OMNIDB

“A web tool for database management.”

User’s Guide
2016

OmniDB - User’s Guide

Rafael T. Castro, Luis Felipe T. Castro and William Ivanski

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 first version was created as an undergrad final project in the Computer Science Course from the Federal University of Paraná. The objective was to trace a common line between popular DBMS, and to study deeply their metadata. The result was a tool capable of connecting and identifying the main structures (tables, keys, indexes and constraints), in a generic way, from 7 DBMS:

- Firebird
- MySQL
- Oracle
- PostgreSQL
- SQLite
- Microsoft SQL Server
- Microsoft Access

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 may host any operational system, requiring only a web host that supports ASP.NET and C#. We will see further details on installation at the next chapter.

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.

2. Installation

2.1. Requirements

In order to run OmniDB, it's necessary to install *Microsoft .NET* (in Windows) or *Mono* and *XSP* (in Linux and Mac OS X). Further installation instructions for these technologies are explained below for each OS.

2.1.1. Microsoft Windows

If you are running Microsoft Windows 7 or newer, there's a big chance you already have Microsoft .NET installed. Otherwise, you can download it here[1] and then click twice in the file downloaded to start installation.

2.1.2. Linux

If you are running Debian or derivatives (which includes Ubuntu), Mono is already installed in default repositories. However, if you want to utilize Mono from the *Xamarin* Official Repository[2], open a terminal and type the following code:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3FA7E0328\
081BFF6A14DA29AA6A19B38D3D831EF
echo "deb http://download.mono-project.com/repo/debian wheezy main" | sudo tee /\
etc/apt/sources.list.d/mono-xamarin.list
sudo apt-get update
```

If running Debian 8.0 or later you must also add the following repository:

```
echo "deb http://download.mono-project.com/repo/debian wheezy-libjpeg62-compat m\
ain" | sudo tee -a /etc/apt/sources.list.d/mono-xamarin.list
sudo apt-get update
```

Regardless of your preference, to install Mono and XSP, type the following:

[1] https://www.microsoft.com/en-us/download/details.aspx?id=42642

[2] http://www.mono-project.com/download/#download-lin

```
sudo apt-get install mono-complete mono-xsp4
```

If you are running RedHat, CentOS or derivatives, you must include Xamarin's default repository. Open a terminal and type the following code:

```
su
rpm --import "http://keyserver.ubuntu.com/pks/lookup?op=get&search=0x3FA7E032808\
1BFF6A14DA29AA6A19B38D3D831EF"
yum-config-manager --add-repo http://download.mono-project.com/repo/centos/
```

Then, to install Mono and XSP:

```
yum install mono-complete xsp
```

2.1.3. Mac OS X

Download Mono's installer from the official page[3] and install it in your Mac OS X.

2.2. Download and first-time execution of OmniDB

Download the most recent version of OmniDB at the official page[4]. The file will be similar to *OmniDB-1.0.zip*, according to the version. Then, extract that file, which will create a folder called *OmniDB-1.0*.

Now you should be able to run the web server that will host OmniDB, in order to access it from the web browser. The web server depends on the operational system currently installed on your machine.

2.2.1. Linux and Mac OS X

Open a terminal e enter the folder *OmniDB-1.0* that you just extracted. Then, type the following command:

