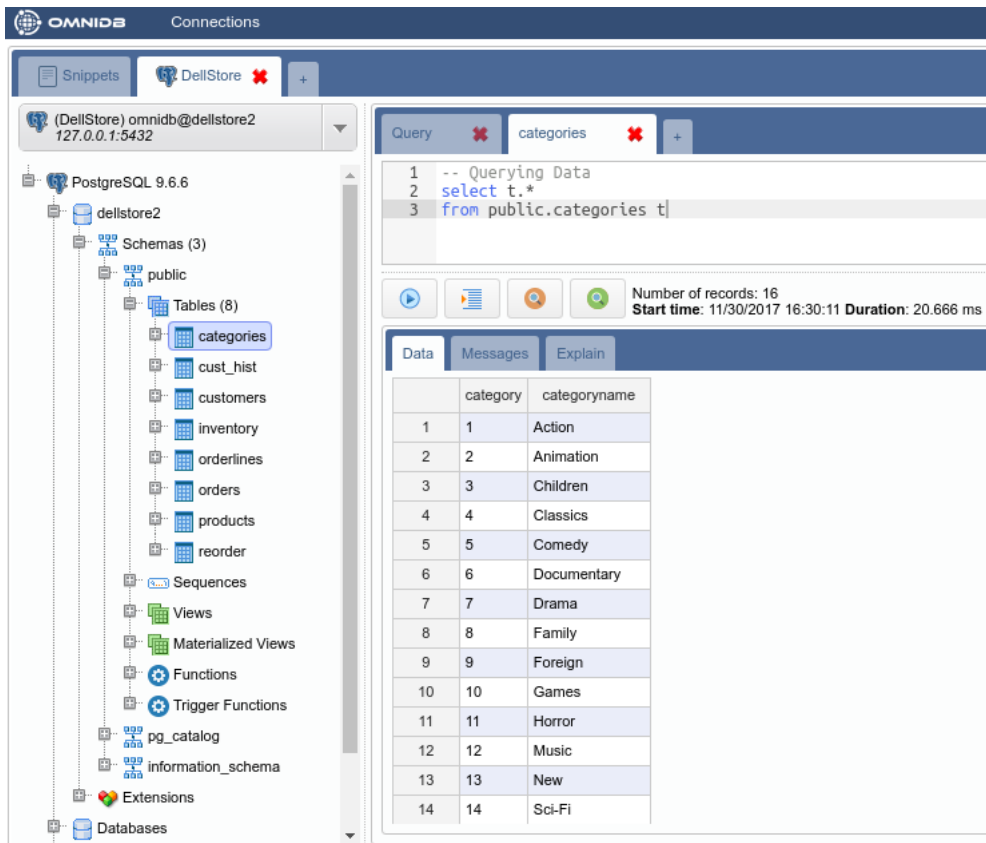
Now click to expand the node *Tables*, and you will see all tables contained in the schema *public*. Expand any table and you will see its columns, primary key, foreign keys, constraints, indexes, rules, triggers and partitions.



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



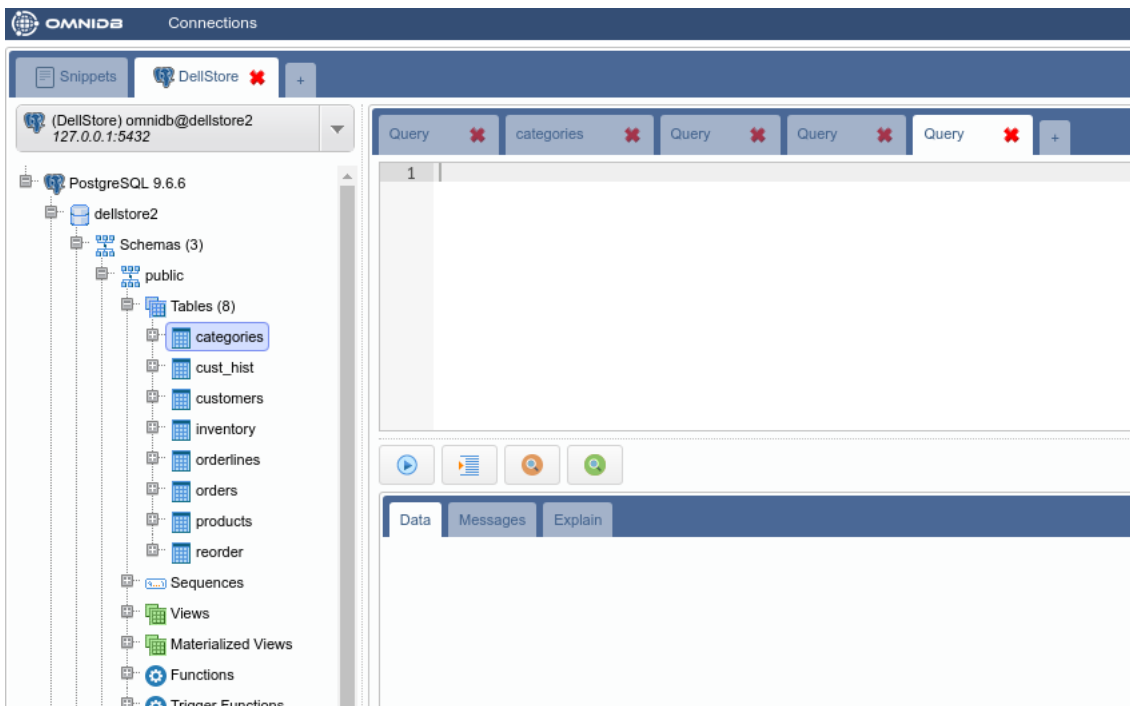
Notice that OmniDB opens a new SQL editor with a simple query to list table records. The records are displayed in a grid right below the editor. This grid can be controlled with keyboard as if you were using a spreadsheet manager. You can also copy data from single cells or block of cells (that can be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



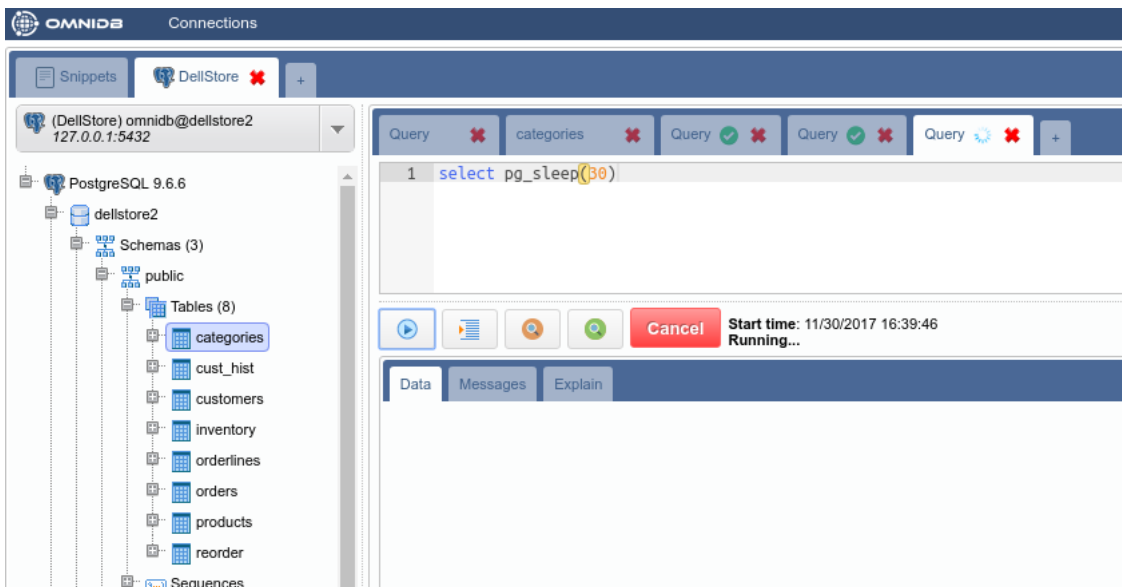
You can edit the query on the SQL editor, writing simple or more complex queries. To execute, click on the action button or hit the keystroke **Ctrl-Q**. If the results exceed 50 registers, then extra buttons *Fetch More* and *Fetch All* will appear. More details in the next chapters.

4.4 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*.



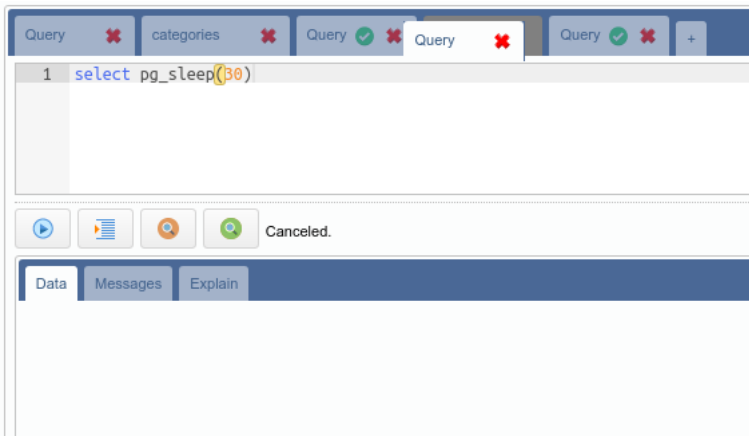
On OmniDB, you can execute several SQL statements and procedures in parallel. When it is executing, an icon will be shown in the tab to indicate its current state. If some process is finished and it is not in the current tab, that tab will show a green icon indicating the routine being executed there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



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



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



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

Starting from OmniDB version 2.3.0, all SQL Query tabs are automatically saved whenever you execute them. Even if you close OmniDB window or browser tab, they are already stored in OmniDB *User Database*. They will be automatically restored when you open OmniDB again (if you are using app), open it in another browser window (if you are using server), or even if you clicked in the *Connections* window or logged out. Removing an outer tab or inner tab by the interface makes it permanently deleted, so it will not be restored.