```
xsp4 --port 9000
```

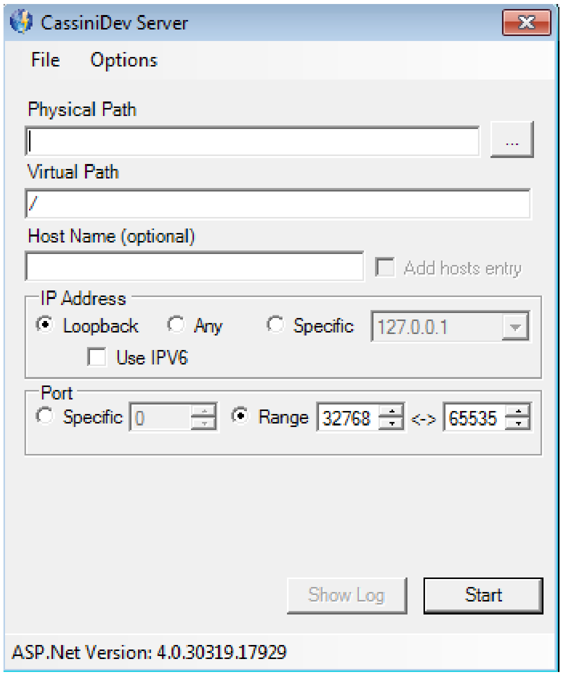
We recommend to always utilize port 9000. But you can still use another one if 9000 is already being used by another application.

[3] http://www.mono-project.com/download/#download-mac

[4] http://www.omnidb.com.br

2.2.2. Microsoft Windows

Open *Windows Explorer* and enter the folder *OmniDB-1.0* that you just extracted. Inside that folder there's a file called *cassini.exe*. Double-click it to initiate the web server *Microsoft Cassini*, which initially will look like the image below:



Physical Path

Virtual Path /

Host Name (optional)

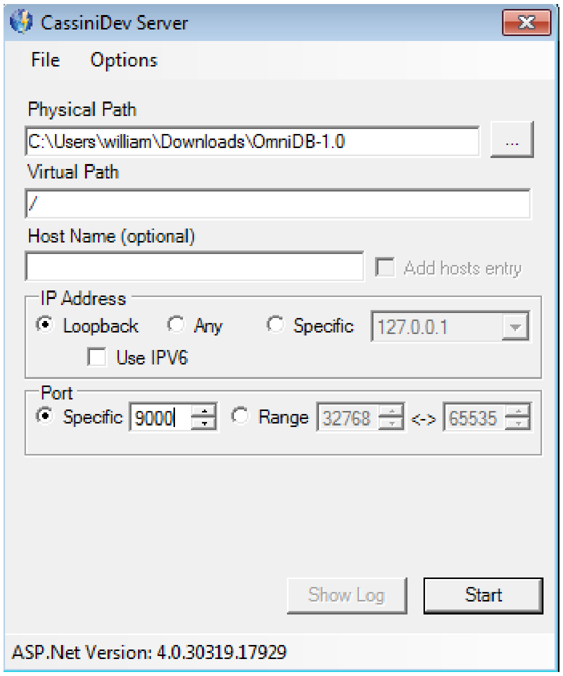
☐ Add hosts entry

IP Address
● Loopback ○ Any ○ Specific 127.0.0.1
☐ Use IPV6

Port
○ Specific 0 ● Range 32768 <-> 65535

ASP.Net Version: 4.0.30319.17929

Change the option *Physical Path* to point at OmniDB's folder, also, switch the port to 9000 (or one that is avaiable in case 9000 is already in use). The configuration must be set as shown below:



Physical Path: C:\Users\william\Downloads\OmniDB-1.0

Virtual Path: /

Host Name (optional): □ Add hosts entry

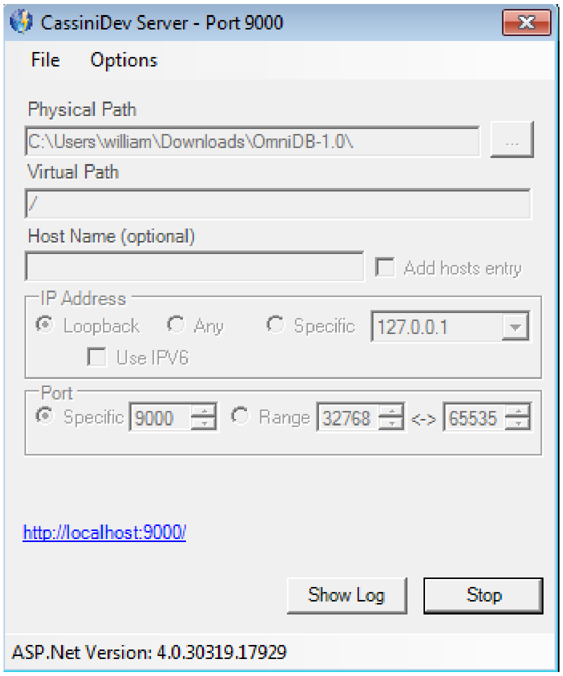
IP Address: ● Loopback ○ Any ○ Specific 127.0.0.1

□ Use IPV6

Port: ● Specific 9000 ○ Range 32768 <-> 65535

ASP.Net Version: 4.0.30319.17929

Then, click *Start*. Now, you should notice that Microsoft Cassini is running, according to the image below:



Physical Path: C:\Users\william\Downloads\OmniDB-1.0\

Virtual Path: /

Host Name (optional): □ Add hosts entry

IP Address: ● Loopback ○ Any ○ Specific 127.0.0.1

□ Use IPV6

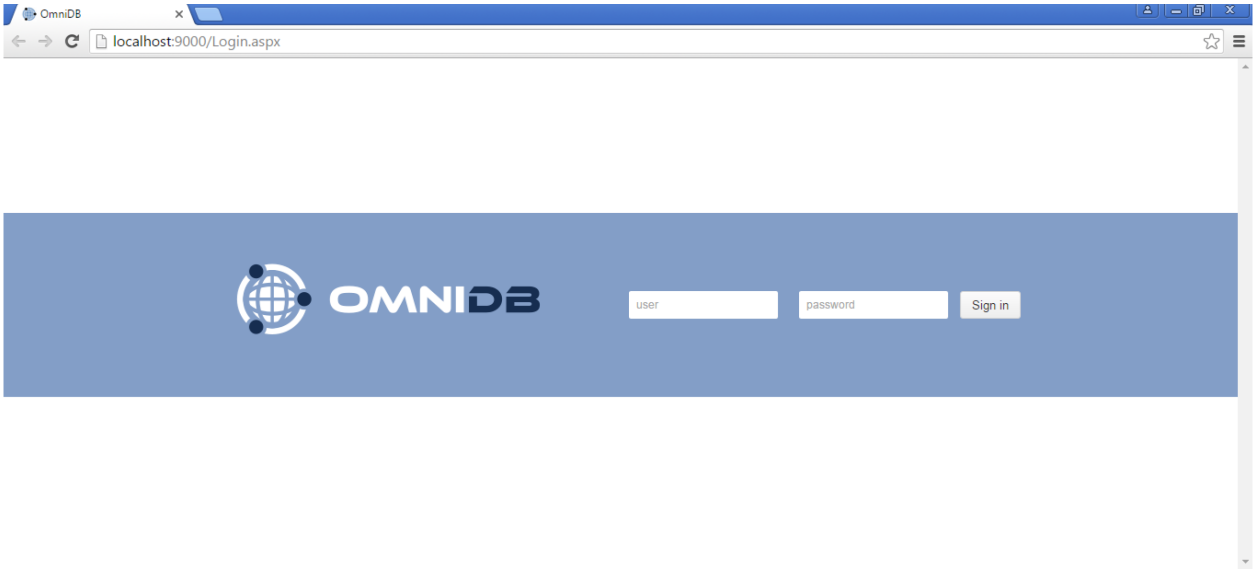
Port: ● Specific 9000 ○ Range 32768 <-> 65535

http://localhost:9000/

ASP.Net Version: 4.0.30319.17929

2.3. Accessing OmniDB in your browser

Now that the web server (XSP or Cassini) is running, you may access OmniDB web app on your favorite browser. Type in address bar: *http://localhost:9000*. If you are running in any port other than 9000, utilize the appropriate port number. If everything went fine, you shall see a page like this:



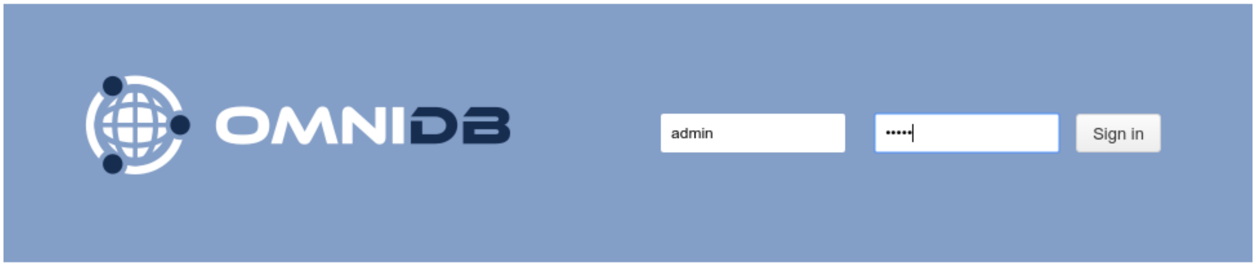
Now you know that OmniDB is running correctly. In the next chapter, we will see how to login for the first time, how to create an user and to utilize OmniDB.

3. Creating Users and Connections

Last chapter explained how to install and run OmniDB for the first time. The tool still needs a few steps before we can start using it.

3.1. Logging as user *admin*

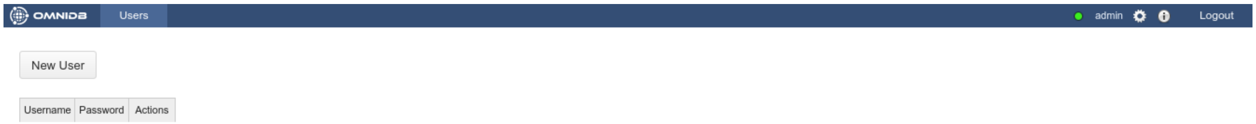
OmniDB comes only with the user *admin*, which is used to manage other users. The first thing to do is sign in as admin, the default password is *admin*.



admin

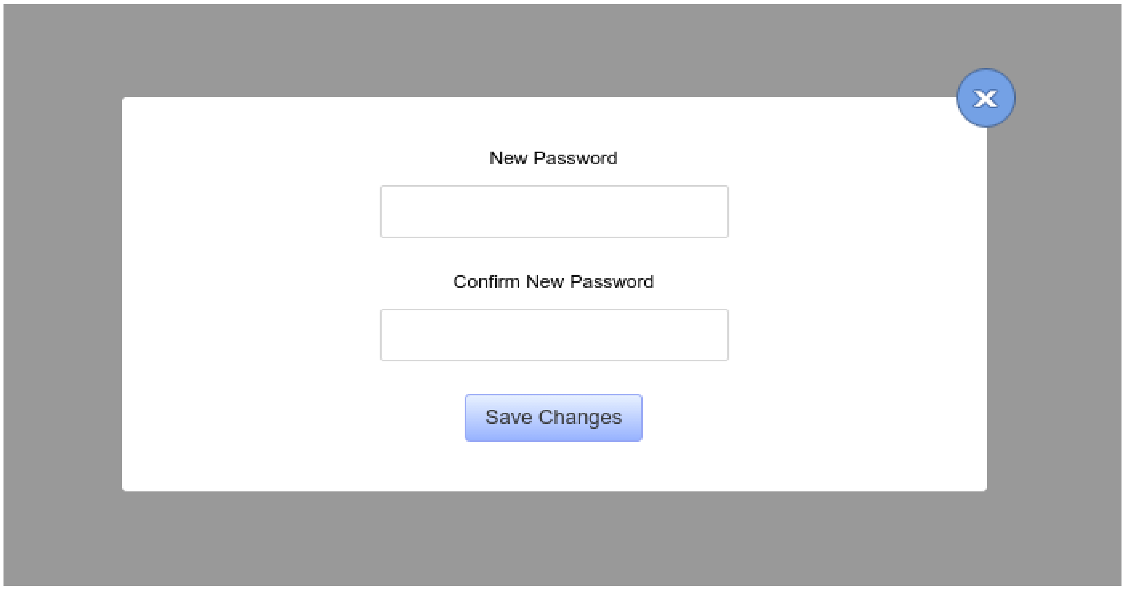
•••••

The next window is the user management interface, which can be accessed only with the user admin.



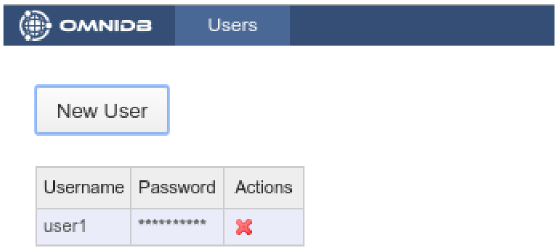
Username	Password	Actions

At the top right corner of the screen next to the logged username there are the options logout, information (that shows a popup with basic information about the tool) and also a gear icon that can be used to change basic settings of the current user. As admin you can only change its password, which is highly recommended.



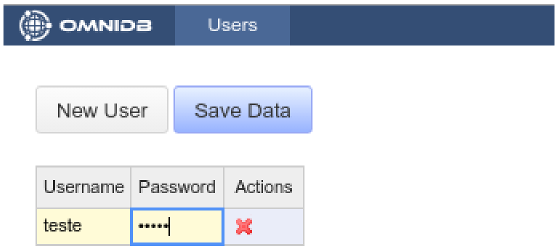
3.2. Creating the first user

After clicking on the *New User* button the tool inserts a new user called *user1* (if that is the first user).



Username	Password	Actions
user1	***********	✖

You will have to change the username and password and then click on the *Save Data* button.

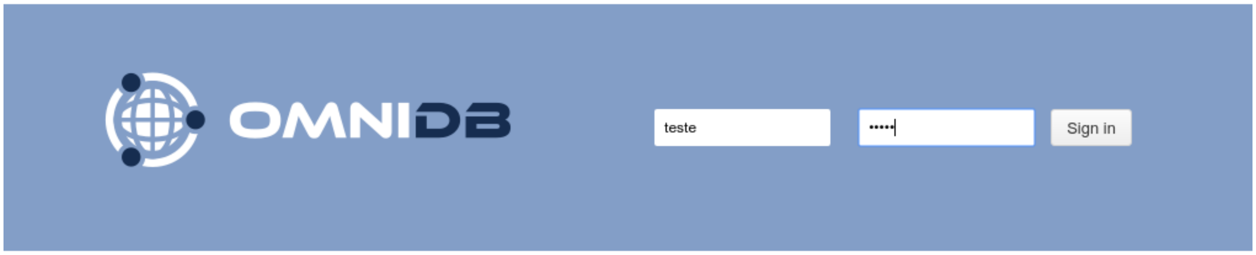


Username	Password	Actions
teste	•••••	✖

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

3.3. Logging with the first user

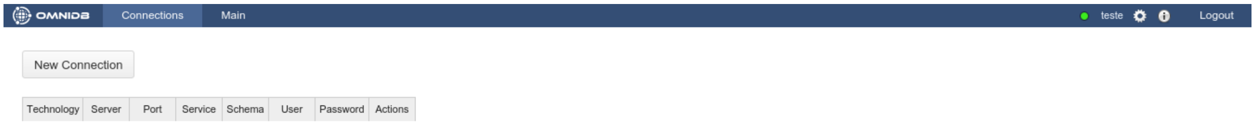
Let us sign in as the user we just created.



teste

•••••

Notice that you were redirected to a different window, which has two links: *Connections* and *Main*.



Technology	Server	Port	Service	Schema	User	Password	Actions

3.4. Creating connections

At the moment OmniDB supports 7 different DBMS: Firebird, MySQL, Oracle, PostgreSQL, SQLite, Microsoft SQL Server and Microsoft Access. Each technology has its unique characteristics, and the information required to establish a connection to each of them can also be different. OmniDB stores data to each connection according to the selected technology.

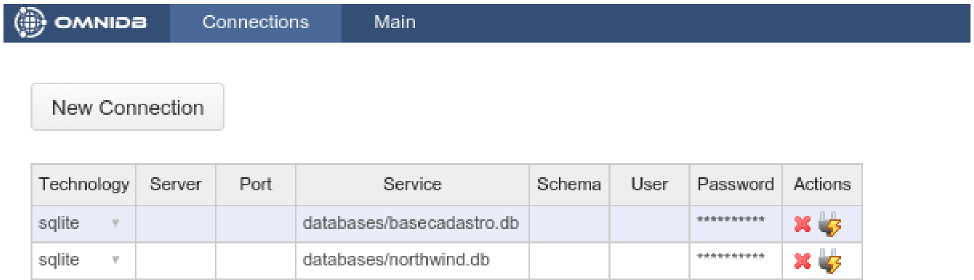
SGBD	Server	Port	Service	Schema	User	Password
Access			File path			
Firebird	IP Address	Port	File path		User	Password
MySQL	IP Address	Port	Database name		User	Password
Oracle	IP Address	Port	Database name		User	Password
PostgreSQL	IP Address	Port	Database name	Schema	User	Password
SQLite			File path			
SQL Server	IP Address	Port	Database name	Schema or empty for default	User	Password

Notice that the path to the file (when dealing with Firebird, SQLite and Access) can be absolute or relative to the path were OmniDB was installed. When connecting to Firebird database files it might be necessary to change the file permissions (check Firebird's online documentation).

We will now create two connections with SQLite databases that come with OmniDB:

- **databases/basecadastro.db**
- **databases/northwind.db**

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.

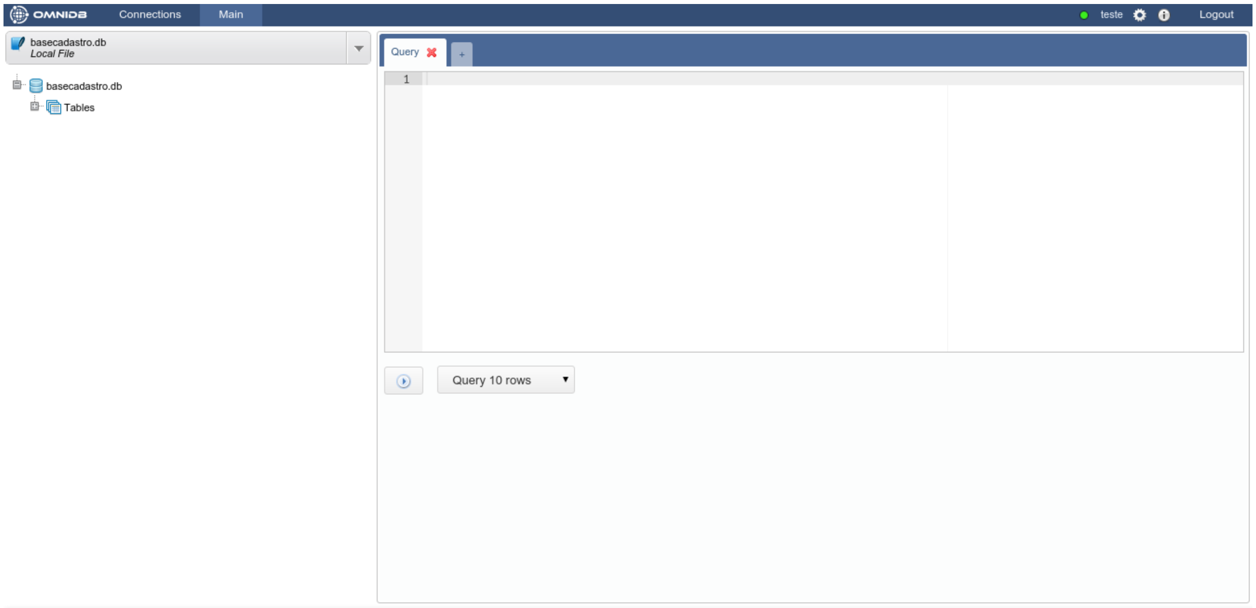


Technology	Server	Port	Service	Schema	User	Password	Actions
sqlite ▾			databases/basecadastro.db			**********	✖
sqlite ▾			databases/northwind.db			**********	✖

For each connection there is an actions column where you can delete and test them. Be advised that when handling SQLite databases, if the file does not exist it will be created (if the user that is running the server have permission to do so).

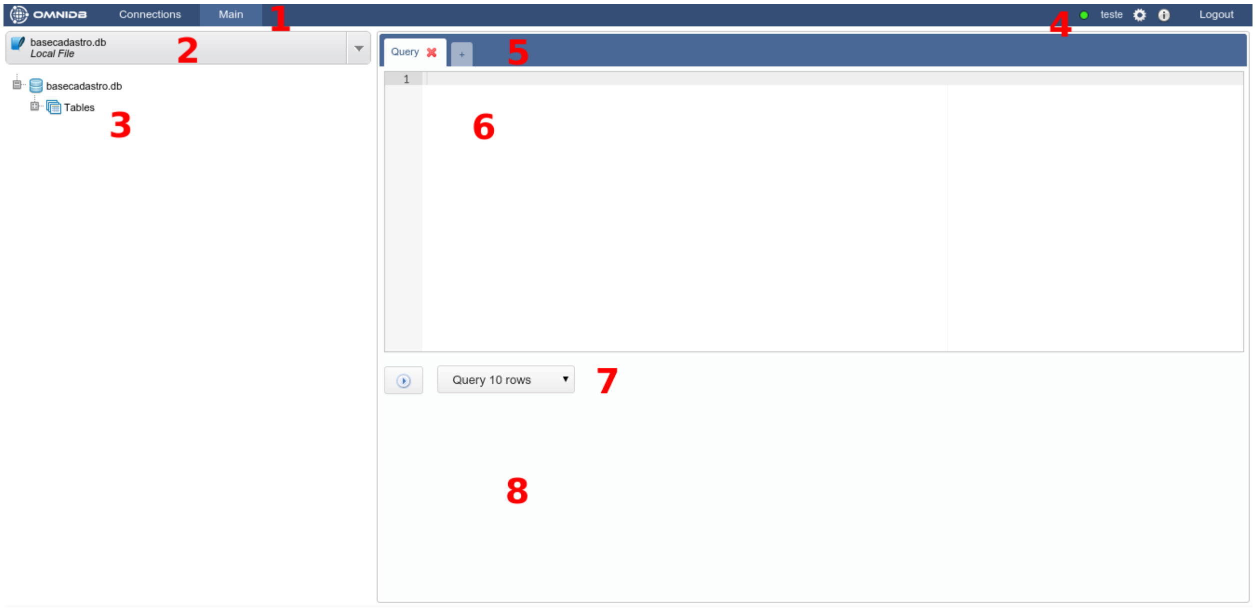
4. Main Window

After creating at least one connection the user can enter the *Main Window*. This interface will be your workspace.



4.1. Sections of the main window

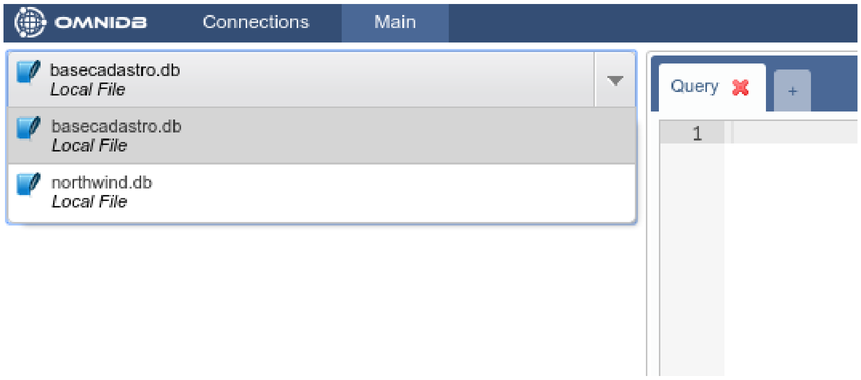
This interface has several elements:



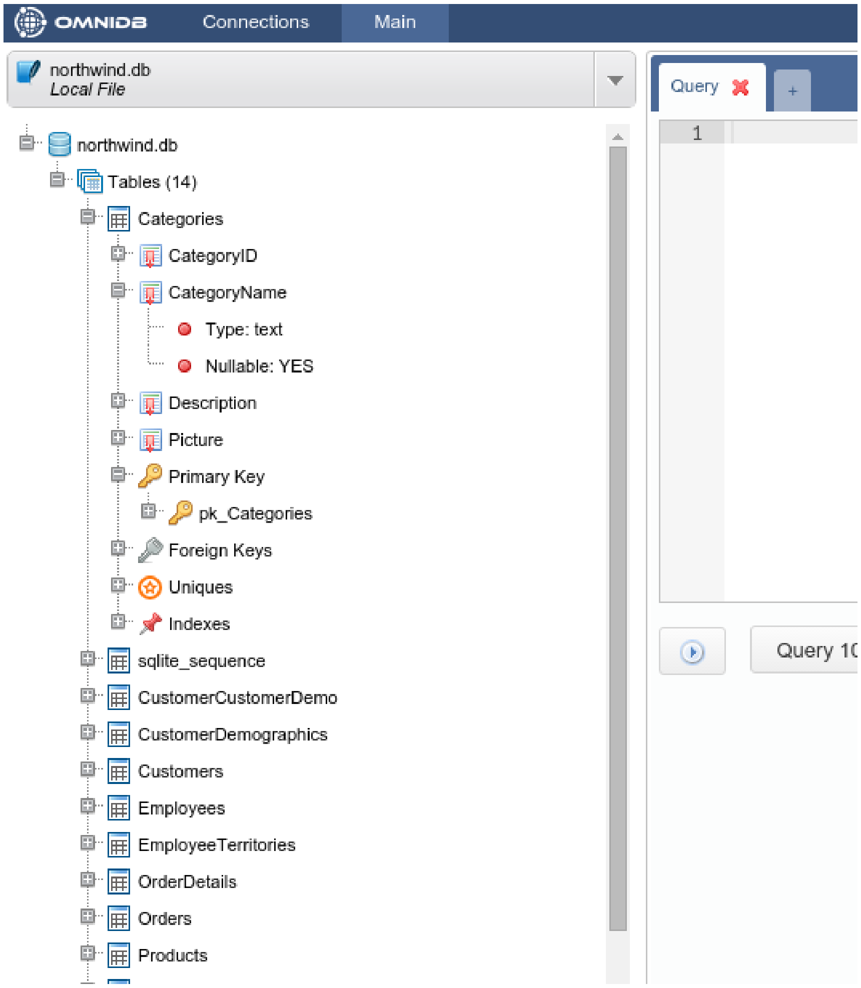
- **1) Links**: Enables the user to navigate between connections and main window
- **2) Connections selector**: Shows all connections and let the user select the current one.
- **3) Structures tree of the current connection**: Displays a tree with all tables of the current connection. For each table displays columns, primary, foreign and unique constraints and also indexes.
- **4) Options**: State of the connection with the web server, username, settings, information and logout.
- **5) Tabs with SQL editors**: OmniDB lets you work with multiple SQL editors at the same time. Each editor is inside a tab that can be renamed and removed.
- **6) Current SQL editor**: Editor of the current tab, with syntax highlighting and autocomplete.
- **7) Options of the current SQL editor**: Action button and editor mode (script, execute or query).
- **8) Result**: Shows a grid with the result when dealing with queries or command details when executing scripts.

4.2. Knowing your workspace

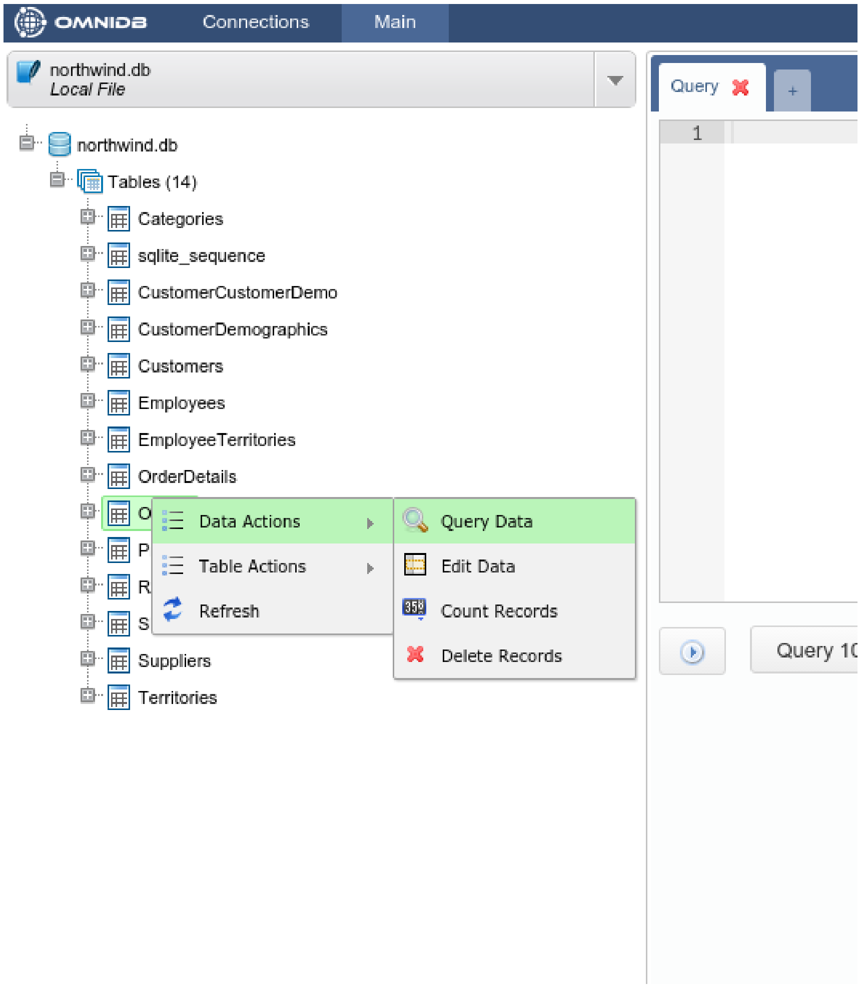
Take a look at your connections selector. OmniDB always points to the first available connection but you can change it by clicking on the selector.



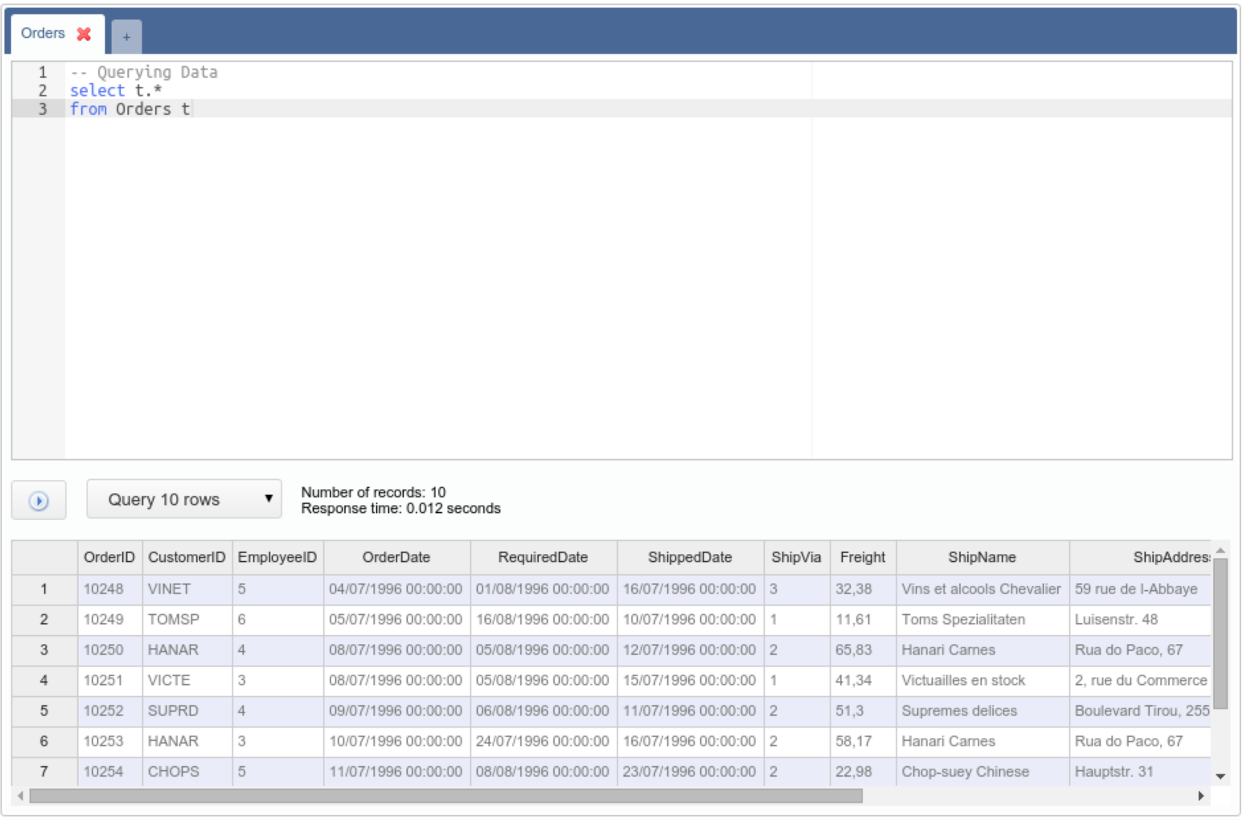
Select the *northwind.db* connection. Now go to the tree right below the selector and click to expand the node Tables. You will see all tables contained on this database. Expand any table and you will see its columns, primary key, foreign keys, unique constraints and indexes. Each column is also expansible, 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 (Microsoft Office Excel, LibreOffice Calc). You can also copy data from single cells or block of cells (that can be selected with the mouse) and paste on any spreadsheet manager.



```sql
-- Querying Data
select t.*
from Orders t
```

Query 10 rows

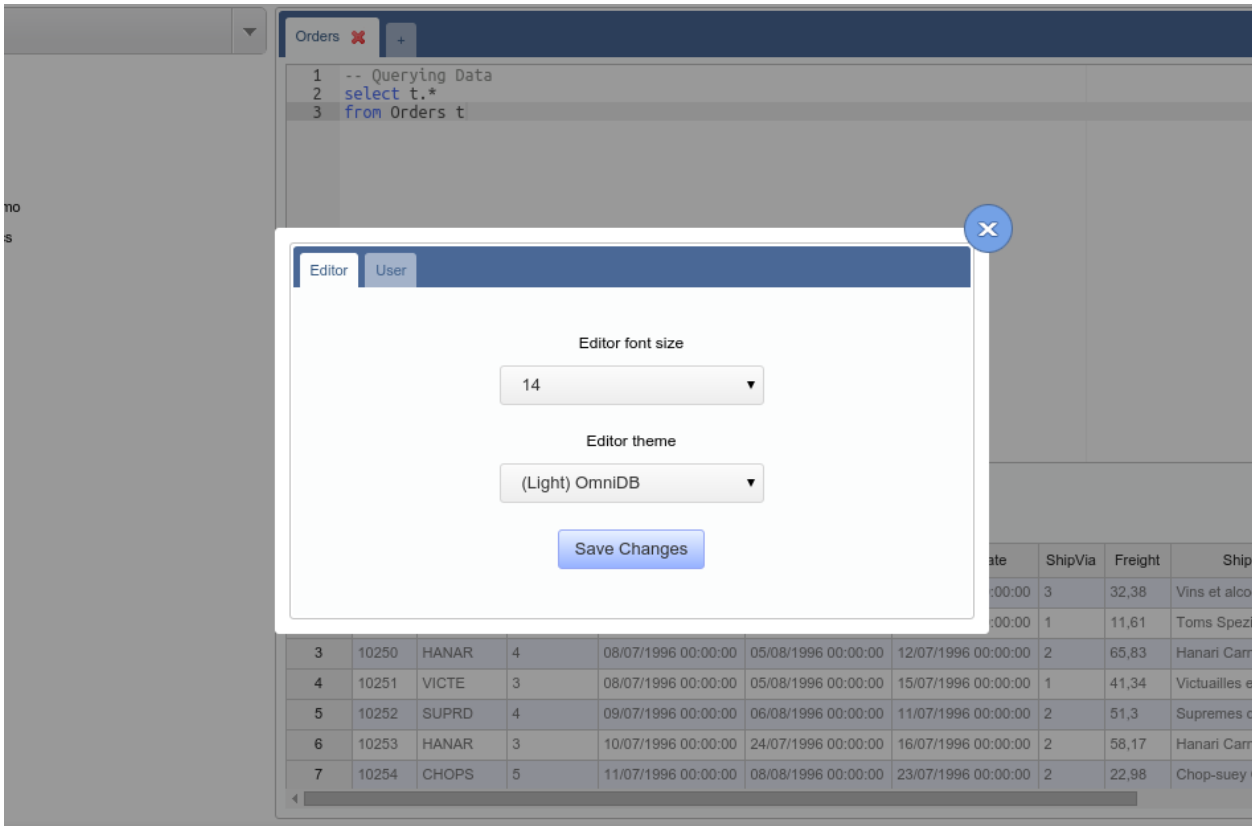
Number of records: 10
Response time: 0.012 seconds

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName	ShipAddres
1	10248	VINET	5	04/07/1996 00:00:00	01/08/1996 00:00:00	16/07/1996 00:00:00	3	32,38	Vins et alcools Chevalier	59 rue de l-Abbaye
2	10249	TOMSP	6	05/07/1996 00:00:00	16/08/1996 00:00:00	10/07/1996 00:00:00	1	11,61	Toms Spezialitaten	Luisenstr. 48
3	10250	HANAR	4	08/07/1996 00:00:00	05/08/1996 00:00:00	12/07/1996 00:00:00	2	65,83	Hanari Carnes	Rua do Paco, 67
4	10251	VICTE	3	08/07/1996 00:00:00	05/08/1996 00:00:00	15/07/1996 00:00:00	1	41,34	Victuailles en stock	2, rue du Commerce
5	10252	SUPRD	4	09/07/1996 00:00:00	06/08/1996 00:00:00	11/07/1996 00:00:00	2	51,3	Supremes delices	Boulevard Tirou, 255
6	10253	HANAR	3	10/07/1996 00:00:00	24/07/1996 00:00:00	16/07/1996 00:00:00	2	58,17	Hanari Carnes	Rua do Paco, 67
7	10254	CHOPS	5	11/07/1996 00:00:00	08/08/1996 00:00:00	23/07/1996 00:00:00	2	22,98	Chop-suey Chinese	Hauptstr. 31

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 chapter 8.

4.3. Changing User Settings

By clicking on the gear icon (top right corner of the screen) a popup with two tabs will show up. The first one lets you change editor settings and the other lets you change your password.



```
-- Querying Data
select t.*
from Orders t
```

Editor font size

14

Editor theme

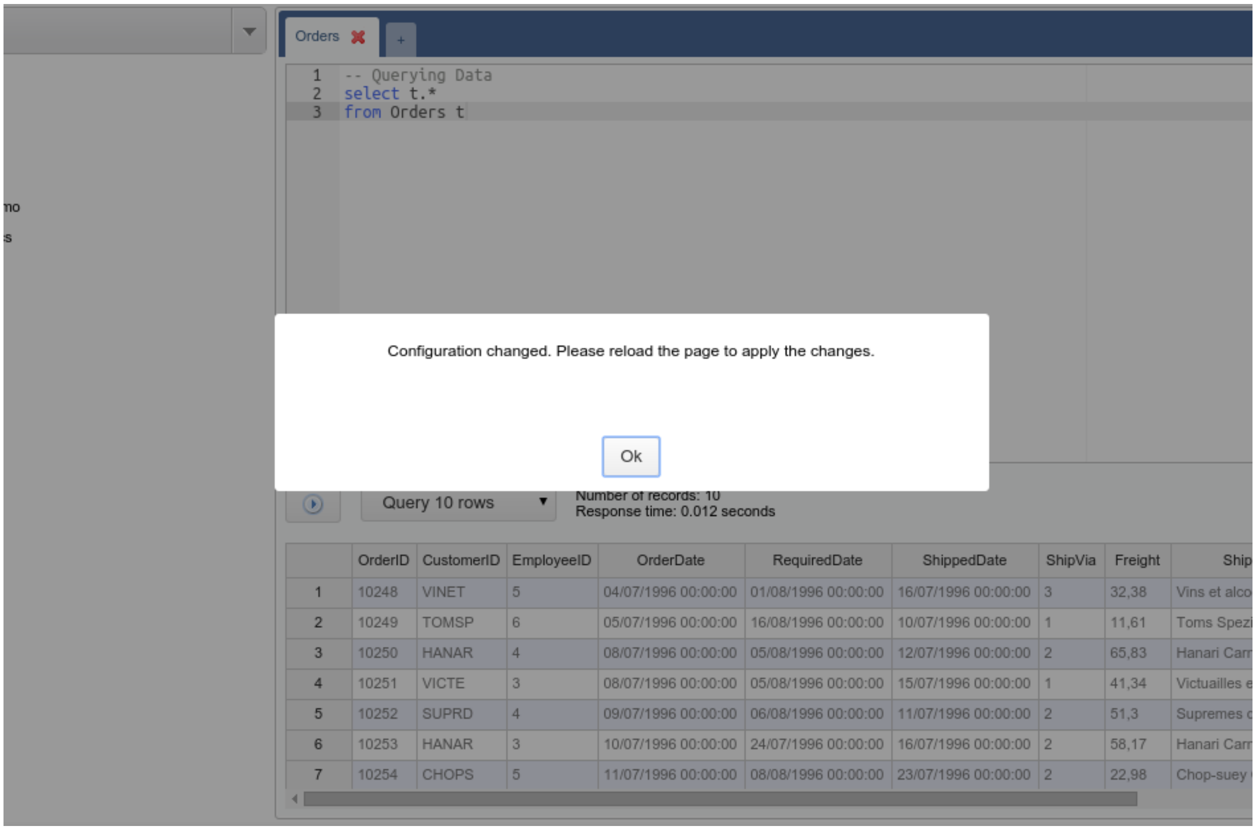
(Light) OmniDB

						ate	ShipVia	Freight	Ship
						:00:00	3	32,38	Vins et alco
						:00:00	1	11,61	Toms Spezi
3	10250	HANAR	4	08/07/1996 00:00:00	05/08/1996 00:00:00	12/07/1996 00:00:00	2	65,83	Hanari Carr
4	10251	VICTE	3	08/07/1996 00:00:00	05/08/1996 00:00:00	15/07/1996 00:00:00	1	41,34	Victuailles e
5	10252	SUPRD	4	09/07/1996 00:00:00	06/08/1996 00:00:00	11/07/1996 00:00:00	2	51,3	Supremes d
6	10253	HANAR	3	10/07/1996 00:00:00	24/07/1996 00:00:00	16/07/1996 00:00:00	2	58,17	Hanari Carr
7	10254	CHOPS	5	11/07/1996 00:00:00	08/08/1996 00:00:00	23/07/1996 00:00:00	2	22,98	Chop-suey

There are two options on the *editor* tab:

- *Editor font size*: Font size of all SQL editors. The default value is 14 but you can choose any value between 10 and 18;
- *Editor theme*: Color scheme of all SQL editors. OmniDB comes with a lot of different themes. They change background color, font and even syntax highlight colors.

Notice that when changing the settings of the tab editor, they will be applied after refreshing the page.



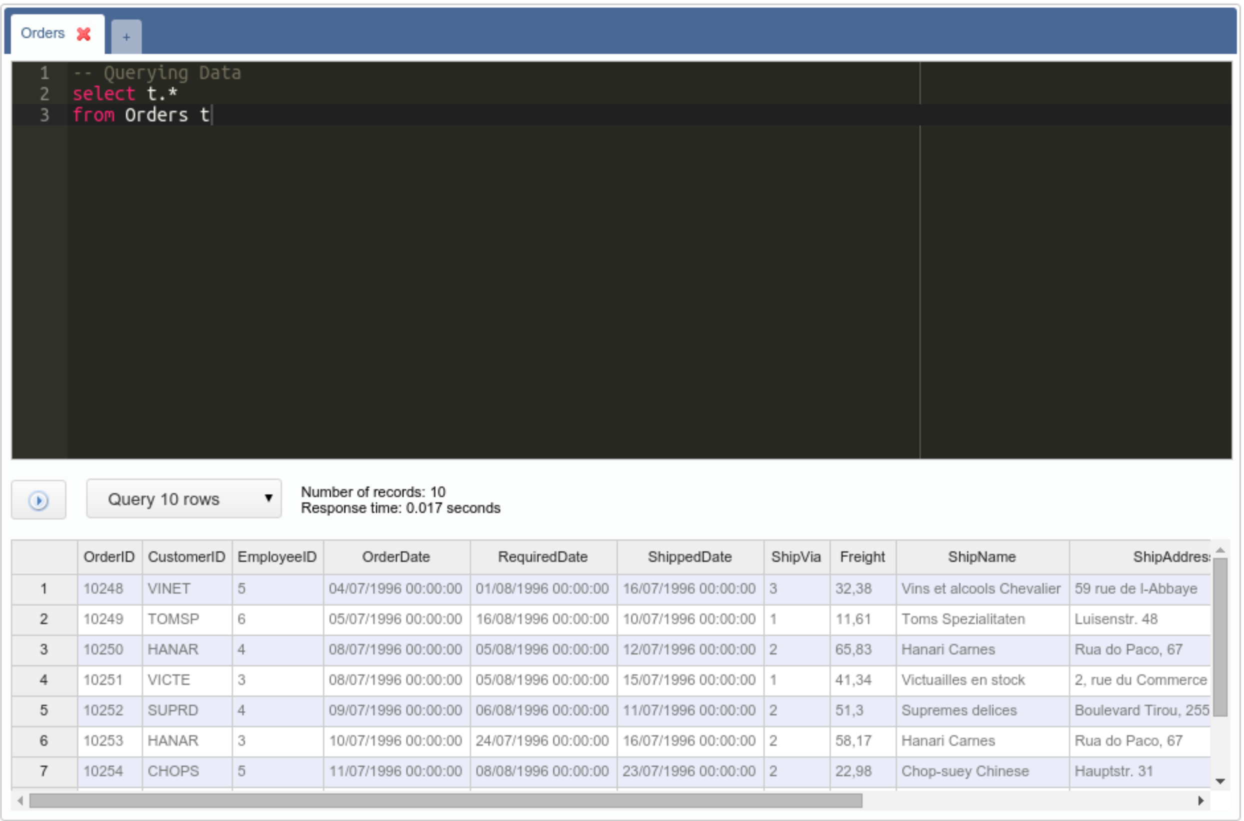
```
-- Querying Data
select t.*
from Orders t
```

Configuration changed. Please reload the page to apply the changes.

Number of records: 10
Response time: 0.012 seconds

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	Ship
1	10248	VINET	5	04/07/1996 00:00:00	01/08/1996 00:00:00	16/07/1996 00:00:00	3	32,38	Vins et alco
2	10249	TOMSP	6	05/07/1996 00:00:00	16/08/1996 00:00:00	10/07/1996 00:00:00	1	11,61	Toms Spezi
3	10250	HANAR	4	08/07/1996 00:00:00	05/08/1996 00:00:00	12/07/1996 00:00:00	2	65,83	Hanari Carn
4	10251	VICTE	3	08/07/1996 00:00:00	05/08/1996 00:00:00	15/07/1996 00:00:00	1	41,34	Victuailles e
5	10252	SUPRD	4	09/07/1996 00:00:00	06/08/1996 00:00:00	11/07/1996 00:00:00	2	51,3	Supremes d
6	10253	HANAR	3	10/07/1996 00:00:00	24/07/1996 00:00:00	16/07/1996 00:00:00	2	58,17	Hanari Carn
7	10254	CHOPS	5	11/07/1996 00:00:00	08/08/1996 00:00:00	23/07/1996 00:00:00	2	22,98	Chop-suey

By selecting font size 16 and theme *(Dark) Monokai*, your editor will have the following appearance:



```
-- Querying Data
select t.*
from Orders t
```

Number of records: 10
Response time: 0.017 seconds

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName	ShipAddres
1	10248	VINET	5	04/07/1996 00:00:00	01/08/1996 00:00:00	16/07/1996 00:00:00	3	32,38	Vins et alcools Chevalier	59 rue de l-Abbaye
2	10249	TOMSP	6	05/07/1996 00:00:00	16/08/1996 00:00:00	10/07/1996 00:00:00	1	11,61	Toms Spezialitaten	Luisenstr. 48
3	10250	HANAR	4	08/07/1996 00:00:00	05/08/1996 00:00:00	12/07/1996 00:00:00	2	65,83	Hanari Carnes	Rua do Paco, 67
4	10251	VICTE	3	08/07/1996 00:00:00	05/08/1996 00:00:00	15/07/1996 00:00:00	1	41,34	Victuailles en stock	2, rue du Commerce
5	10252	SUPRD	4	09/07/1996 00:00:00	06/08/1996 00:00:00	11/07/1996 00:00:00	2	51,3	Supremes delices	Boulevard Tirou, 255
6	10253	HANAR	3	10/07/1996 00:00:00	24/07/1996 00:00:00	16/07/1996 00:00:00	2	58,17	Hanari Carnes	Rua do Paco, 67
7	10254	CHOPS	5	11/07/1996 00:00:00	08/08/1996 00:00:00	23/07/1996 00:00:00	2	22,98	Chop-suey Chinese	Hauptstr. 31

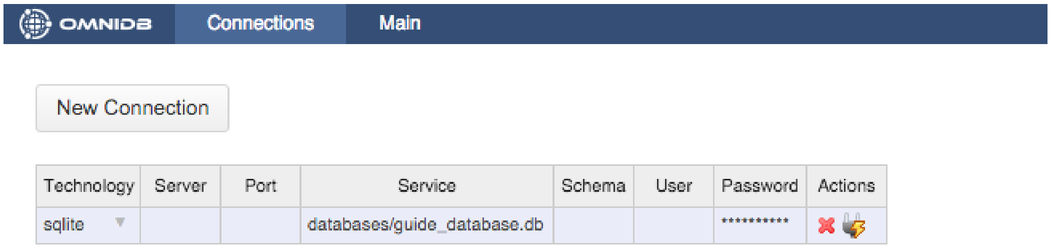
5. Table Creation

OmniDB has a table creation interface that lets you configure columns, constraints and indexes. A couple 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.

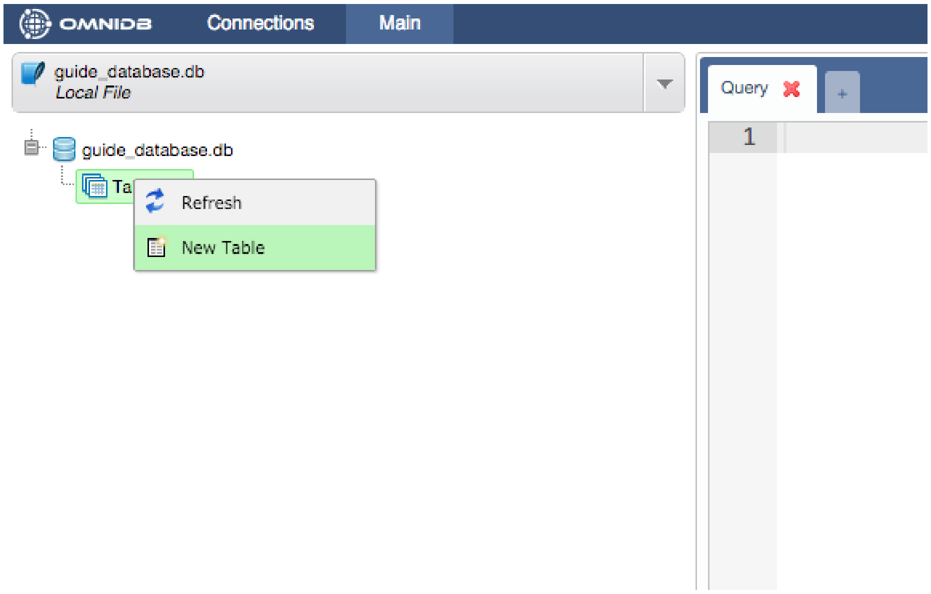
5.1. Creating tables

We will create example tables (Customer and Address) in a SQLite database that will also be created. To create the database, go to the connections window and create a new connection.



Technology	Server	Port	Service	Schema	User	Password	Actions
sqlite ▼			databases/guide_database.db			**********	✖

Go back to the main window, 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:

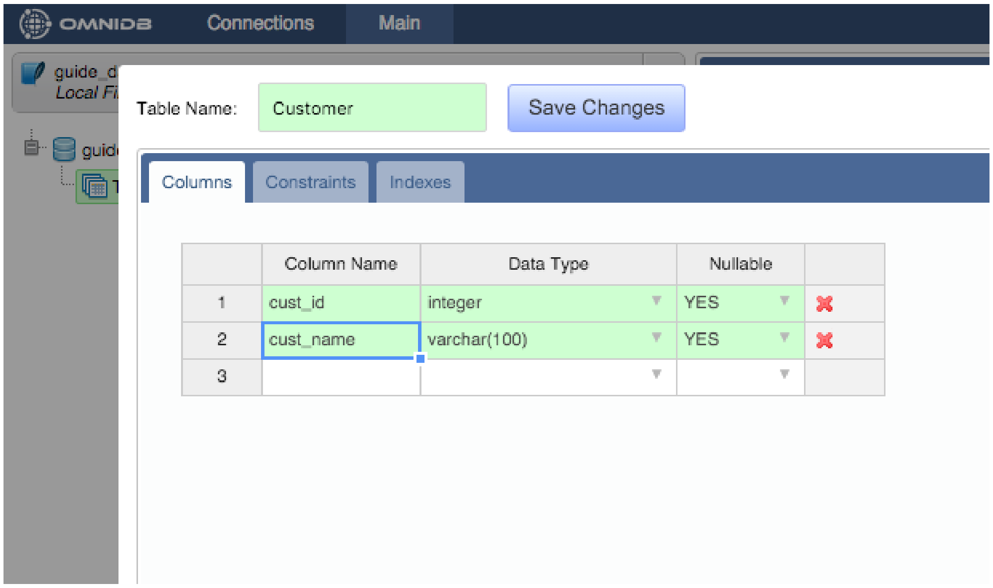


Table Name: Customer

	Column Name	Data Type	Nullable	
1	cust_id	integer	YES	✖
2	cust_name	varchar(100)	YES	✖
3				

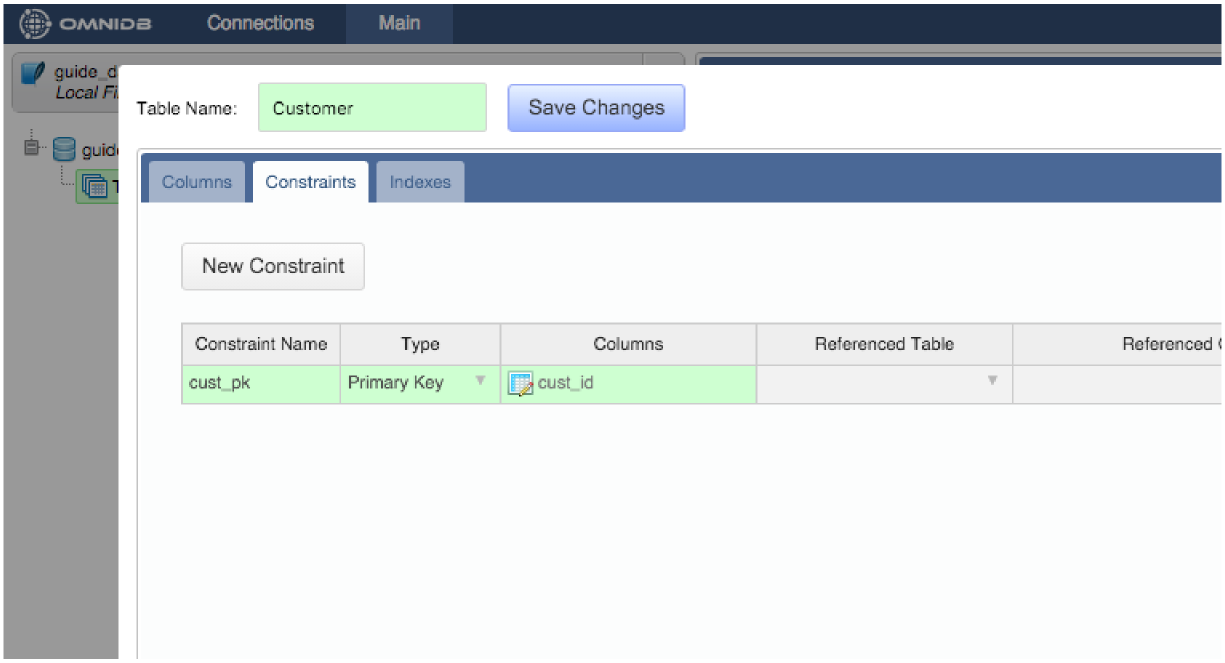


Table Name: Customer

Constraint Name	Type	Columns	Referenced Table	Referenced (
cust_pk	Primary Key	cust_id		

And then, create the table Address with a primary key and a foreign key:

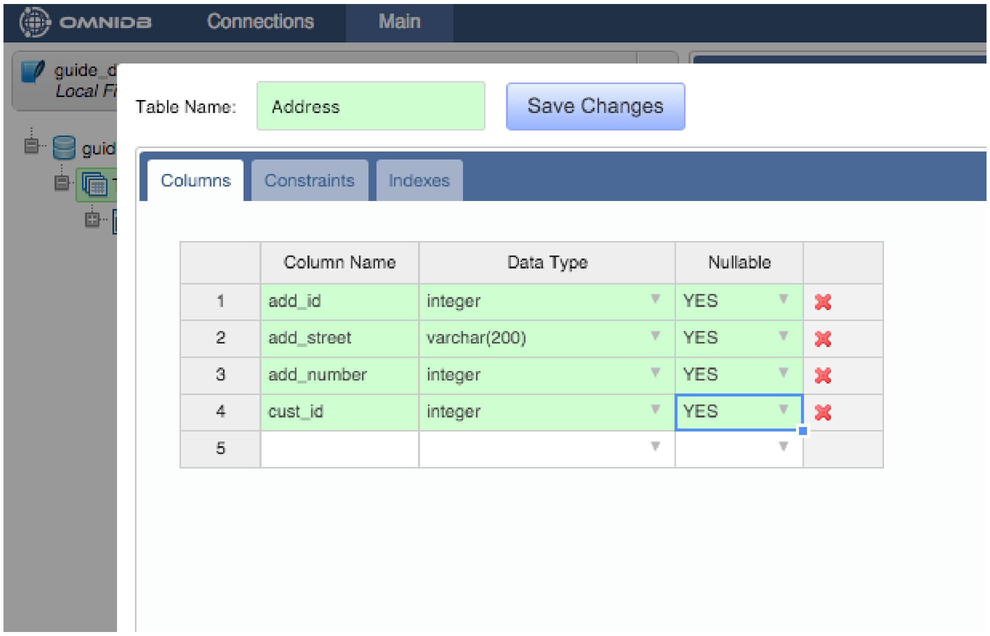


Table Name: Address

	Column Name	Data Type	Nullable	
1	add_id	integer	YES	✖
2	add_street	varchar(200)	YES	✖
3	add_number	integer	YES	✖
4	cust_id	integer	YES	✖
5				

OmniDB automatically searches for primary key and unique constraints when selecting a table during foreign key creation:

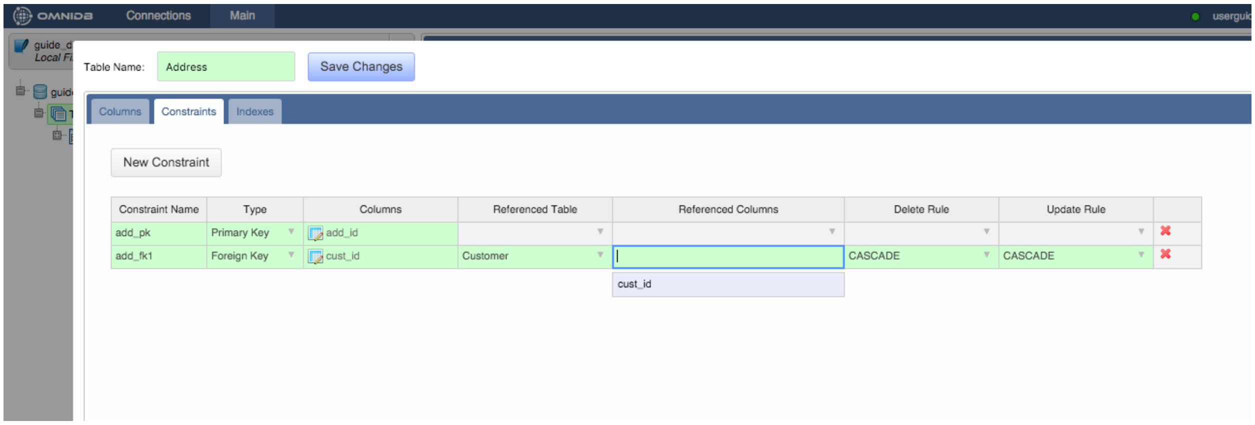
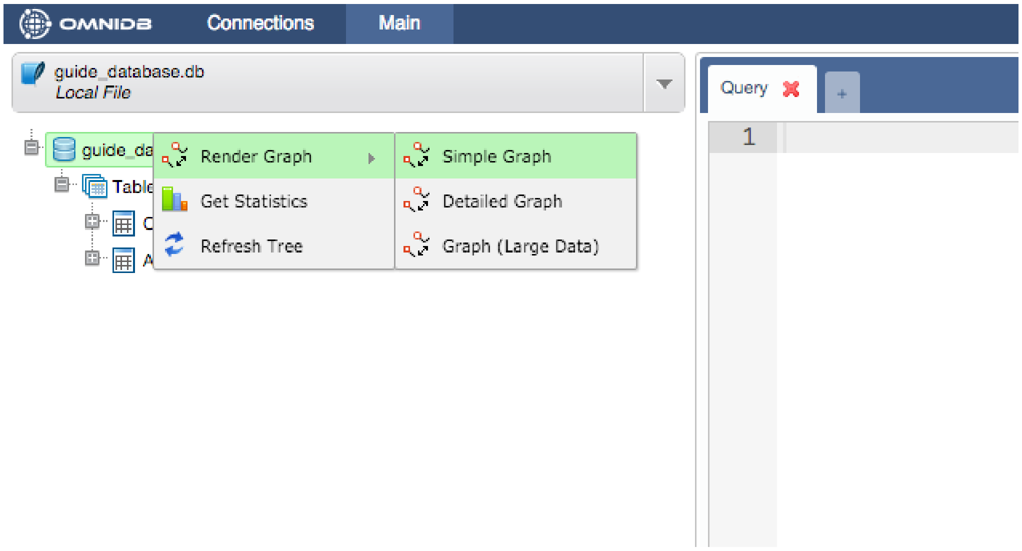


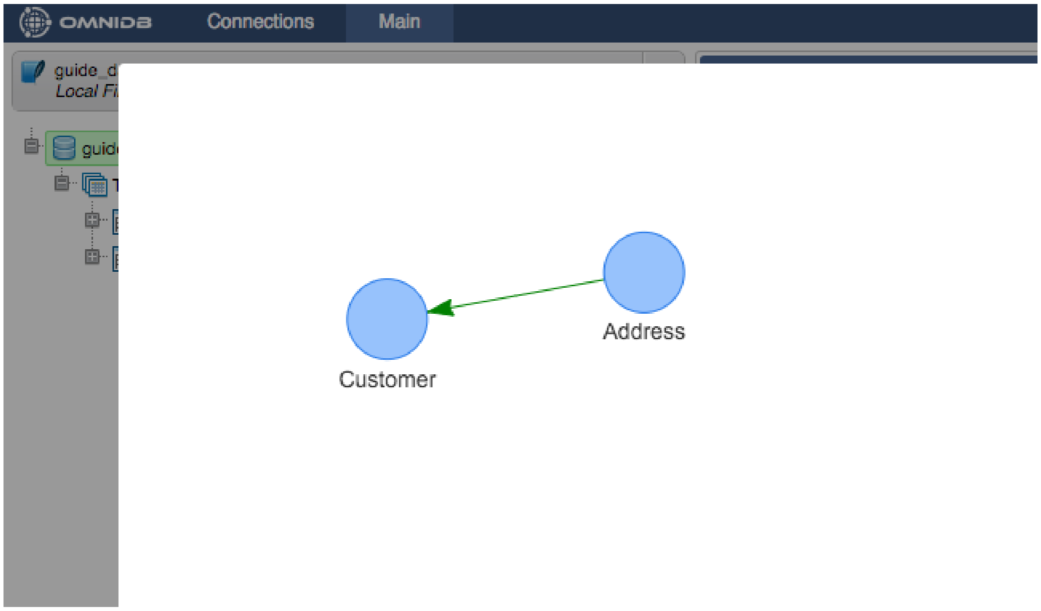
Table Name: Address

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

cust_id

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

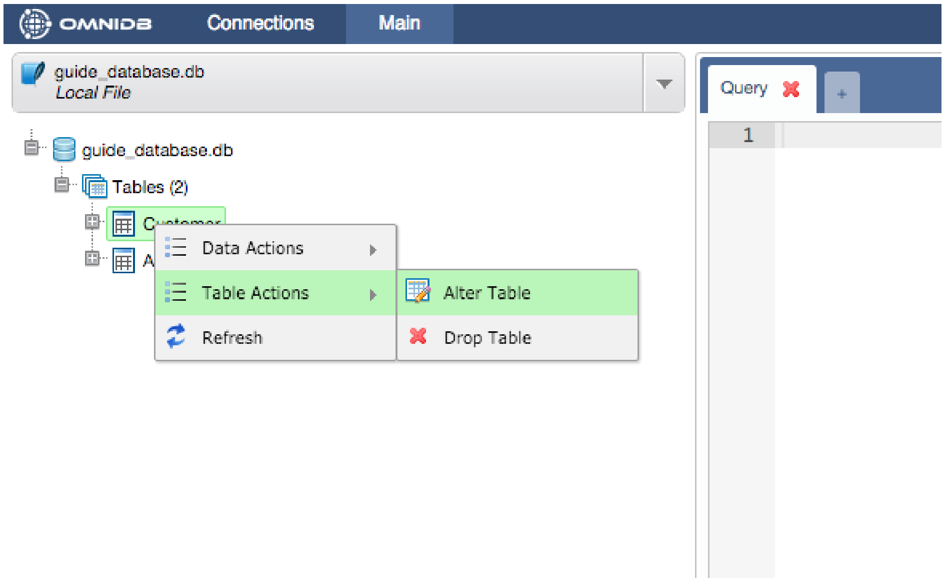



OMNIDB
Connections
Main
guide_d
Local Fi
guid
T
Customer
Address

6. Altering and Removing Tables

6.1. 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, created on the last chapter. 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:

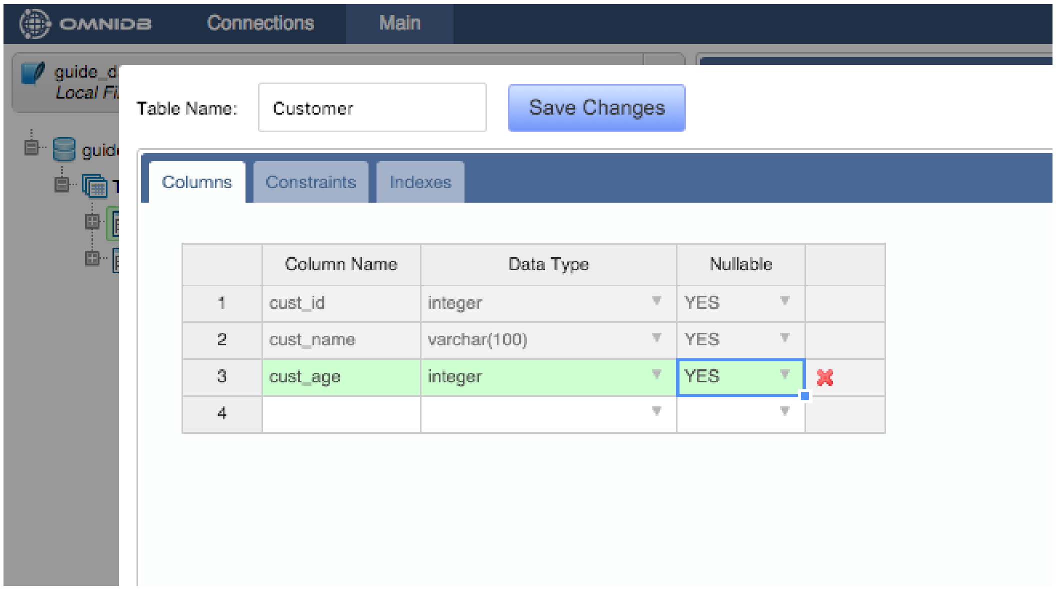
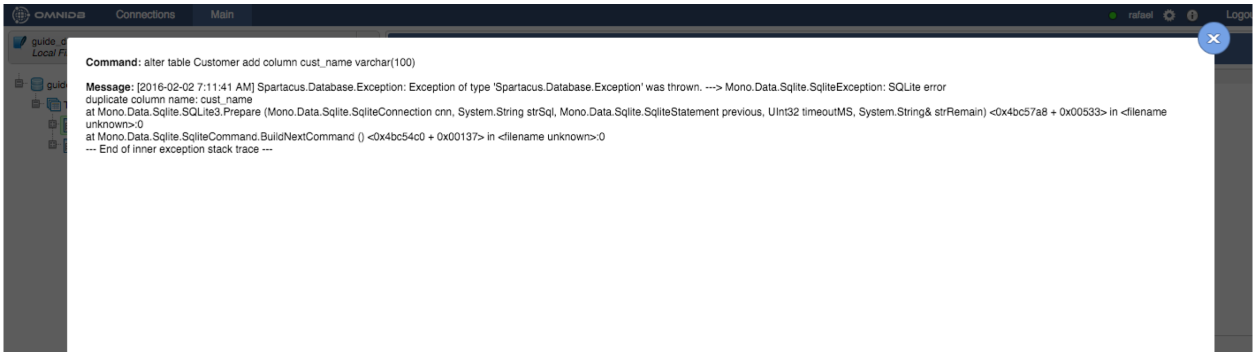


Table Name: Customer

	Column Name	Data Type	Nullable	
1	cust_id	integer	YES	
2	cust_name	varchar(100)	YES	
3	cust_age	integer	YES	✖
4				

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



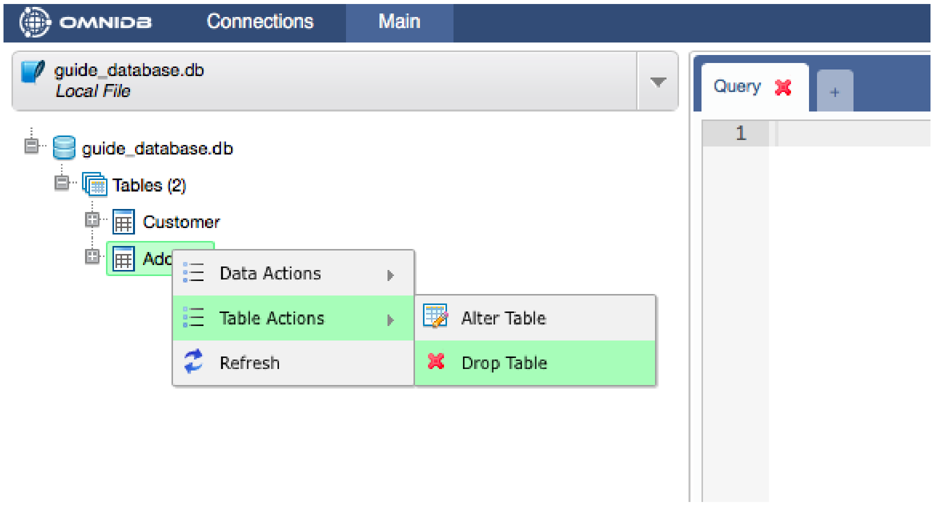
Command: alter table Customer add column cust_name varchar(100)