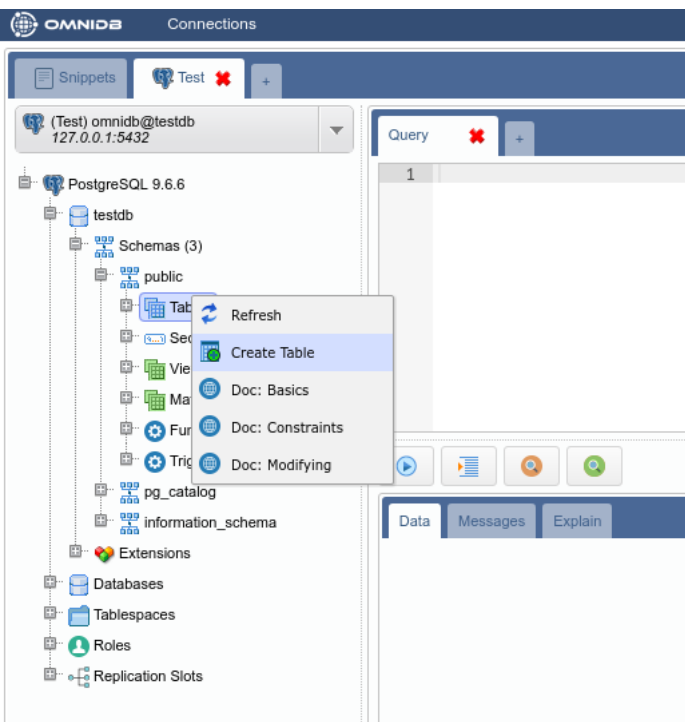
5 Creating, Changing and Removing Tables

5.1 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 **Create Table** action:



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

Query New Table +

Table Name:

Columns Constraints Indexes

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

Query New Table +

Table Name:

Columns Constraints Indexes

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

Click on the *Save Changes* button. Right-click the *Tables* tree node and click *Refresh*. Note how the table appears in the *Tables* tree node:

OMNIDB Connections

Snippets Test +

(Test) omnidb@testdb 127.0.0.1:5432

PostgreSQL 9.6.6

- testdb
 - Schemas (3)
 - public
 - Tables (1)
 - customer
 - Columns (2)
 - Primary Key (1)
 - cust_pk
 - cust_id
 - Foreign Keys
 - Uniques
 - Checks
 - Excludes
 - Indexes (1)
 - cust_pk (Unique)
 - cust_id
 - Rules
 - Triggers
 - Partitions
 - Sequences

Query public.customer +

Table Name:

Columns Constraints Indexes

	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:

Query public.customer New Table +

Table Name: Save Changes

Columns Constraints Indexes

	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				

Query public.customer New Table +

Table Name: Save Changes

Columns Constraints Indexes

New Constraint

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	✖

Don't forget to click on the *Save Changes* button when done. 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*:

OMNIDB Connections

Snippets Test +

(Test) omnidb@testdb 127.0.0.1:5432

PostgreSQL 9.6.6

testdb

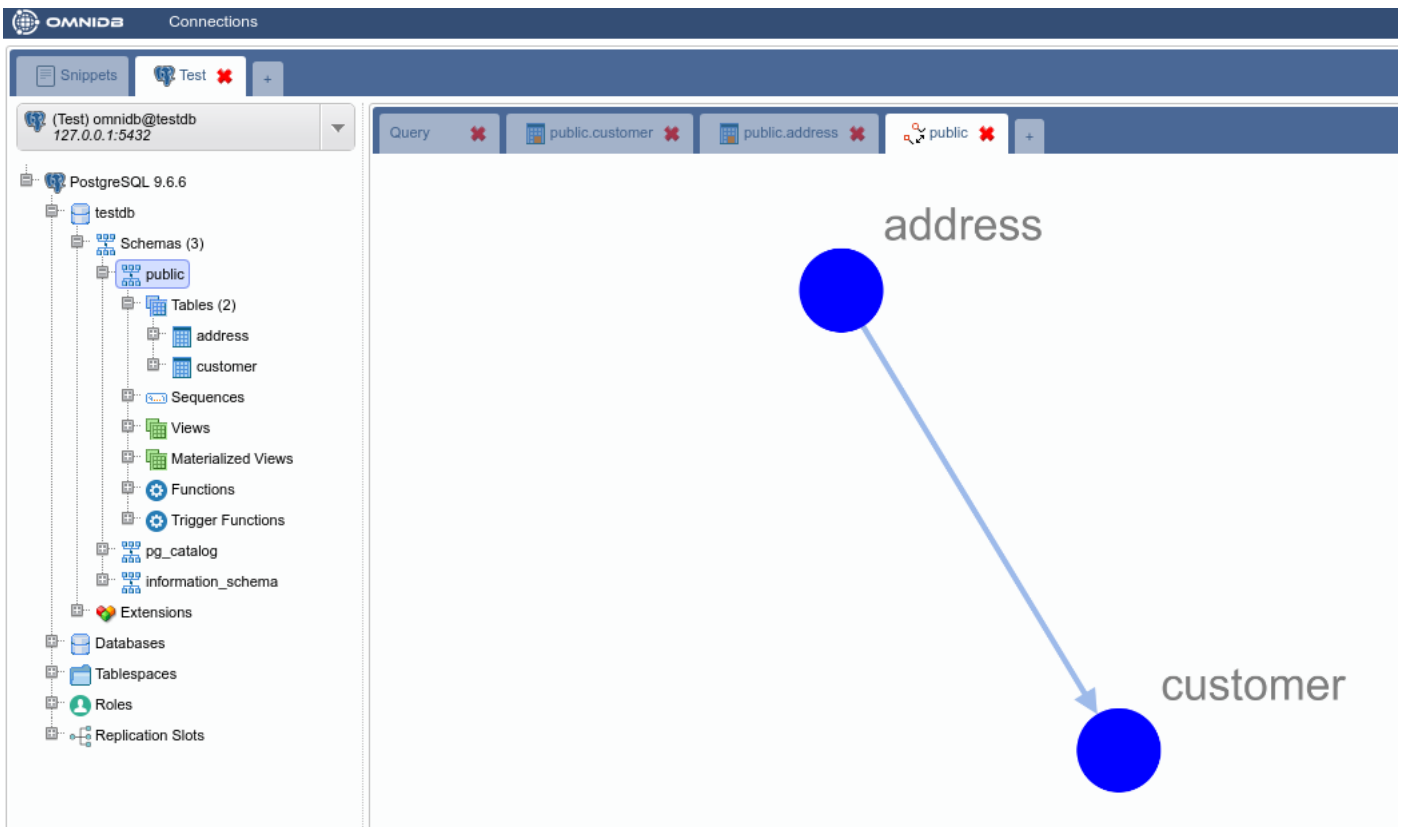
Schemas (3)

- public
 - Render Graph
 - Simple Graph
 - Complete Graph
 - Alter Schema
 - Drop Schema
- Sequences
- Views
- Materialized Views
- Functions
- Trigger Functions
- pg_catalog

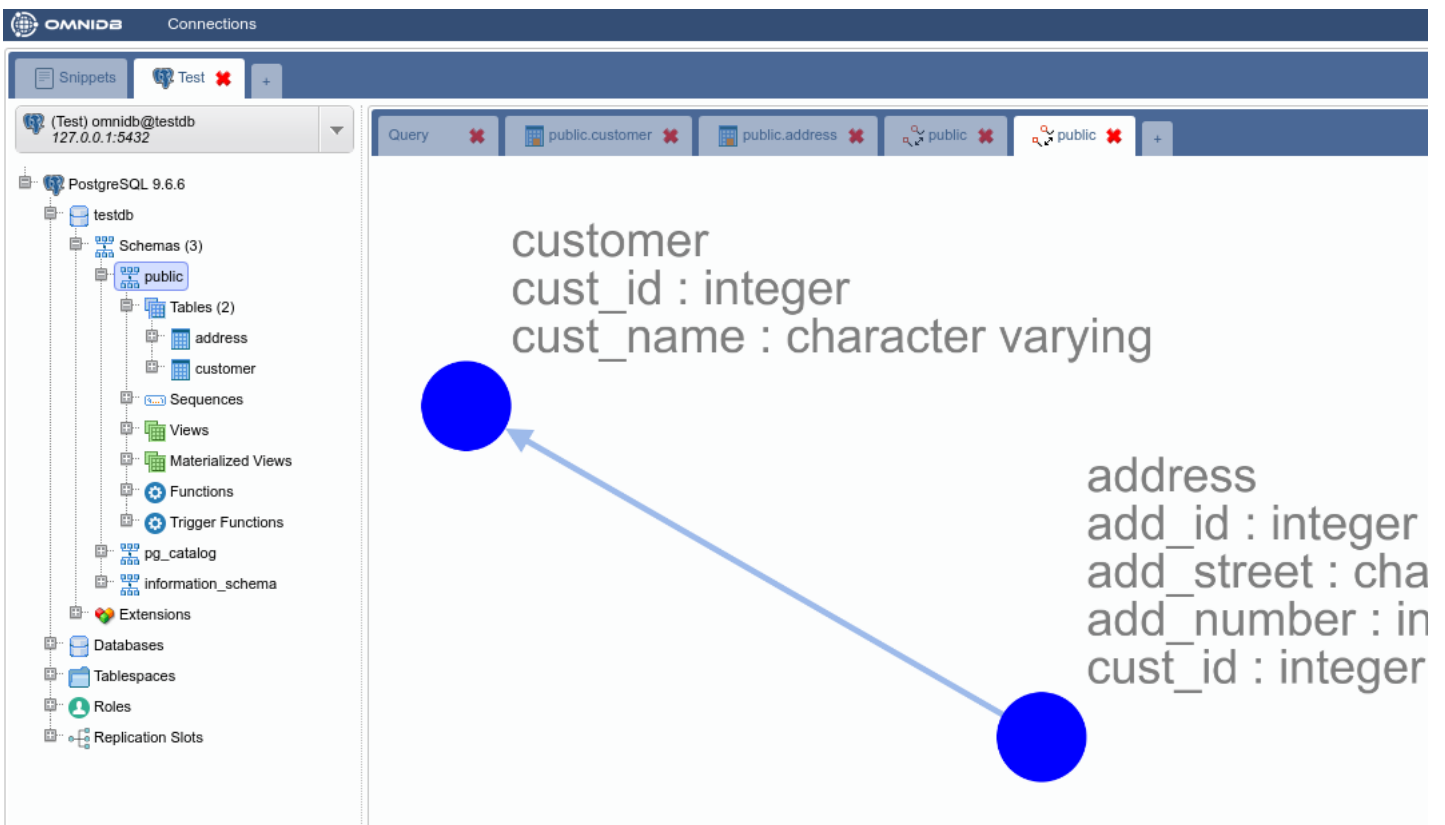
Table Name:

Columns Constraints Indexes

	Name	Data Type
1	add_id	integer
2	add_street	character varying(2
3	add_number	integer
4	cust_id	integer
5		

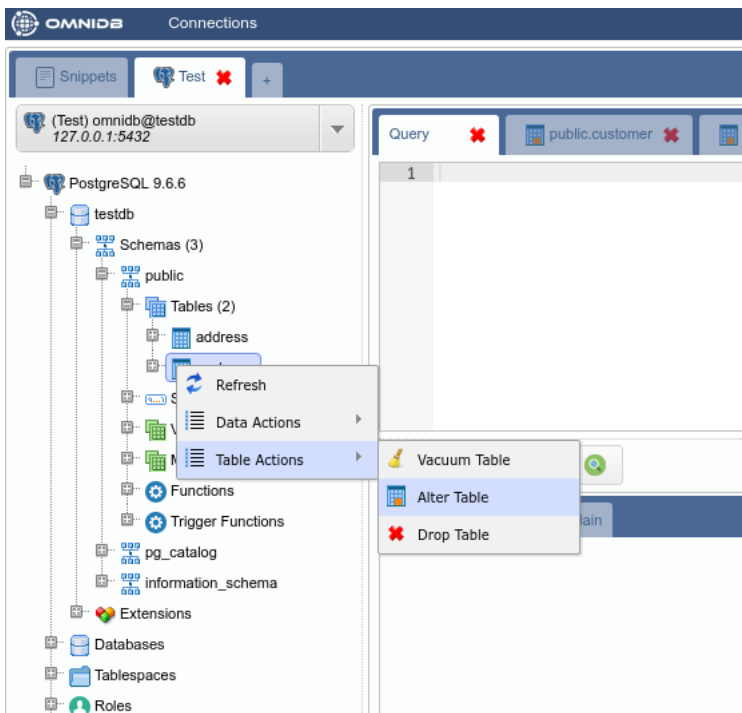


And this is what the *Complete Graph* looks like:

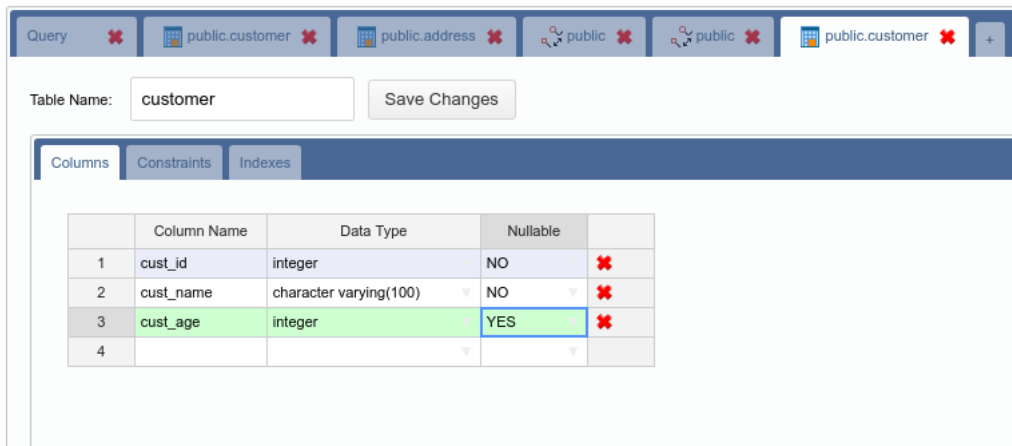


5.2 Editing tables

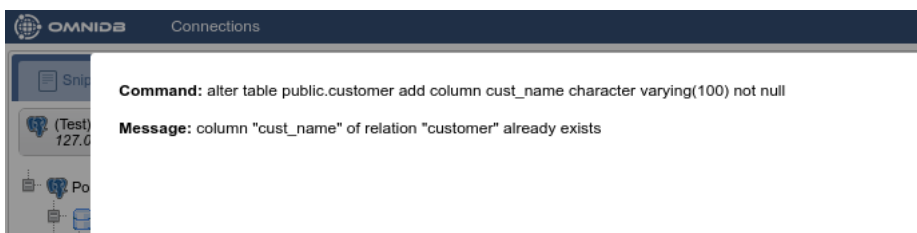
OmniDB also lets you edit existing tables (always following DBMS limitations). To test this feature we will add a new column to the table *Customer*. To access the alter table interface just right click the table node and select the action *Table Actions > Alter Table*:



Add the column `cust_age` and save:

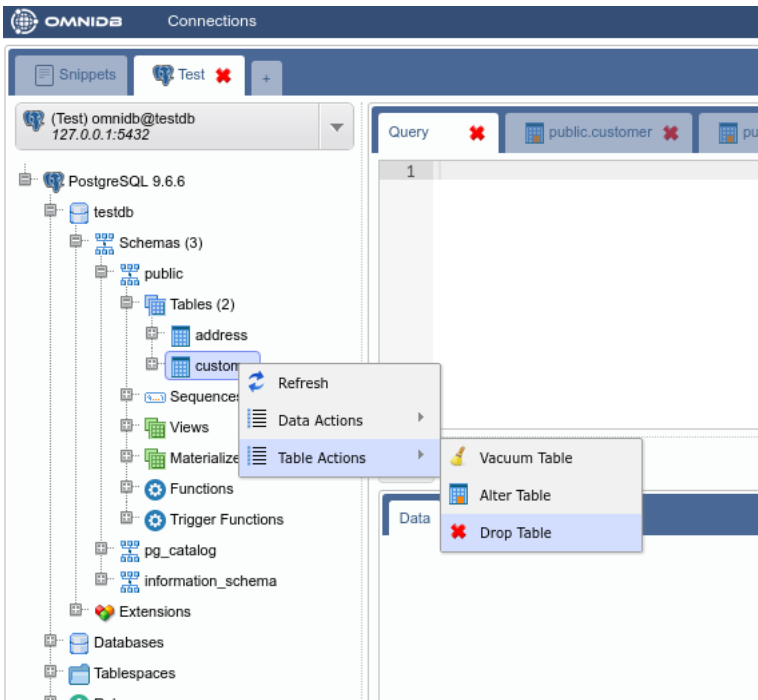


The interface is capable of detecting errors that may occur during alter table operations, showing the command and the error that occurred. To demonstrate it we will try to add the column `cust_name`, which already belongs to this table:



5.3 Removing tables

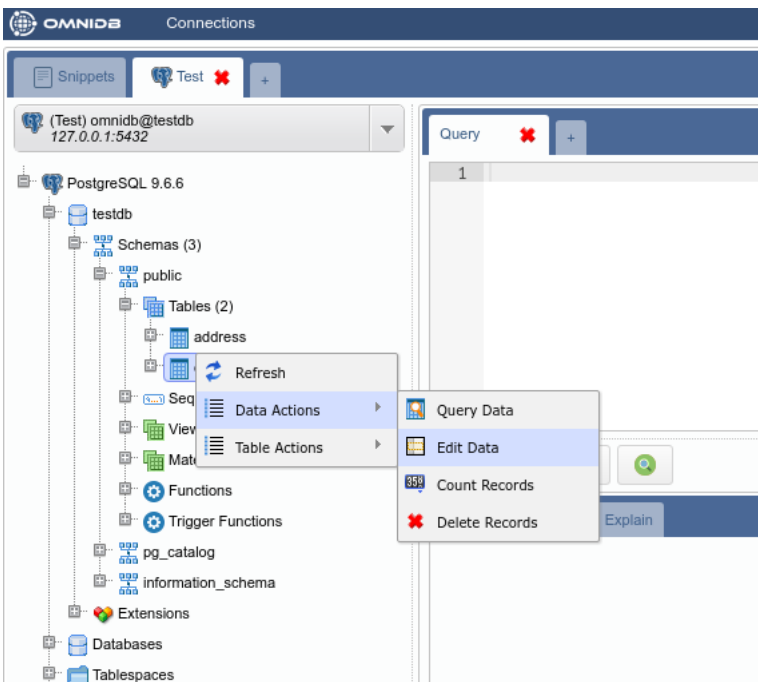
In order to remove a table just right click the table node and select the action *Table Actions > Drop Table*:

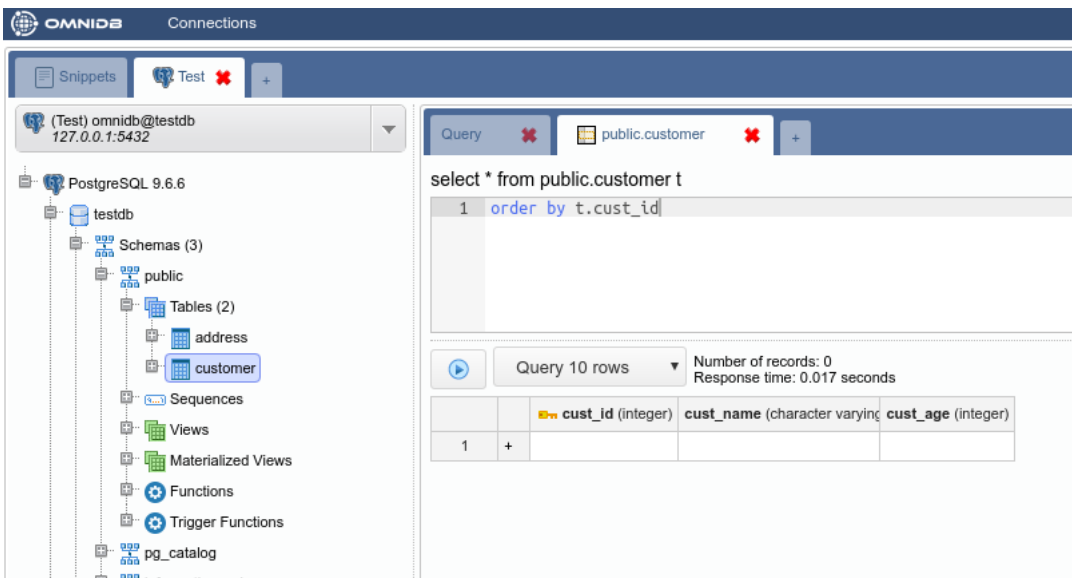


6 Managing Table Data

The tool allows 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 a primary key. Tables that do not have it can only receive new records.

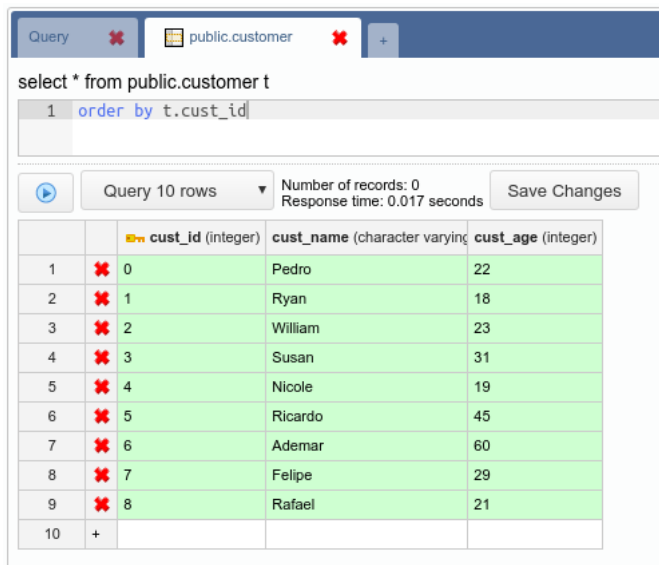
To access the record editing interface, right click the table node and select the action *Data Actions > Edit Data*:



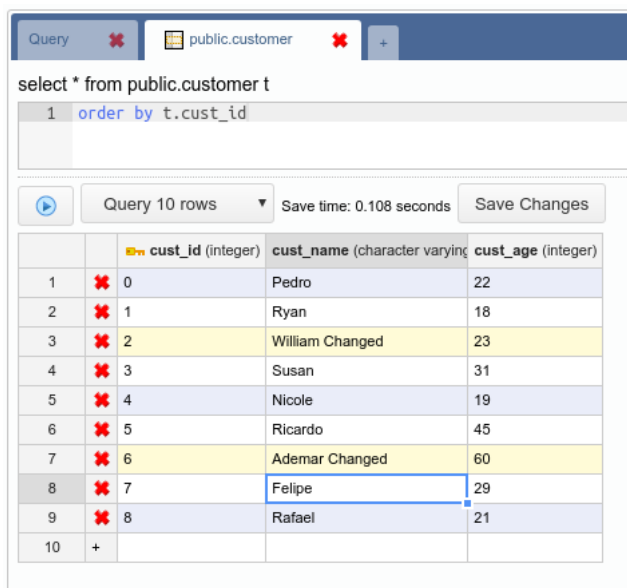


The interface has a SQL editor where you can filter and order records. To prevent that the interface requests too many records, there is a field that limits the number of records to be displayed. The records grid has column names and data types. Columns that belong to the primary key have a key icon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



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:



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:

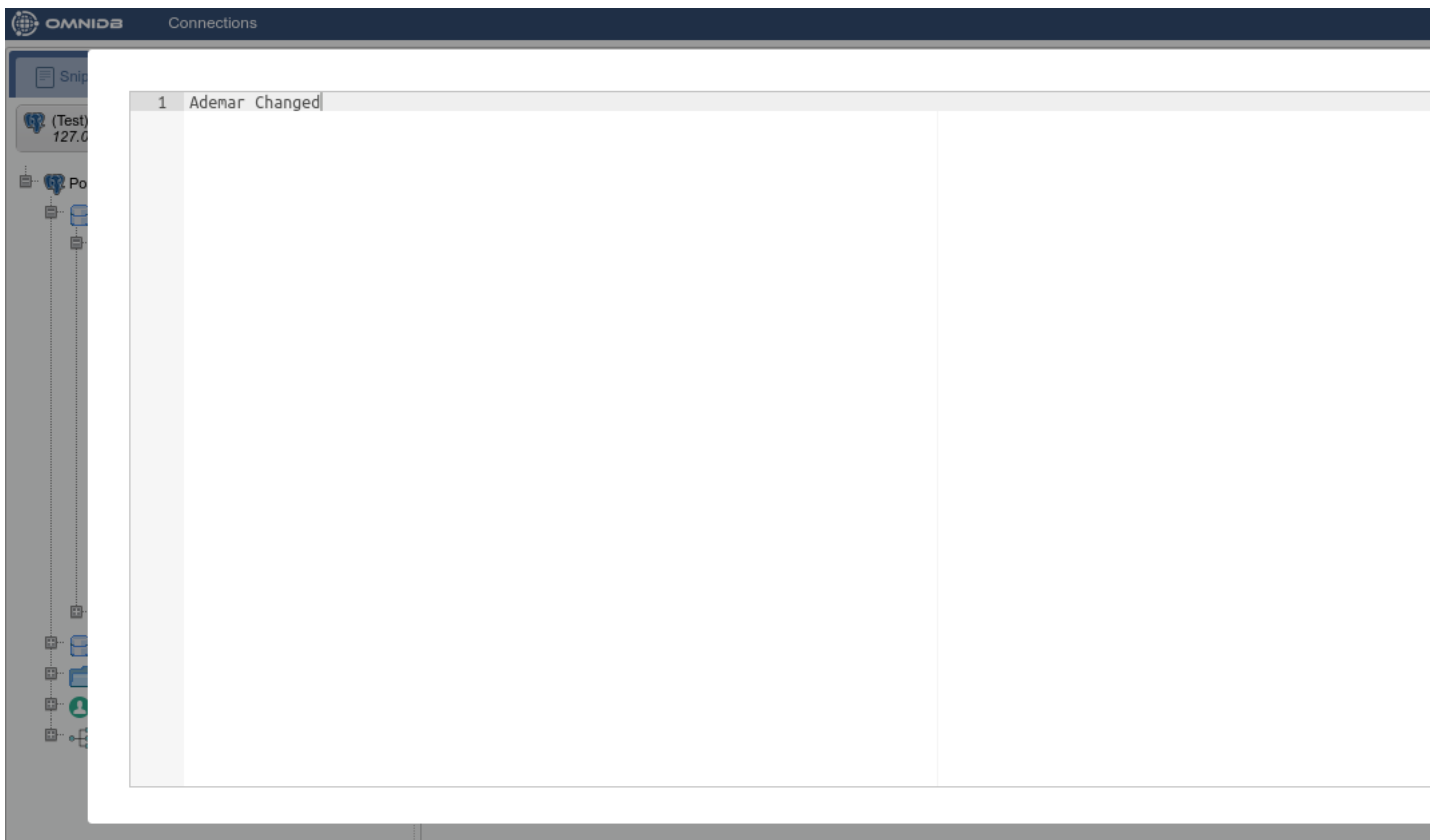
Query public.customer

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

Query 10 rows Save time: 0.038 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	60
8	✖	7	Felipe	
9	✖	8	Rafael	21
10	+			