```
Message: [2016-02-02 7:11:41 AM] Spartacus.Database.Exception: Exception of type 'Spartacus.Database.Exception' was thrown. ---> Mono.Data.Sqlite.SqliteException: SQLite error
duplicate column name: cust_name
at Mono.Data.Sqlite.SQLite3.Prepare (Mono.Data.Sqlite.SqliteConnection cnn, System.String strSql, Mono.Data.Sqlite.SqliteStatement previous, UInt32 timeoutMS, System.String& strRemain) <0x4bc57a8 + 0x00533> in <filename
unknown>:0
at Mono.Data.Sqlite.SqliteCommand.BuildNextCommand () <0x4bc54c0 + 0x00137> in <filename unknown>:0
--- End of inner exception stack trace ---
```

The interface indicates the command and the error that occurred.

6.2. Removing Tables

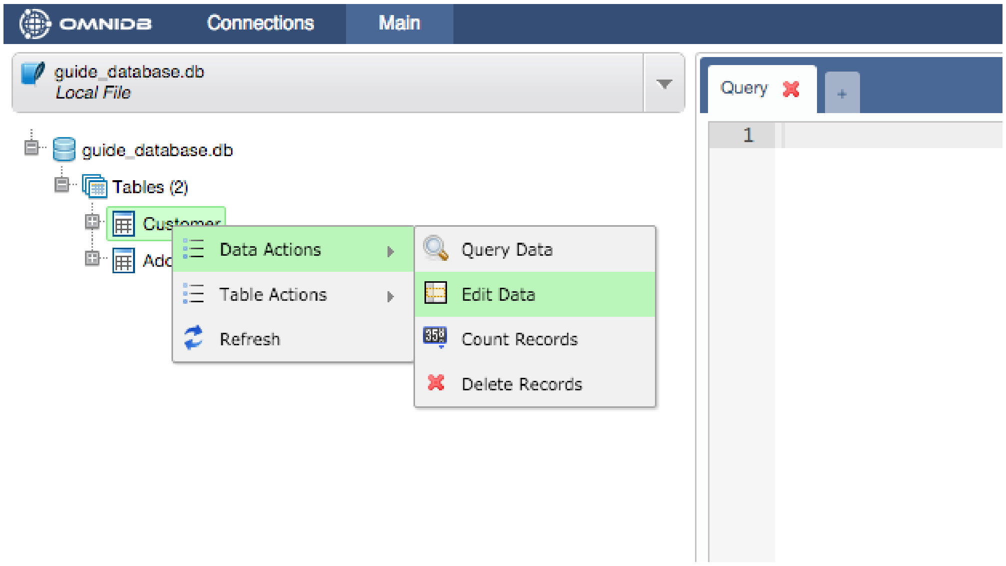
In order to remove a table just right click the table node and select the action *Table Actions > Drop Table*:

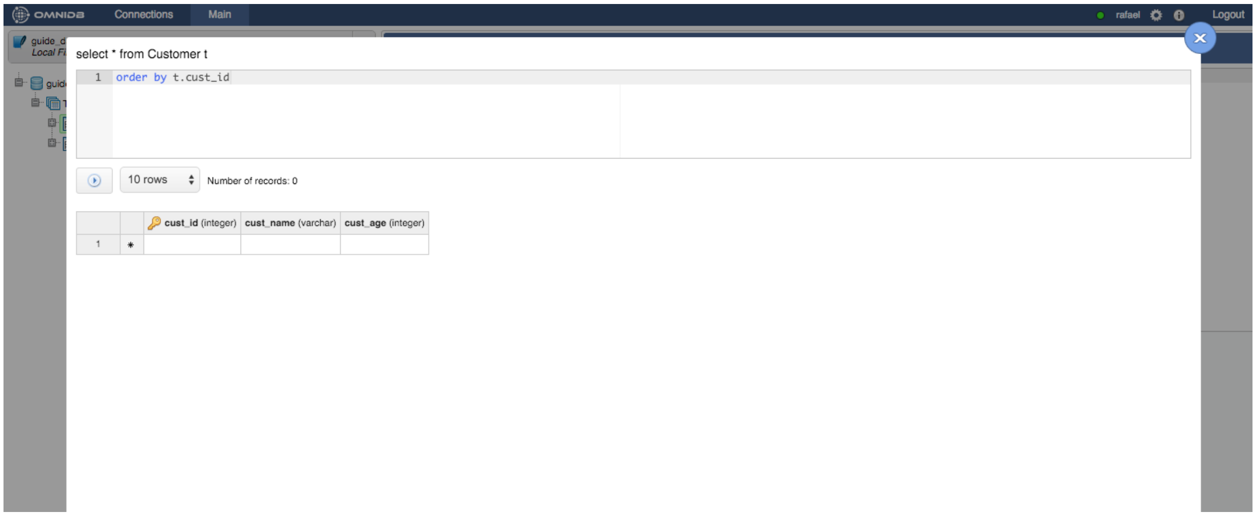

OMNIDB
Connections
Main
guide_database.db
Local File
Query
+
1
guide_database.db
Tables (2)
Customer
Add
Data Actions
Table Actions
Refresh
Alter Table
Drop Table

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 Customer t
order by t.cust_id
```