Edit Content



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):

OMNIDB Connections

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:

Query	public.customer	public.address	+
-------	-----------------	----------------	---


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


▶
Query 10 rows
Number of records: 0
Response time: 0.022 seconds
Save Changes

		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	+				

7 Writing SQL Queries

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:

Query ✖ +

```

1 select *
2 from customer cust
3 where cust.cust_age > 30
4 order by cust

```

cust.

cust.cust_id

cust.cust_name

cust.cust_age

int4

varchar

int4

▶

📄

🔍

🔄

Data

Messages

Explain

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

Query ✖ +

```

1 select *
2 from (select cust.cust_name,
3         (select count(*)
4          from address addr
5          where addr.cust_id = cust.cust_id) as num_addresses
6        from customer cust) subquery
7 where subquery

```

subquery.

subquery.cust_name

subquery.num_addresses

varchar

int8

▶

📄

🔍

🔄

Data

Messages

Explain

If the query raises an error, OmniDB will show the error message in the *Data* tab and the cursor will be placed in the position indicated by the error message:

Query

```

1 select *
2 from (select cust.cust_name,
3         (select count(*)
4          from address addr
5          where addr.cust_id = cust.cust_id) as num_addresses
6         from customer cust) subquery
7 where subquery.cust_nam = 'William'

```

Start time: 12/05/2017 08:26:37 Duration: 8.91 ms

Data Messages Explain

column subquery.cust_nam does not exist
 LINE 7: where subquery.cust_nam = 'William'
 HINT: Perhaps you meant to reference the column "subquery.cust_name".

When the query is successful, OmniDB shows the number of records returned by the query, the start time and the duration of the query. The *Data* tab will show a data grid with the records returned by the query.

Query

```

1 select *
2 from (select cust.cust_name,
3         (select count(*)
4          from address addr
5          where addr.cust_id = cust.cust_id) as num_addresses
6         from customer cust) subquery

```

Number of records: 9
 Start time: 12/05/2017 08:24:52 Duration: 9.519 ms

Data Messages Explain

	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

Just like in the record editing interface each cell can be visualized separately by right clicking it.

If you click on the *Indent SQL* button, OmniDB will reorganize the SQL text to make it prettier:

Query

```

1 select *
2 from
3   (select cust.cust_name,
4        (select count(*)
5         from address addr
6         where addr.cust_id = cust.cust_id) as num_addresses
7        from customer cust) subquery
9 where subquery.cust_name = 'Rafael'

```

Number of records: 1
 Start time: 12/05/2017 09:21:01 Duration: 13.114 ms

Data Messages Explain

	cust_name	num_addresses
1	Rafael	1

8 Visualizing Query Plans

OmniDB 2.2.0 introduced a very useful feature: graphical query plan visualization. This may come in handy when writing or optimizing queries, since it allows you to easily identify performance bottlenecks in your SQL query.

For this feature, *SQL Query* inner tab shows 2 buttons: *Explain* (magnifier in orange circle button) and *Explain Analyze* (magnifier in green circle button).

8.1 Textual visualization

When you click the *Explain* button, OmniDB will execute an EXPLAIN command in your query. Initial visualization is *textual* and will show exactly the output of the EXPLAIN command, but with colored bars representing the estimated cost. The higher the cost, the darker and wider the bar.

```
Query
1 select *
2 from
3   (select cust.cust_name,
4        (select count(*)
5         from address addr
6         where addr.cust_id = cust.cust_id) as num_addresses
7        from customer cust) subquery
9 where subquery.cust_name = 'Rafael'
```

Start time: 12/05/2017 09:52:32 Duration: 9.481 ms

Data Messages Explain

QUERY PLAN

- 1 Seq Scan on customer cust (cost=0.00..38.27 rows=2 width=226)
- 2 Filter: ((cust_name)::text = 'Rafael')::text
- 3 SubPlan 1
- 4 Aggregate (cost=12.13..12.14 rows=1 width=8)
- 5 Seq Scan on address addr (cost=0.00..12.12 rows=1 width=0)
- 6 Filter: (cust_id = cust.cust_id)

When you click the *Explain Analyze* button, OmniDB will execute an EXPLAIN ANALYZE command in your query. Beware that this command will really execute the query. Also, the textual visualization will show much more information, and the costs are not estimated as in those provided by the EXPLAIN command; they are real costs.

```
Query
2 from
3   (select cust.cust_name,
4        (select count(*)
5         from address addr
6         where addr.cust_id = cust.cust_id) as num_addresses
7        from customer cust) subquery
9 where subquery.cust_name = 'Rafael'
```

Start time: 12/05/2017 10:03:26 Duration: 15.073 ms

Data Messages Explain

QUERY PLAN

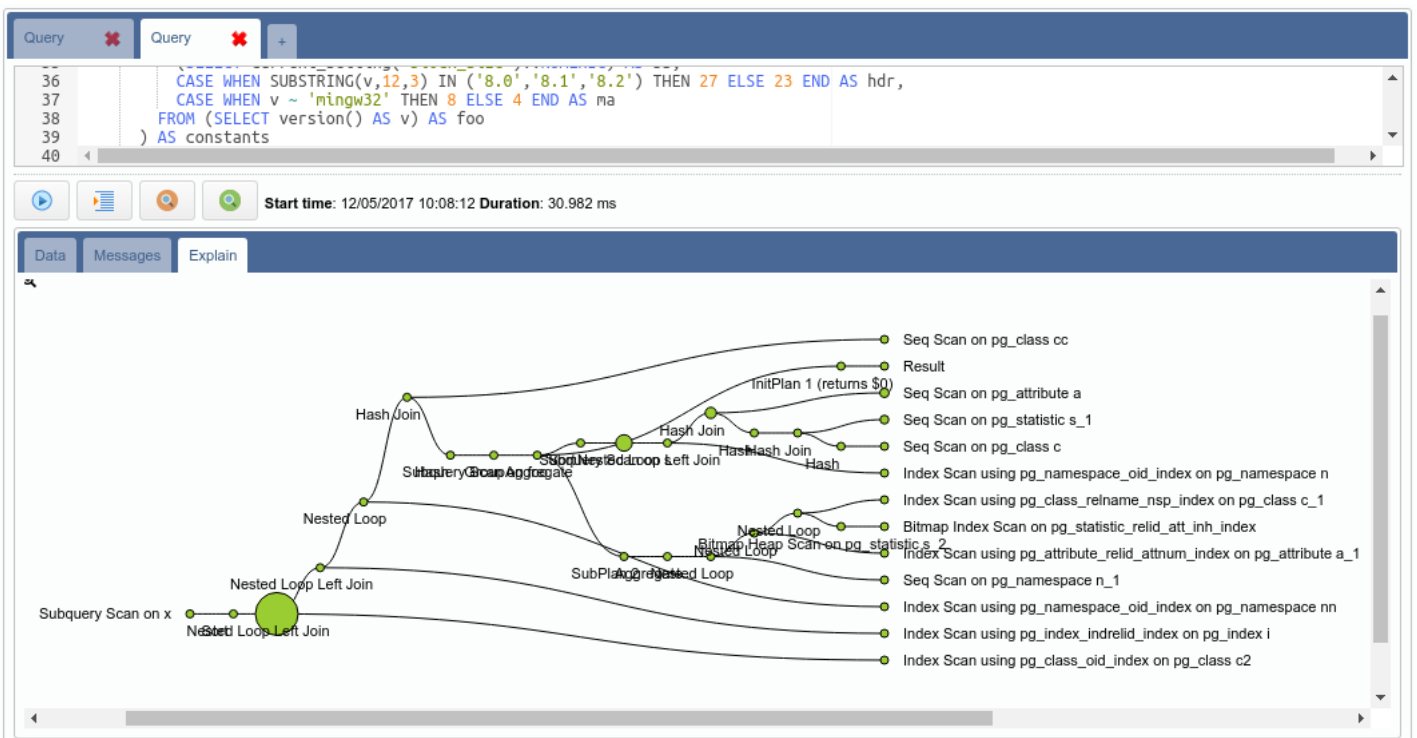
- 1 Seq Scan on customer cust (cost=0.00..38.27 rows=2 width=226) (actual time=0.029..0.029 rows=1 loops=1)
- 2 Filter: ((cust_name)::text = 'Rafael')::text
- 3 Rows Removed by Filter: 8
- 4 SubPlan 1
- 5 Aggregate (cost=12.13..12.14 rows=1 width=8) (actual time=0.017..0.017 rows=1 loops=1)
- 6 Seq Scan on address addr (cost=0.00..12.12 rows=1 width=0) (actual time=0.013..0.014 rows=1 loops=1)
- 7 Filter: (cust_id = cust.cust_id)
- 8 Rows Removed by Filter: 8
- 9 Planning time: 0.323 ms
- 10 Execution time: 0.097 ms

8.2 Tree visualization

Both *Explain* and *Explain Analyze* modes also can graphically represent the textual output into a *tree* diagram. Each circle represent a node executed by the query plan, and the larger the circle, the higher the cost.



When queries become more and more complex, also its query plan can be very complex. With such queries (like the *check bloat* query we executed below) the tree visualization can be very interesting:



The query plan visualization component allows you to easily switch between textual and 2 tree visualizations, which can be zoomed in and out.

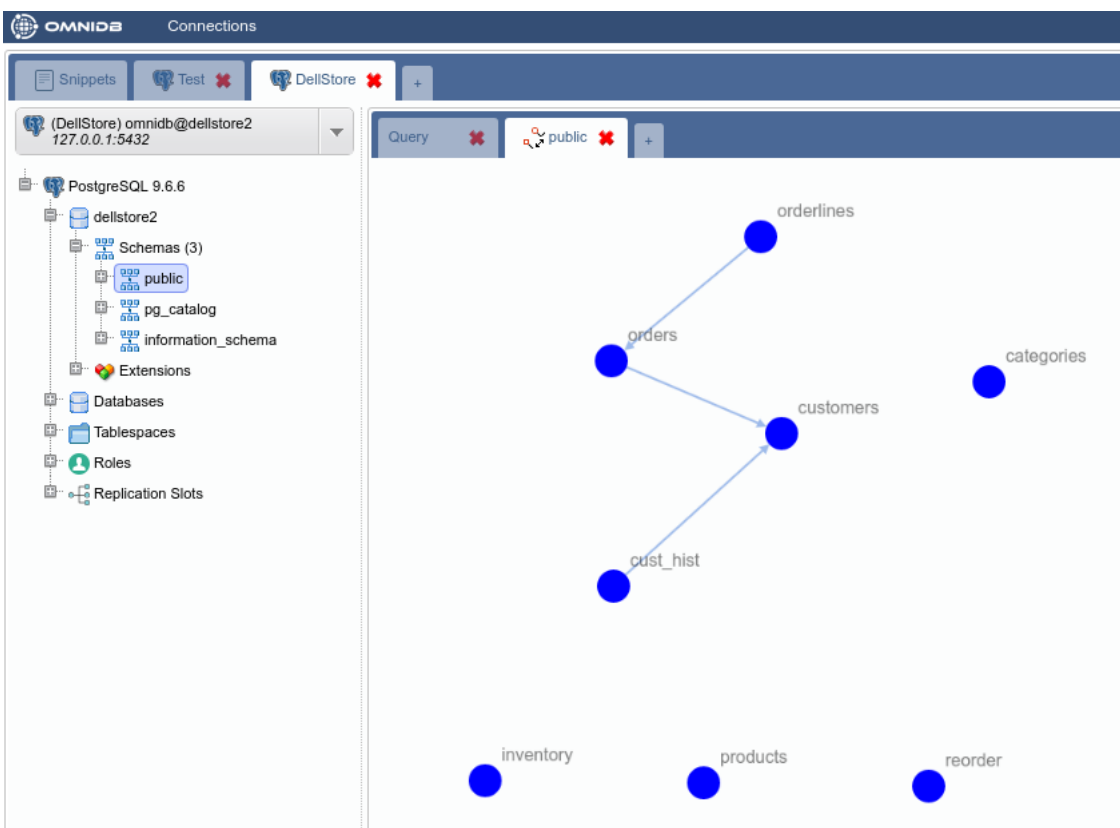
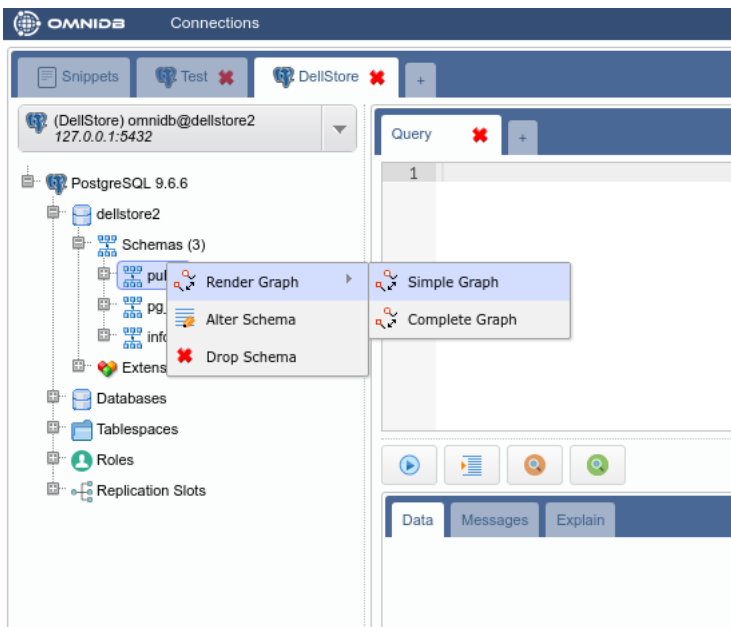
9 Visualizing Data

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*.

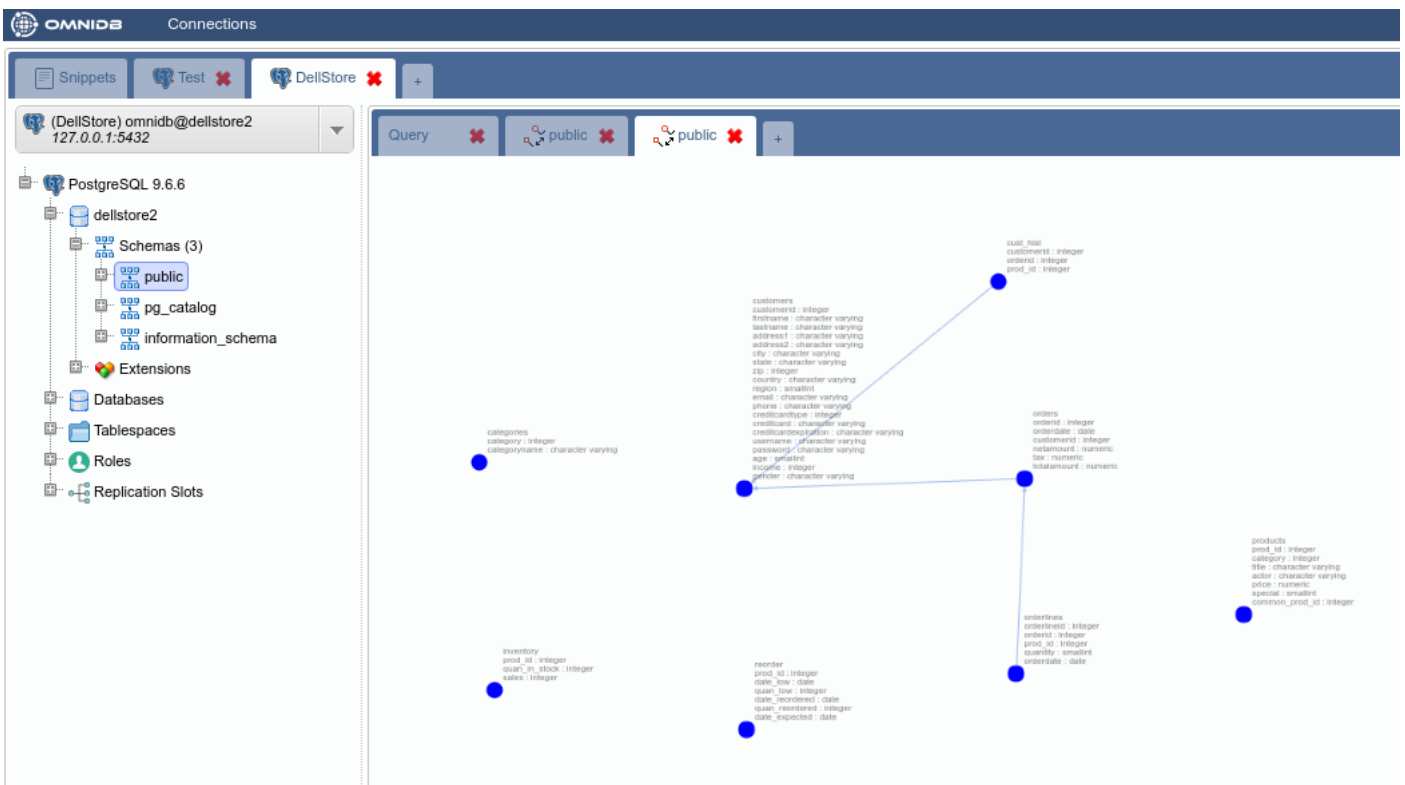
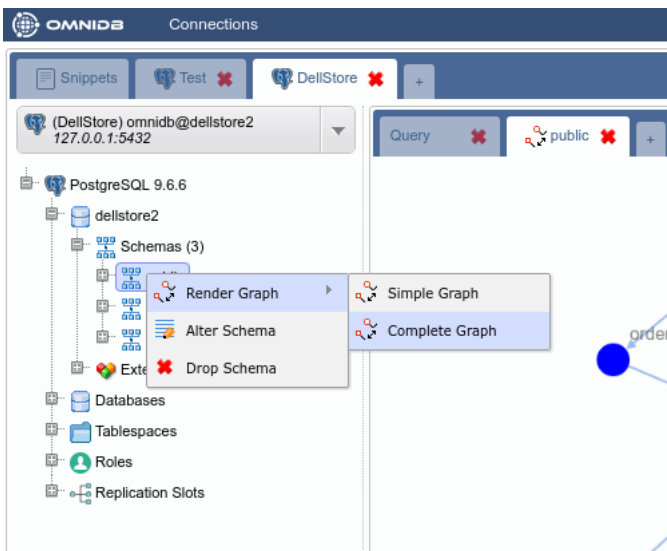
9.1 Simple graph