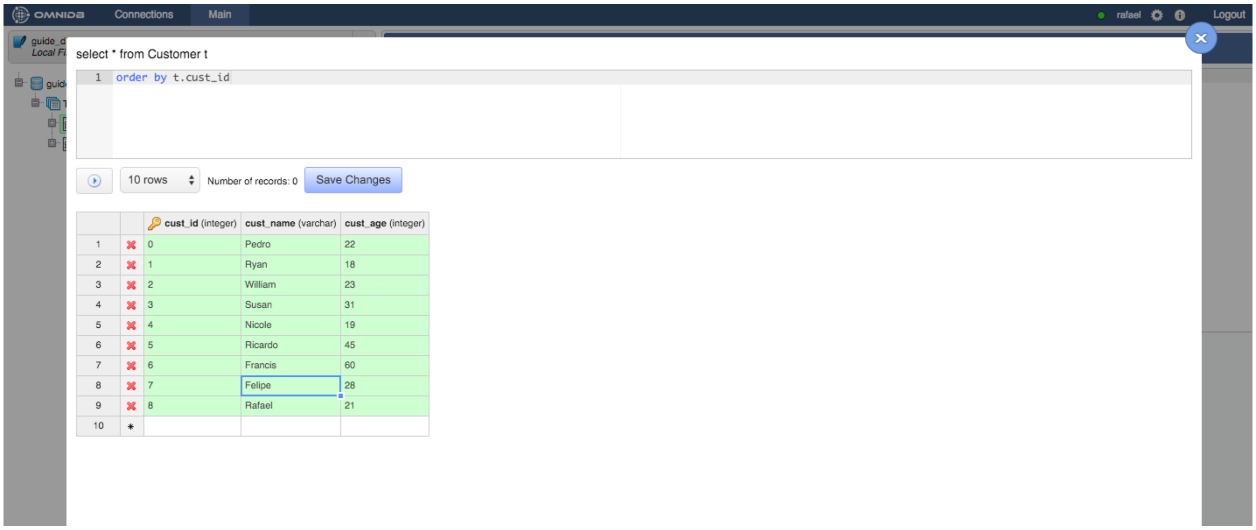
10 rows

Number of records: 0

		cust_id (integer)	cust_name (varchar)	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:

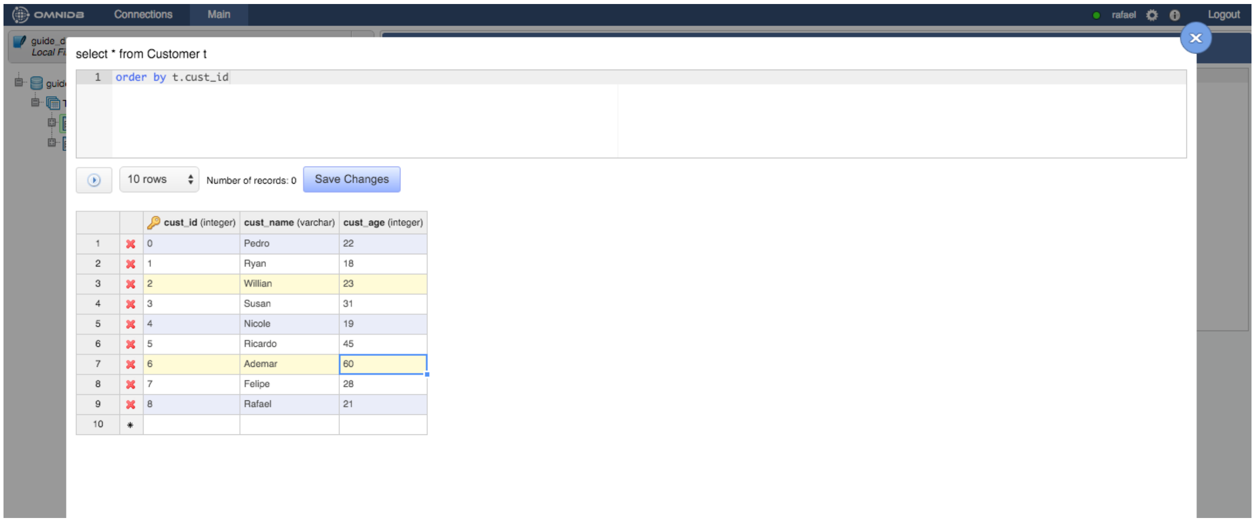


select * from Customer t

```
order by t.cust_id
```

		cust_id (integer)	cust_name (varchar)	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	Francis	60
8	✖	7	Felipe	28
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:

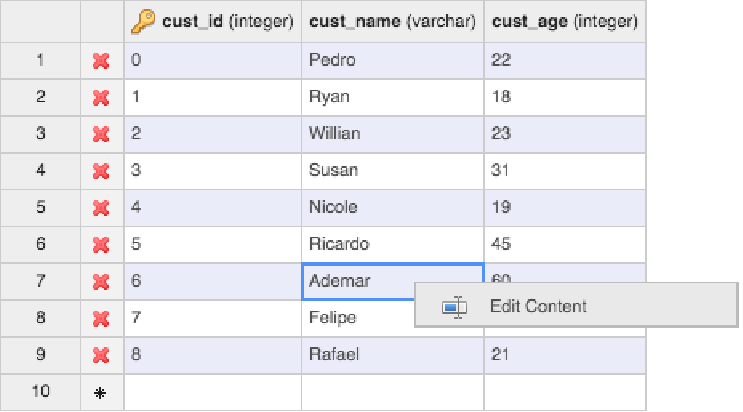


select * from Customer t

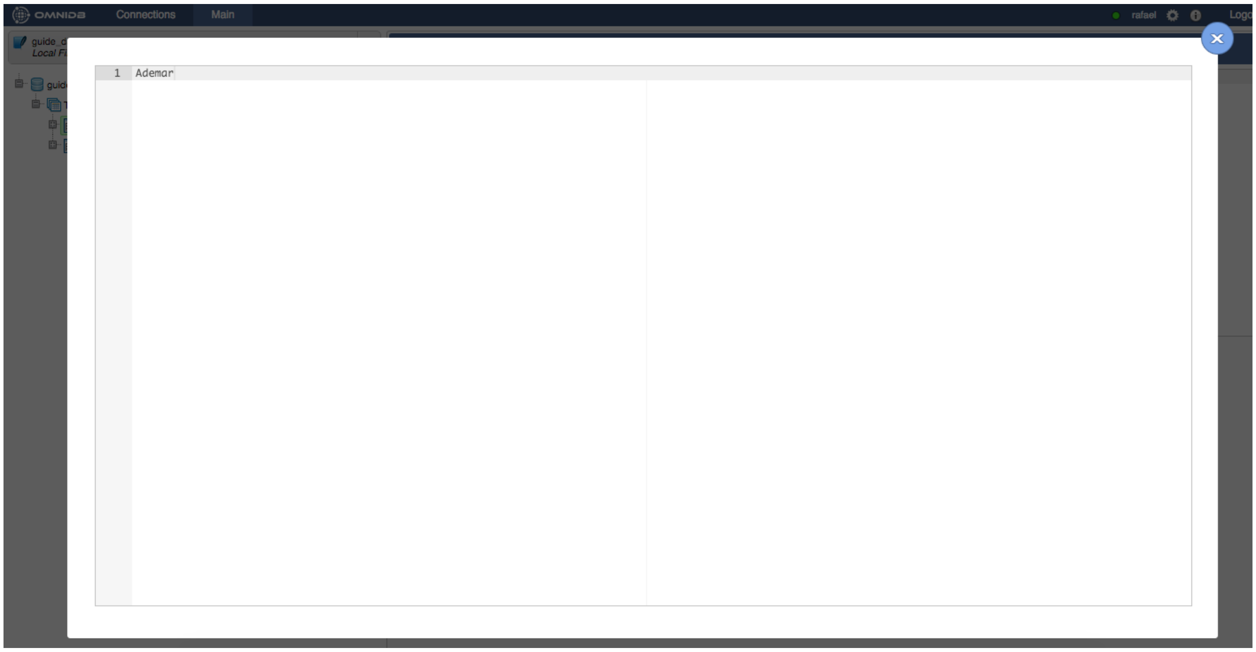
```
order by t.cust_id
```

		cust_id (integer)	cust_name (varchar)	cust_age (integer)
1	✖	0	Pedro	22
2	✖	1	Ryan	18
3	✖	2	Willian	23
4	✖	3	Susan	31
5	✖	4	Nicole	19
6	✖	5	Ricardo	45
7	✖	6	Ademar	60
8	✖	7	Felipe	28
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:

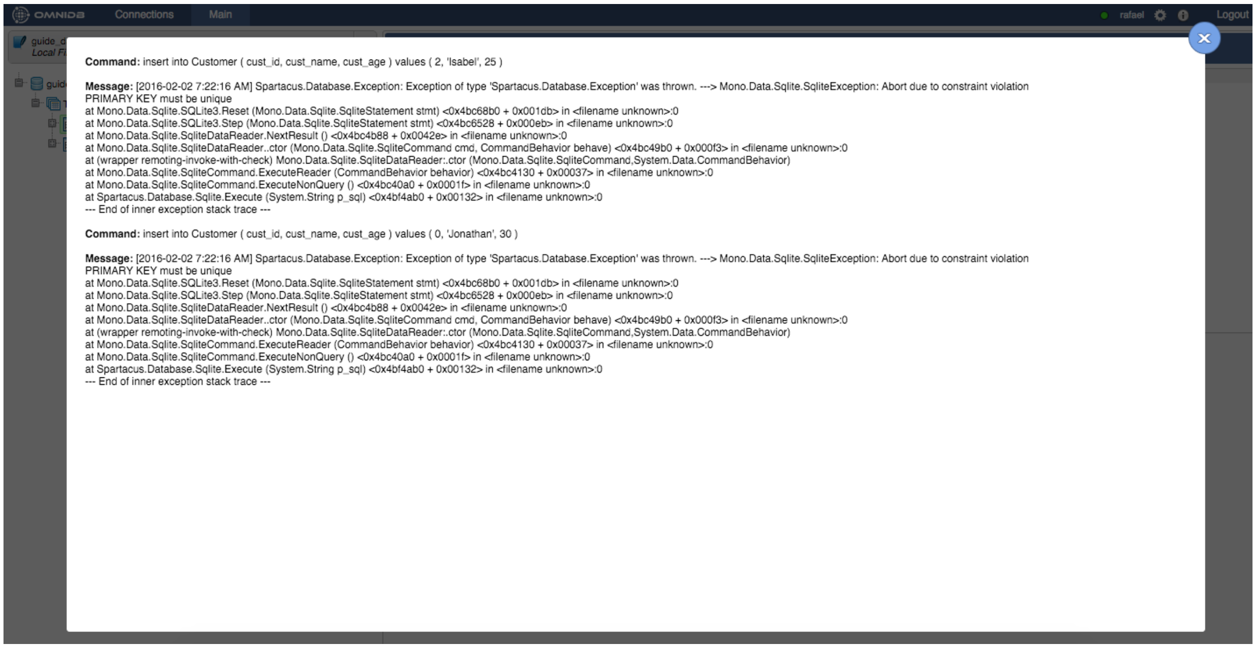


		cust_id (integer)	cust_name (varchar)	cust_age (integer)
1	✖	0	Pedro	22
2	✖	1	Ryan	18
3	✖	2	Willian	23
4	✖	3	Susan	31
5	✖	4	Nicole	19
6	✖	5	Ricardo	45
7	✖	6	Ademar	60
8	✖	7	Felipe	
9	✖	8	Rafael	21
10	*			



```
Ademar
```

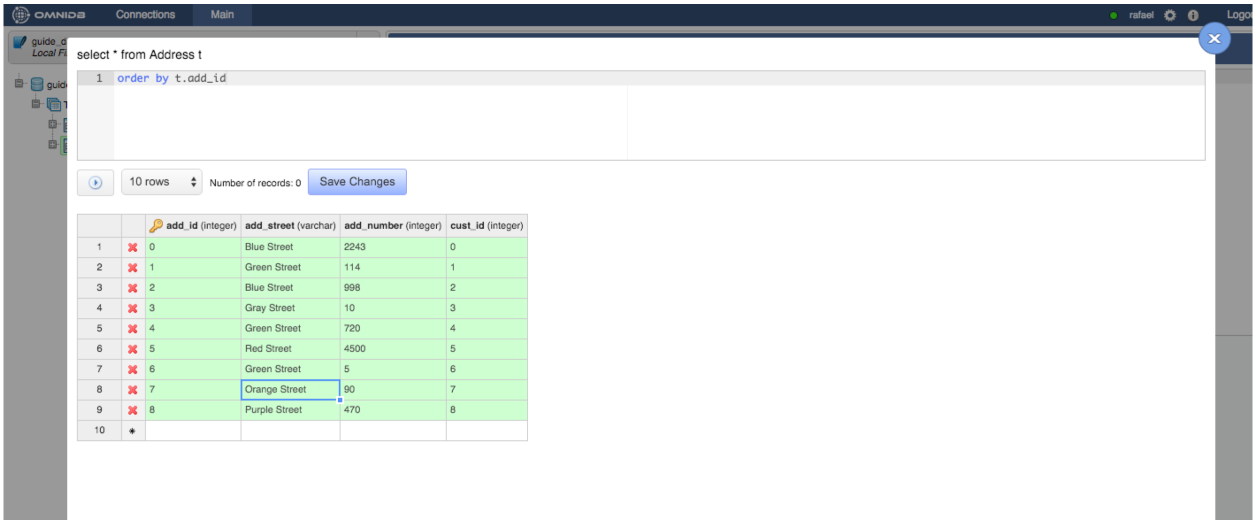
The interface detects errors that may occur during operations related to records. To demonstrate, let us insert two records with existing cust_id (primary key):



```
Command: insert into Customer ( cust_id, cust_name, cust_age ) values ( 2, 'Isabel', 25 )

Message: [2016-02-02 7:22:16 AM] Spartacus.Database.Exception: Exception of type 'Spartacus.Database.Exception' was thrown. ---> Mono.Data.Sqlite.SqliteException: Abort due to constraint violation
PRIMARY KEY must be unique
at Mono.Data.Sqlite.SQLite3.Reset (Mono.Data.Sqlite.SqliteStatement stmt) <0x4bc68b0 + 0x001db> in <filename unknown>:0
at Mono.Data.Sqlite.SQLite3.Step (Mono.Data.Sqlite.SqliteStatement stmt) <0x4bc6528 + 0x000eb> in <filename unknown>:0
at Mono.Data.Sqlite.SqliteDataReader.NextResult () <0x4bc4b88 + 0x0042e> in <filename unknown>:0
at Mono.Data.Sqlite.SqliteDataReader..ctor (Mono.Data.Sqlite.SqliteCommand cmd, CommandBehavior behave) <0x4bc49b0 + 0x000f3> in <filename unknown>:0
at (wrapper remoting-invoke-with-check) Mono.Data.Sqlite.SqliteDataReader:.ctor (Mono.Data.Sqlite.SqliteCommand,System.Data.CommandBehavior)
at Mono.Data.Sqlite.SqliteCommand.ExecuteReader (CommandBehavior behavior) <0x4bc4130 + 0x00037> in <filename unknown>:0
at Mono.Data.Sqlite.SqliteCommand.ExecuteNonQuery () <0x4bc40a0 + 0x0001f> in <filename unknown>:0
at Spartacus.Database.Sqlite.Execute (System.String p_sql) <0x4bf4ab0 + 0x00132> in <filename unknown>:0
--- End of inner exception stack trace ---

Command: insert into Customer ( cust_id, cust_name, cust_age ) values ( 0, 'Jonathan', 30 )

Message: [2016-02-02 7:22:16 AM] Spartacus.Database.Exception: Exception of type 'Spartacus.Database.Exception' was thrown. ---> Mono.Data.Sqlite.SqliteException: Abort due to constraint violation
PRIMARY KEY must be unique
at Mono.Data.Sqlite.SQLite3.Reset (Mono.Data.Sqlite.SqliteStatement stmt) <0x4bc68b0 + 0x001db> in <filename unknown>:0
at Mono.Data.Sqlite.SQLite3.Step (Mono.Data.Sqlite.SqliteStatement stmt) <0x4bc6528 + 0x000eb> in <filename unknown>:0
at Mono.Data.Sqlite.SqliteDataReader.NextResult () <0x4bc4b88 + 0x0042e> in <filename unknown>:0
at Mono.Data.Sqlite.SqliteDataReader..ctor (Mono.Data.Sqlite.SqliteCommand cmd, CommandBehavior behave) <0x4bc49b0 + 0x000f3> in <filename unknown>:0
at (wrapper remoting-invoke-with-check) Mono.Data.Sqlite.SqliteDataReader:.ctor (Mono.Data.Sqlite.SqliteCommand,System.Data.CommandBehavior)
at Mono.Data.Sqlite.SqliteCommand.ExecuteReader (CommandBehavior behavior) <0x4bc4130 + 0x00037> in <filename unknown>:0
at Mono.Data.Sqlite.SqliteCommand.ExecuteNonQuery () <0x4bc40a0 + 0x0001f> in <filename unknown>:0
at Spartacus.Database.Sqlite.Execute (System.String p_sql) <0x4bf4ab0 + 0x00132> in <filename unknown>:0
--- End of inner exception stack trace ---
```

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

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



```
select * from Address t
order by t.add_id
```

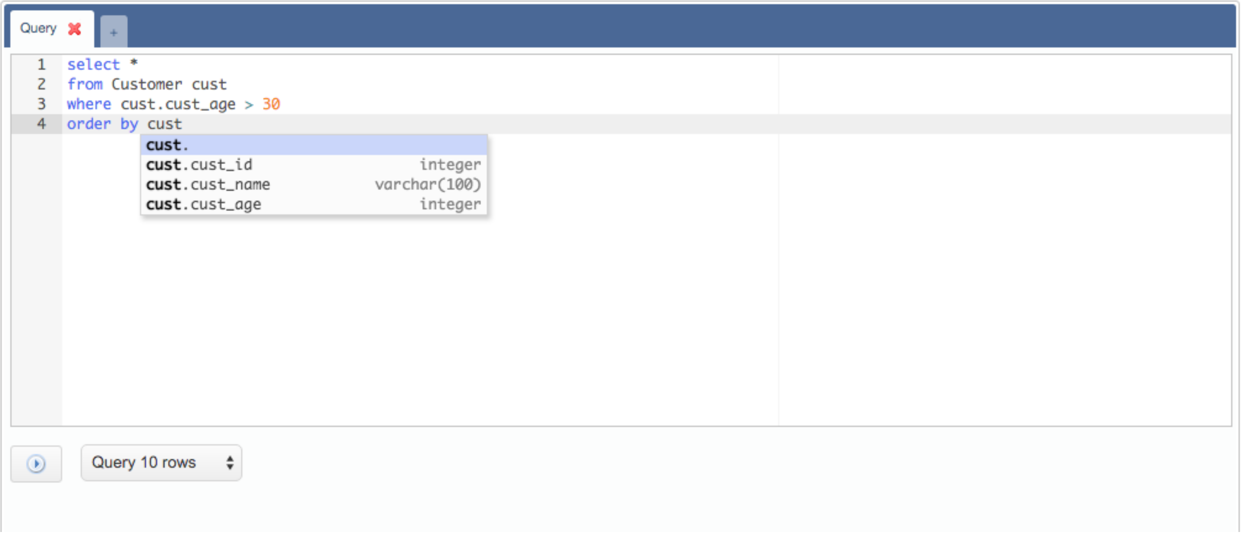
Number of records: 0

		add_id (integer)	add_street (varchar)	add_number (integer)	cust_id (integer)
1	✗	0	Blue Street	2243	0
2	✗	1	Green Street	114	1
3	✗	2	Blue Street	998	2
4	✗	3	Gray Street	10	3
5	✗	4	Green Street	720	4
6	✗	5	Red Street	4500	5
7	✗	6	Green Street	5	6
8	✗	7	Orange Street	90	7
9	✗	8	Purple Street	470	8
10	*				

8. SQL Editor

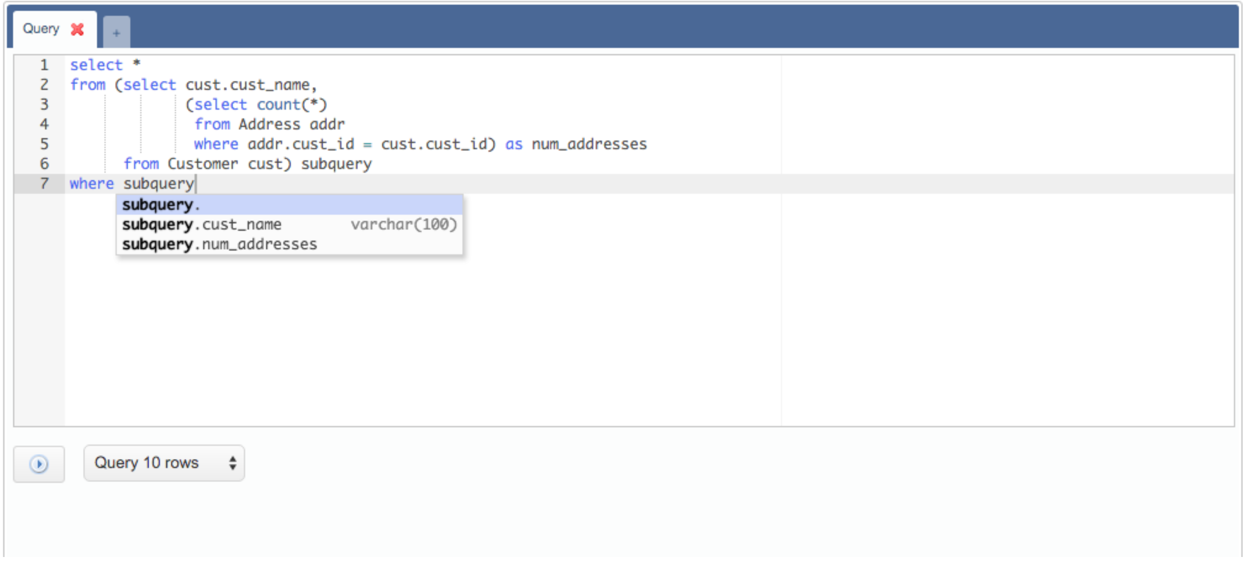
The tool comes with a tab system where each tab contains a SQL editor, an action button, a field to select the type of command and a space to display the result.

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
```

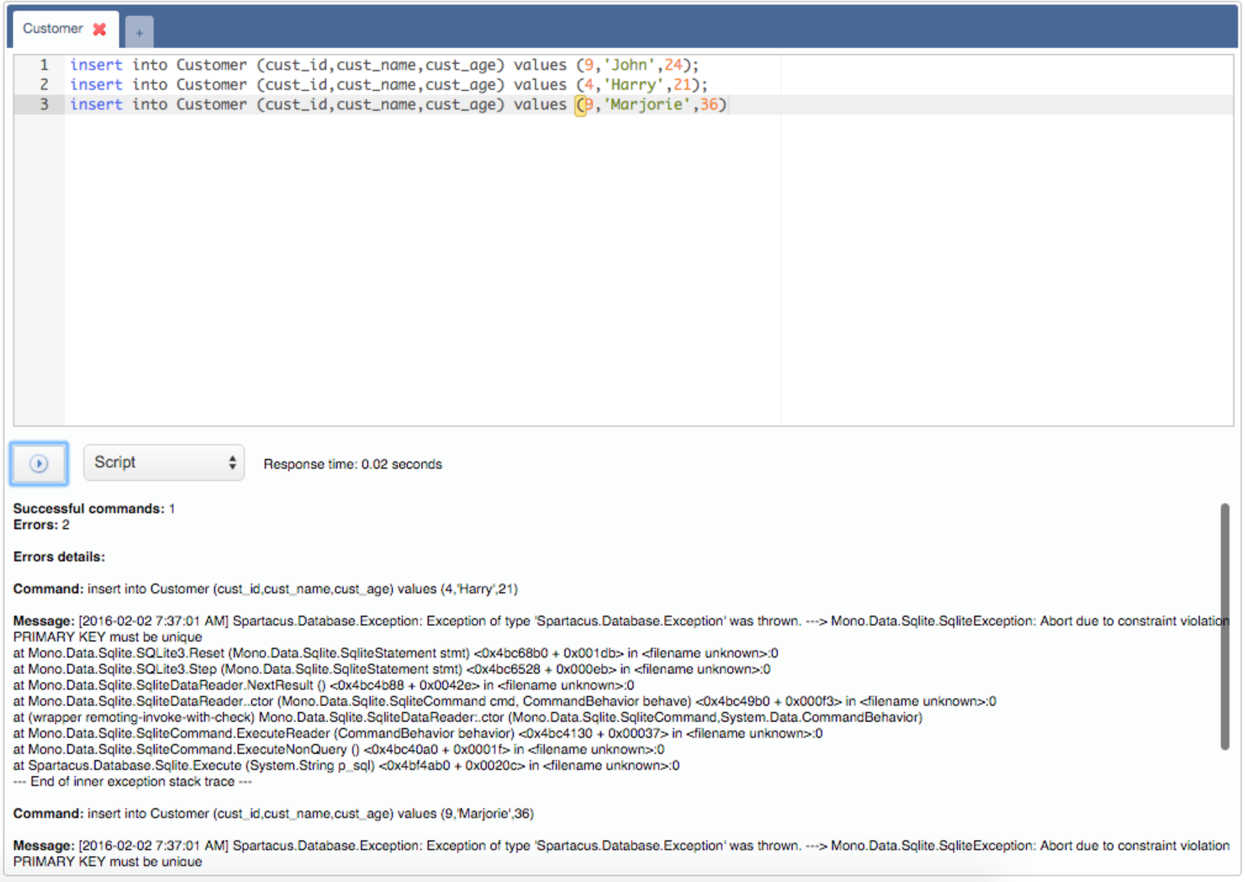
Besides autocompleting table columns the editor also searches for columns contained in subqueries:



```
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
```

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

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

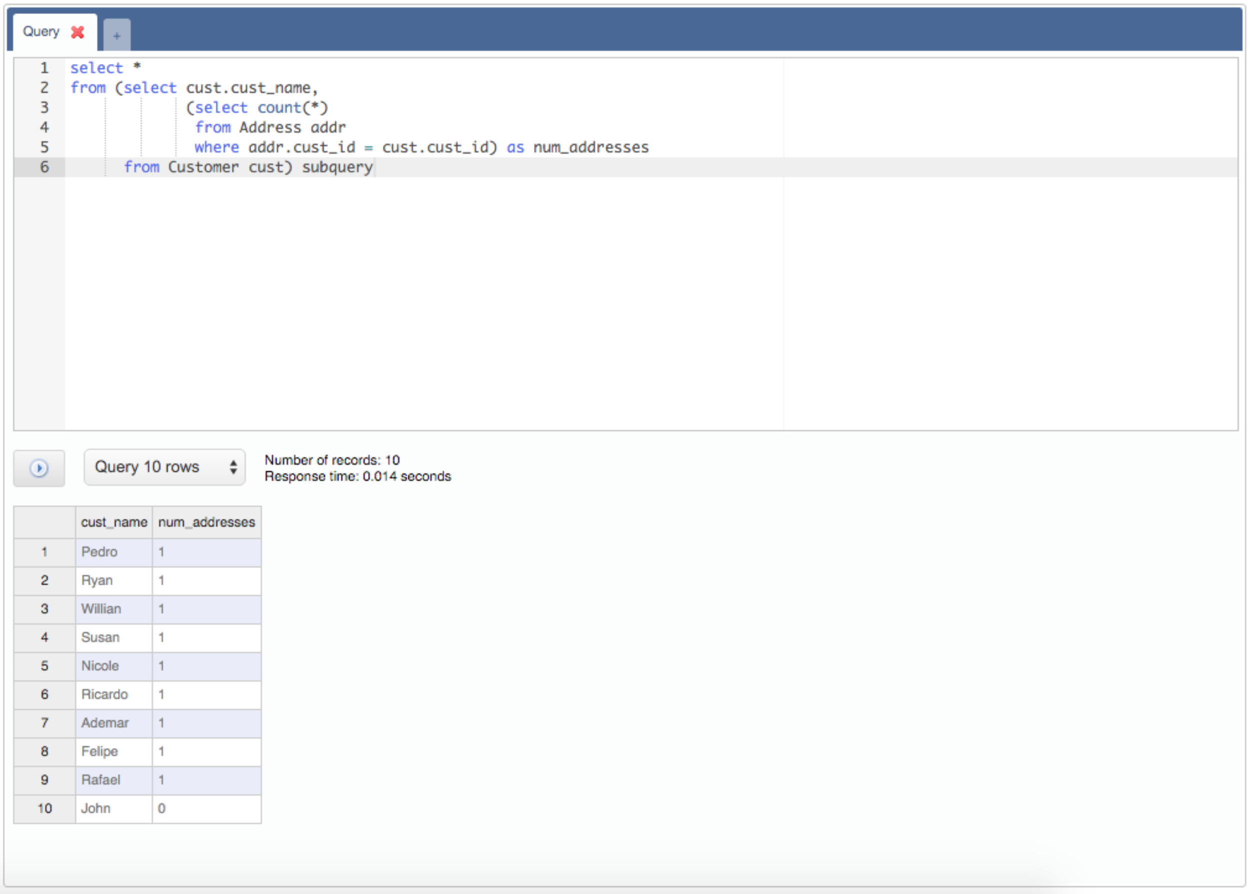


```sql
insert into Customer (cust_id,cust_name,cust_age) values (9,'John',24);
insert into Customer (cust_id,cust_name,cust_age) values (4,'Harry',21);
insert into Customer (cust_id,cust_name,cust_age) values (9,'Marjorie',36)
```

```
Response time: 0.02 seconds

Successful commands: 1
Errors: 2

Errors details:

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

Message: [2016-02-02 7:37:01 AM] Spartacus.Database.Exception: Exception of type 'Spartacus.Database.Exception' was thrown. ---> Mono.Data.Sqlite.SqliteException: Abort due to constraint violation
PRIMARY KEY must be unique
at Mono.Data.Sqlite.SQLite3.Reset (Mono.Data.Sqlite.SqliteStatement stmt) <0x4bc68b0 + 0x001db> in <filename unknown>:0
at Mono.Data.Sqlite.SQLite3.Step (Mono.Data.Sqlite.SqliteStatement stmt) <0x4bc6528 + 0x000eb> in <filename unknown>:0
at Mono.Data.Sqlite.SqliteDataReader.NextResult () <0x4bc4b88 + 0x0042e> in <filename unknown>:0
at Mono.Data.Sqlite.SqliteDataReader..ctor (Mono.Data.Sqlite.SqliteCommand cmd, CommandBehavior behave) <0x4bc49b0 + 0x000f3> in <filename unknown>:0
at (wrapper remoting-invoke-with-check) Mono.Data.Sqlite.SqliteDataReader:.ctor (Mono.Data.Sqlite.SqliteCommand,System.Data.CommandBehavior)
at Mono.Data.Sqlite.SqliteCommand.ExecuteReader (CommandBehavior behavior) <0x4bc4130 + 0x00037> in <filename unknown>:0
at Mono.Data.Sqlite.SqliteCommand.ExecuteNonQuery () <0x4bc40a0 + 0x0001f> in <filename unknown>:0
at Spartacus.Database.Sqlite.Execute (System.String p_sql) <0x4bf4ab0 + 0x0020c> in <filename unknown>:0
--- End of inner exception stack trace ---