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

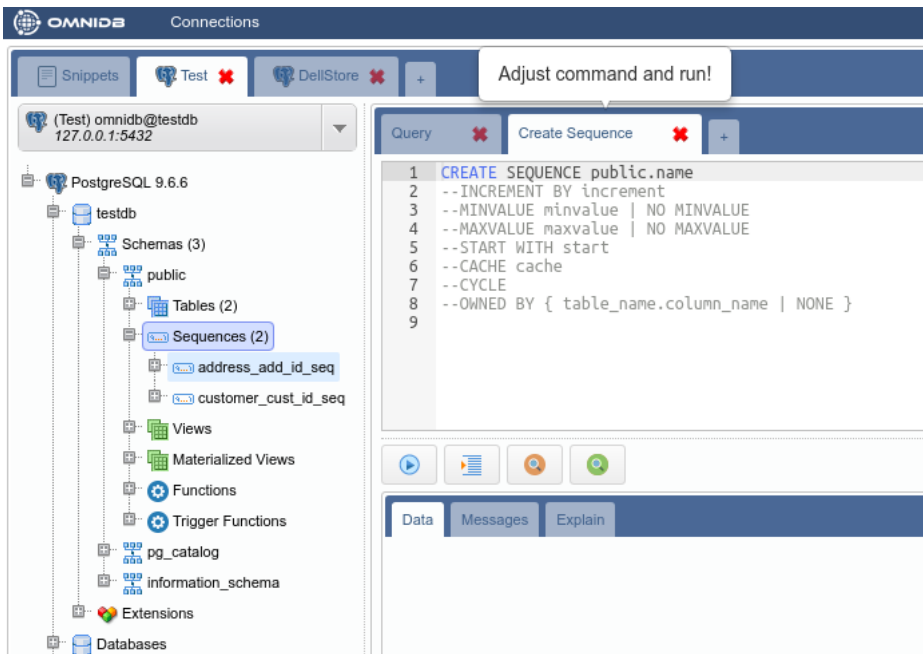
This graph displays tables with all its columns and respective data types. Additionally, edges now are labeled with information about the specific foreign key. To access it just right click the root node of the tree and then select the action *Render Graph > Complete Graph*:



10 Managing other PostgreSQL Elements

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 *Create Sequence*.



After you change the name of the sequence, you can uncomment other command options and set them accordingly to your needs. When the entire command looks fine, just execute it and the new sequence will be created:

The screenshot shows the OMNIDB interface with a connection to PostgreSQL 9.6.6. The left sidebar displays the database structure: testdb > Schemas (3) > public > Sequences (3). The 'seq_test' sequence is selected, showing its properties: Minimum Value: 1, Maximum Value: 9223372036854775807, Current Value: 1, and Increment: 2. The right pane shows the SQL query used to create the sequence:

```
1 CREATE SEQUENCE public.seq_test
2 INCREMENT BY 2
3 --MINVALUE minvalue | NO MINVALUE
4 --MAXVALUE maxvalue | NO MAXVALUE
5 --START WITH start
6 --CACHE cache
7 --CYCLE
8 --OWNED BY { table_name.column_name | NONE }
9
```

Below the query, the execution status is shown: Start time: 12/05/2017 11:37:06 Duration: 116.53 ms. The bottom tabs are Data, Messages, and Explain, with 'Data' selected, showing 'Done.'

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.

This screenshot shows the same OMNIDB interface, but with the 'seq_test' sequence selected in the left sidebar. A right-click context menu is open over the 'seq_test' sequence, showing two options: 'Alter Sequence' and 'Drop Sequence'. The 'Drop Sequence' option is highlighted with a red 'X' icon. The sequence properties are still visible in the background: Minimum Value: 1, Maximum Value: 9223372036854775807, Current Value: 1, and Increment: 2.

Query
Create Sequence
Alter Sequence
+

```

1 ALTER SEQUENCE public.seq_test
2 INCREMENT BY 1
3 --MINVALUE minvalue | NO MINVALUE
4 --MAXVALUE maxvalue | NO MAXVALUE
5 --START WITH start
6 --RESTART
7 --RESTART WITH restart
8 --CACHE cache
9 --CYCLE
10 --NO CYCLE
11 --OWNED BY { table_name.column_name | NONE }
12 --OWNER TO { new_owner | CURRENT_USER | SESSION_USER }
13 --RENAME TO new_name
14 --SET SCHEMA new_schema
15

```

Start time: 12/05/2017 11:39:39 Duration: 106.117 ms

Data Messages Explain

Done.

Query
Create Sequence
Alter Sequence
Drop Sequence
+

```

1 DROP SEQUENCE public.seq_test
2 --CASCADE
3

```

Start time: 12/05/2017 11:40:08 Duration: 115.908 ms

Data Messages Explain

Done.

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 icon with an *H* on the upper right corner. OmniDB will open an inner tab with all actions in a paginated grid.

History
+

First Previous 1 / 1 Next Last Refresh Clear List

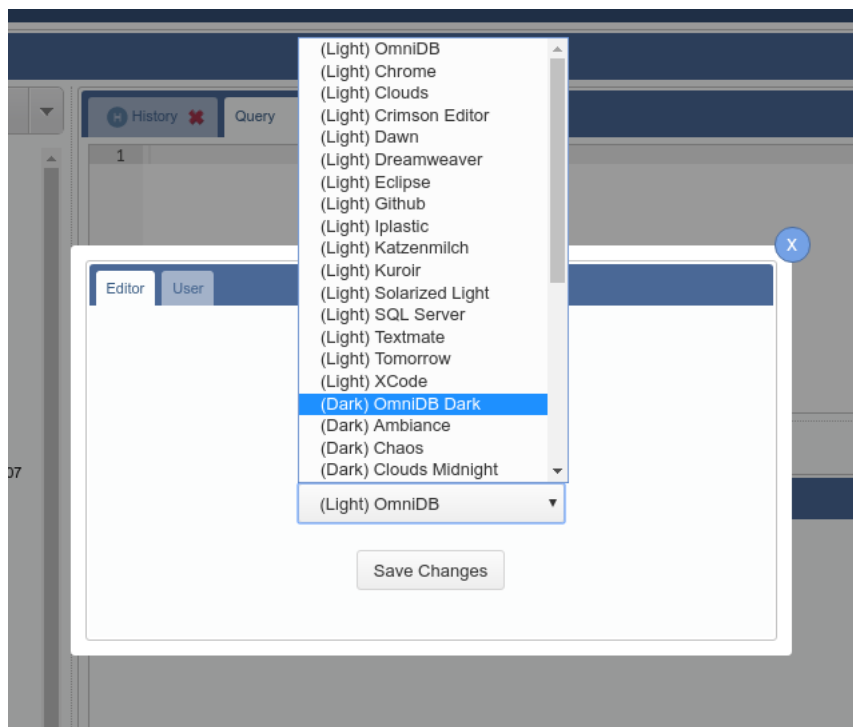
	Start	End	Duration	Status	
1	2017-12-05 13:40:08.201896	2017-12-05 13:40:08.317804	115.908 ms	●	DROP SEQUENCE public.seq_test -
2	2017-12-05 13:39:39.692279	2017-12-05 13:39:39.798396	106.117 ms	●	ALTER SEQUENCE public.seq_test
3	2017-12-05 13:37:06.486254	2017-12-05 13:37:06.602784	116.53 ms	●	CREATE SEQUENCE public.seq_te
4	2017-12-05 11:48:17.332837	2017-12-05 11:48:17.430137	97.3 ms	●	explain select * from (select cust.cus
5	2017-12-05 11:21:01.471111	2017-12-05 11:21:01.484225	13.114 ms	●	select * from (select cust.cust_name,
6	2017-12-05 11:20:45.249422	2017-12-05 11:20:45.258836	9.414 ms	●	select * from (select cust.cust_name,
7	2017-12-05 11:20:33.822952	2017-12-05 11:20:33.916459	93.507 ms	●	-- Querying Data select t.* from publi
8	2017-12-05 11:20:26.960773	2017-12-05 11:20:26.971530	10.757 ms	●	select * from (select cust.cust_name,
9	2017-12-05 11:13:30.014418	2017-12-05 11:13:30.024597	10.179 ms	●	select * from (select cust.cust_name,
10	2017-12-05 10:26:37.547517	2017-12-05 10:26:37.556427	8.91 ms	●	select * from (select cust.cust_name,
11	2017-12-05 10:24:52.766592	2017-12-05 10:24:52.776111	9.519 ms	●	select * from (select cust.cust_name,
12	2017-12-05 10:22:35.513501	2017-12-05 10:22:35.604137	90.636 ms	●	select * from (select cust.cust_name,
13	2017-11-30 19:25:50.454071	2017-11-30 19:25:50.472099	18.028 ms	●	DROP TABLE public.customer --CAS
14	2017-11-30 19:25:46.188620	2017-11-30 19:25:46.217045	28.425 ms	●	DROP TABLE public.address --CAS
15	2017-11-30 18:40:27.042825	2017-11-30 18:40:28.937599	00:00:01	●	select pg_sleep(30)
16	2017-11-30 18:39:46.856919	2017-11-30 18:40:16.874995	00:00:30	●	select pg_sleep(30)
17	2017-11-30 18:39:15.600141	2017-11-30 18:39:20.617678	00:00:05	●	select pg_sleep(5)
18	2017-11-30 18:39:07.229556	2017-11-30 18:39:12.255480	00:00:05	●	select pg_sleep(5)
19	2017-11-30 18:30:11.140417	2017-11-30 18:30:11.161083	20.666 ms	●	-- Querying Data select t.* from publi

Each action shows date time it started, the time it ended, the duration, 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

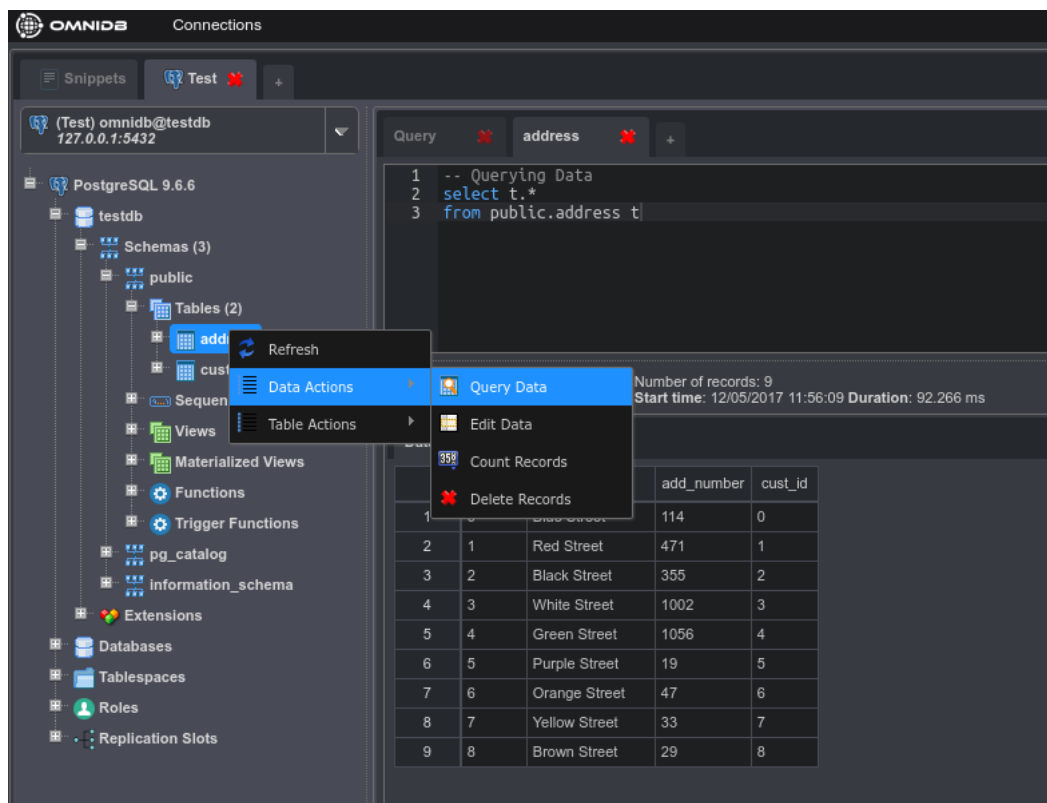
Also in the upper right corner, by clicking in the gear-like icon, OmniDB will open the *User Settings* pop-up. It is composed by two tabs:

- **User:** Allows the user to change its password. More user settings will be added in the 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.



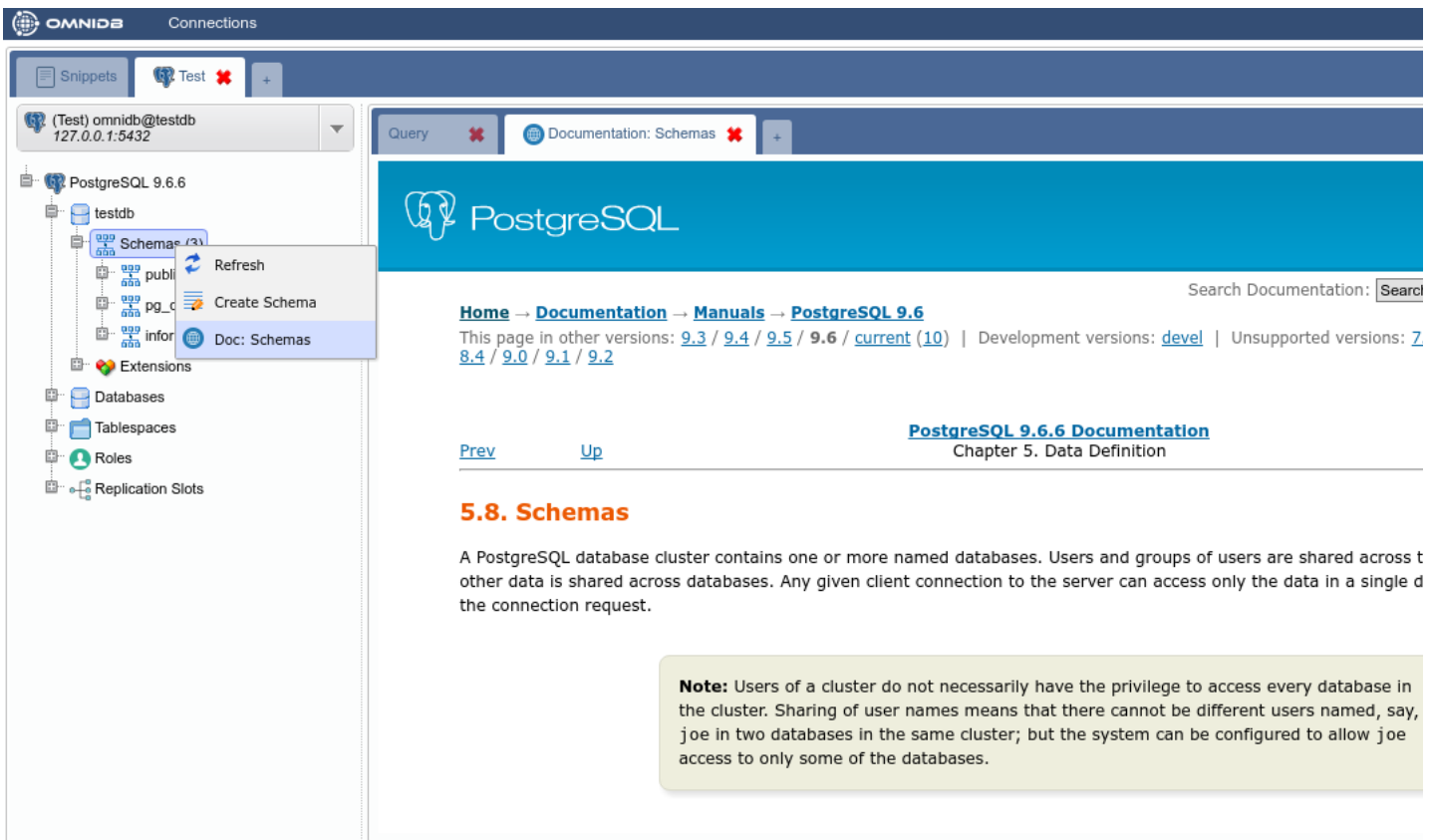
Every change in user settings require that you either:

- 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.



11.3 Contextual Help

Most of tree nodes (generally grouping ones like *Schemas* or *Tables*) offer contextual help. This feature can be accessed by right-clicking the tree node. When you click in the *Doc: ...* option, OmniDB will open an inner tab showing a web browser pointing to the specific page in the online **PostgreSQL Documentation**. Also, it will redirect to the specific page considering the PostgreSQL version you are connected to.



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.
```