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

Message: [2016-02-02 7:37:01 AM] Spartacus.Database.Exception: Exception of type 'Spartacus.Database.Exception' was thrown. ---> Mono.Data.Sqlite.SqliteException: Abort due to constraint violation
PRIMARY KEY must be unique
```

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: 10
Response time: 0.014 seconds

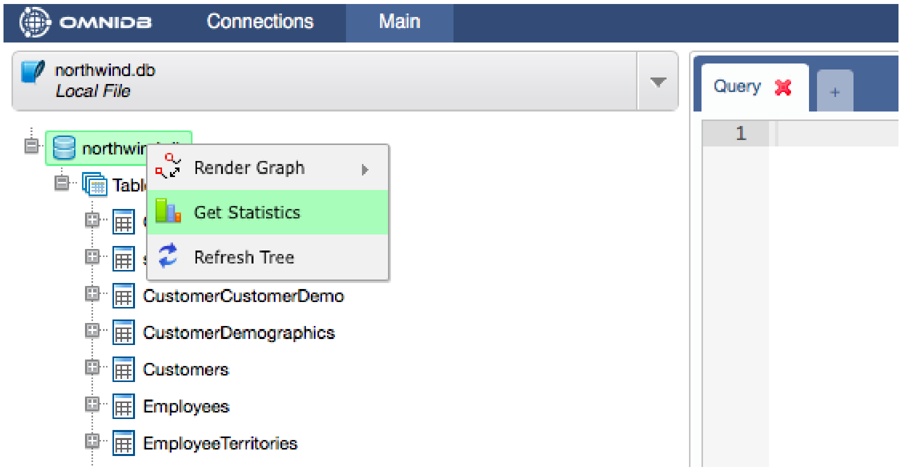
	cust_name	num_addresses
1	Pedro	1
2	Ryan	1
3	Willian	1
4	Susan	1
5	Nicole	1
6	Ricardo	1
7	Ademar	1
8	Felipe	1
9	Rafael	1
10	John	0

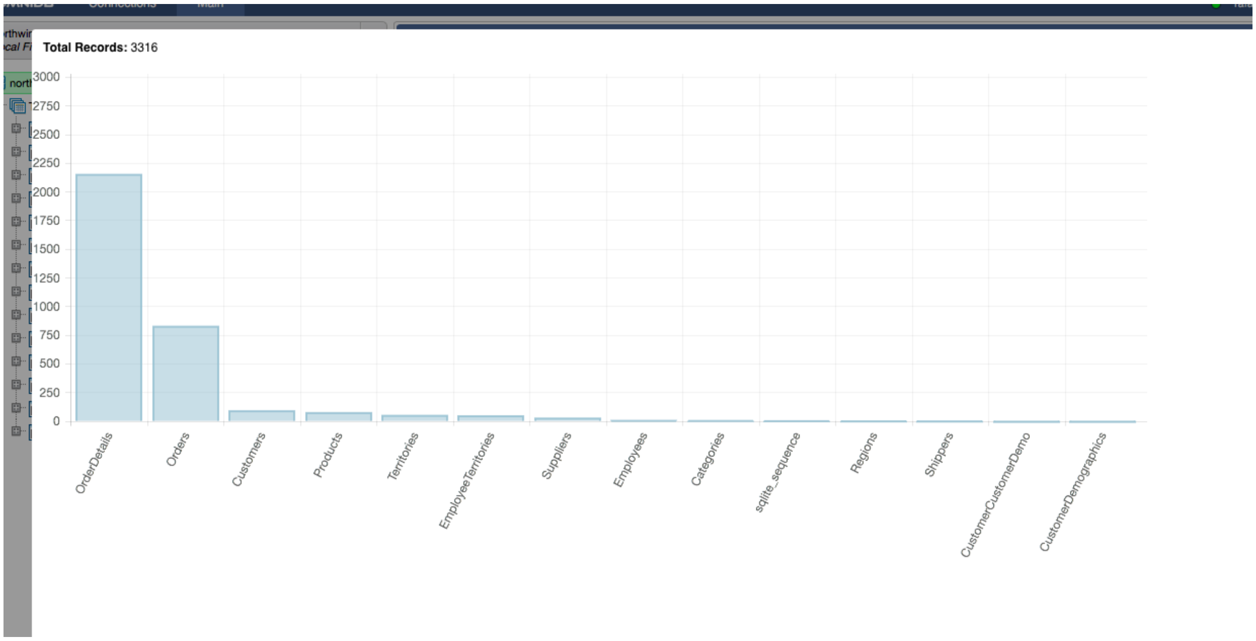
9. Additional Features

OmniDB provides other features to display useful information about the whole schema/database.

9.1. Bar chart with record count

This feature displays a bar chart with tables on the X axis and record count on the Y axis. To access it just right click the root node of the tree and then select the action *Get Statistics*:





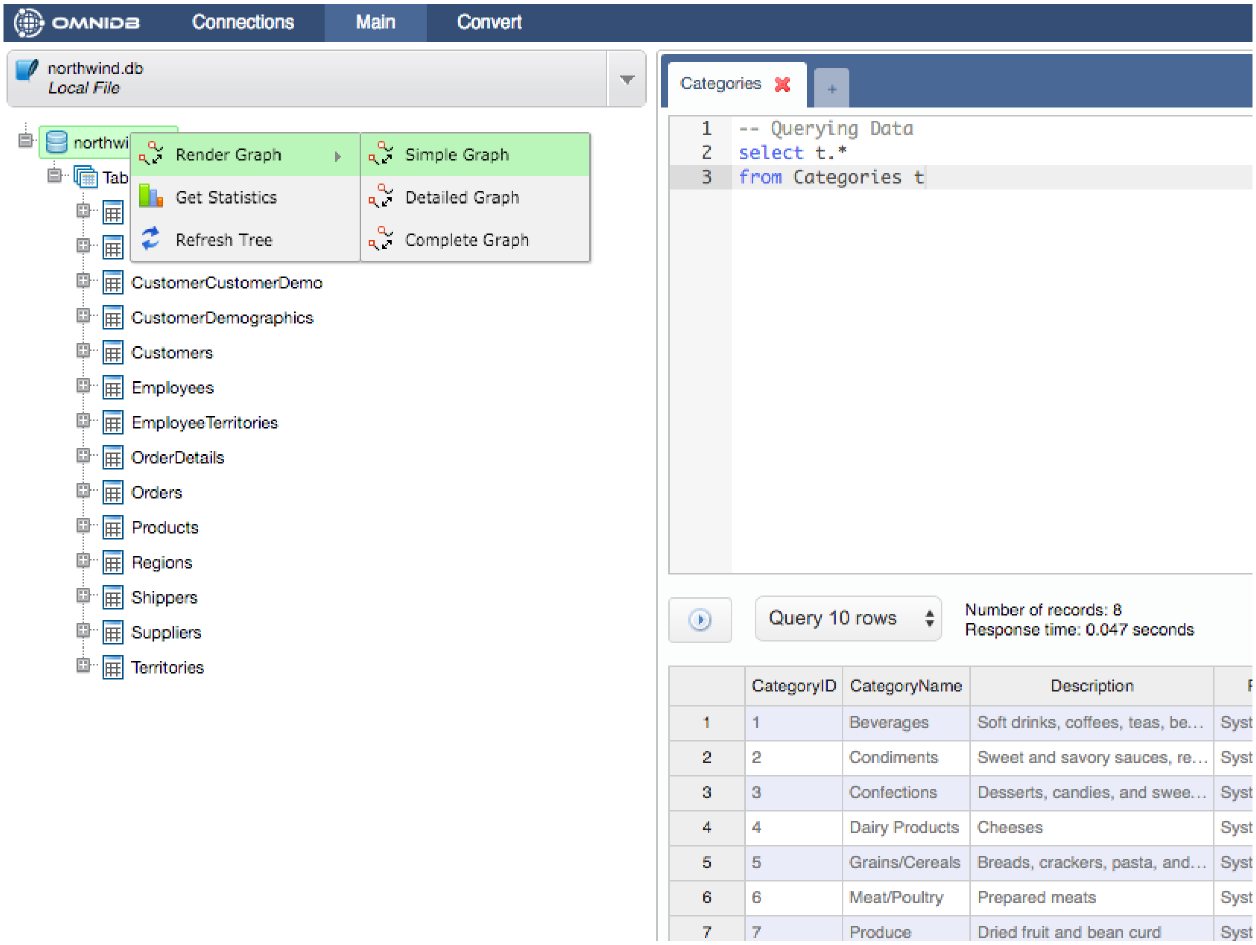
9.2. Graph with tables and relations

This feature displays a graph with nodes representing tables and edges representing table relationships with foreign keys.

There are two types of graphs: Simple Graph and Detailed Graph.

9.2.1. Simple Graph

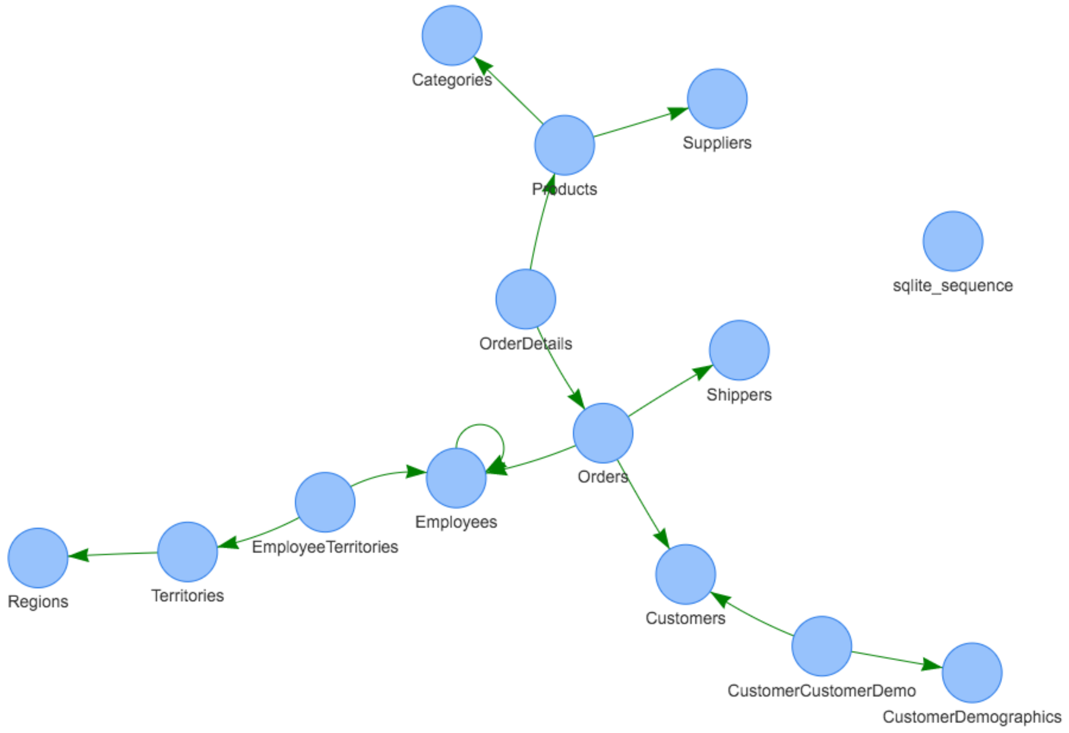
To access it just right click the root node of the tree and then select the action *Render Graph > Simple Graph*:



```
-- Querying Data
select t.*
from Categories t
```

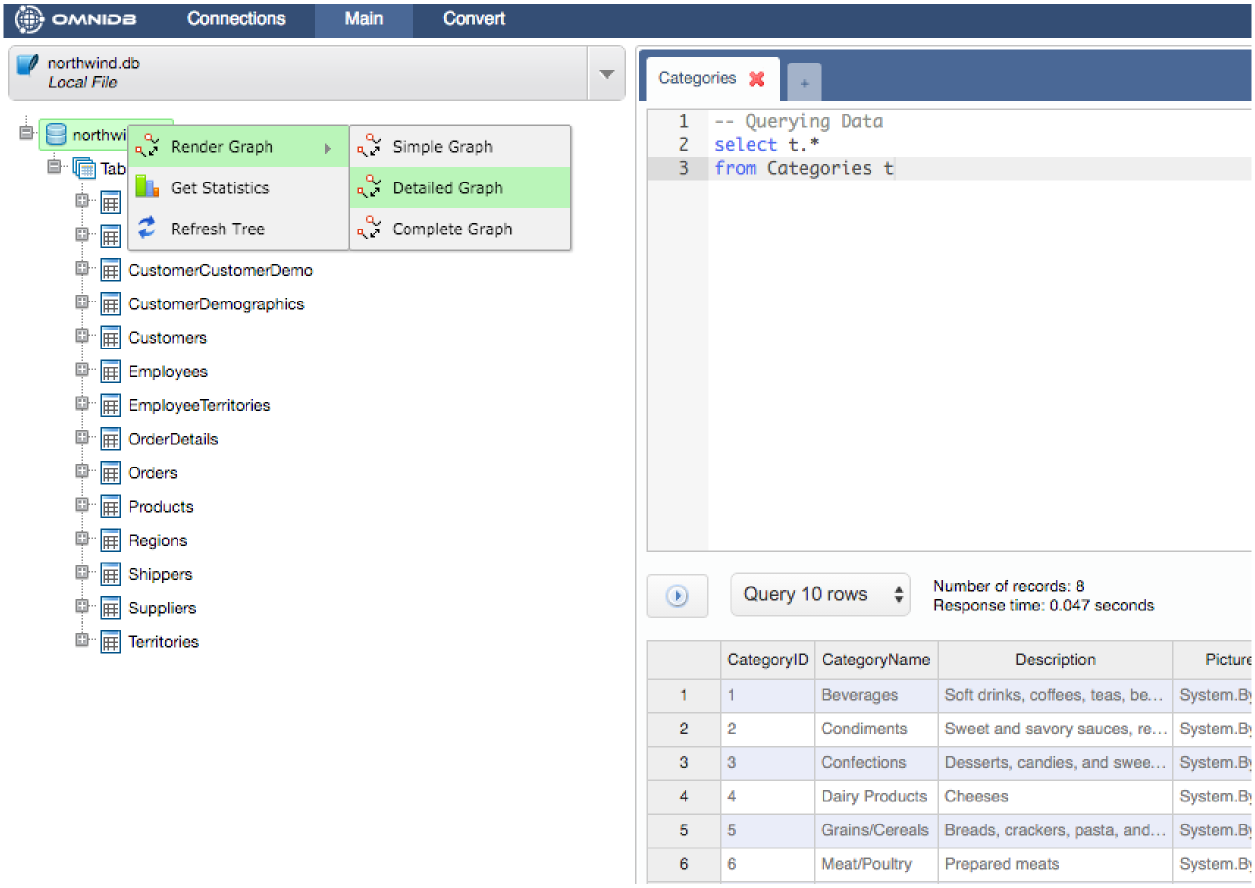
```
Number of records: 8
Response time: 0.047 seconds
```

	CategoryID	CategoryName	Description	F
1	1	Beverages	Soft drinks, coffees, teas, be…	Syst
2	2	Condiments	Sweet and savory sauces, re…	Syst
3	3	Confections	Desserts, candies, and swee…	Syst
4	4	Dairy Products	Cheeses	Syst
5	5	Grains/Cereals	Breads, crackers, pasta, and…	Syst
6	6	Meat/Poultry	Prepared meats	Syst
7	7	Produce	Dried fruit and bean curd	Syst



9.2.2. Detailed Graph

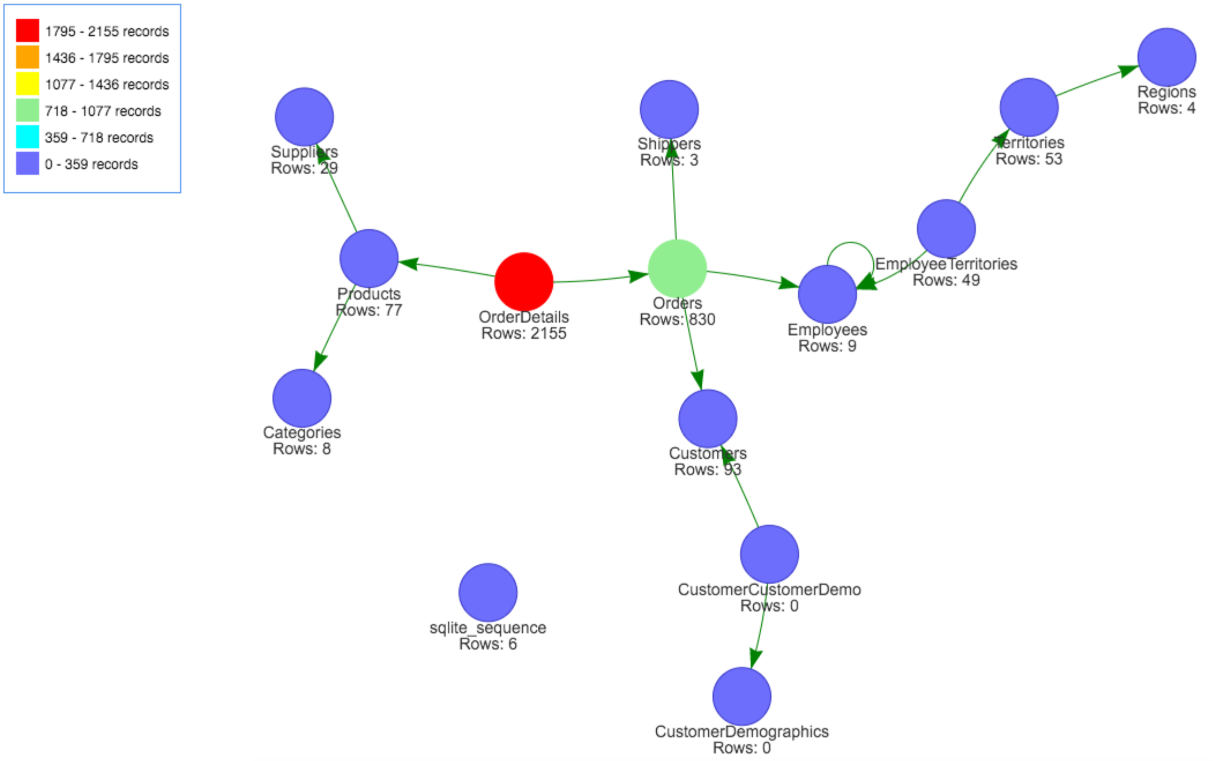
This graph also comes with a color scale on record count, being able to detect denser tables. To access it just right click the root node of the tree and then select the action *Render Graph* > *Detailed Graph*:



```
-- Querying Data
select t.*
from Categories t
```

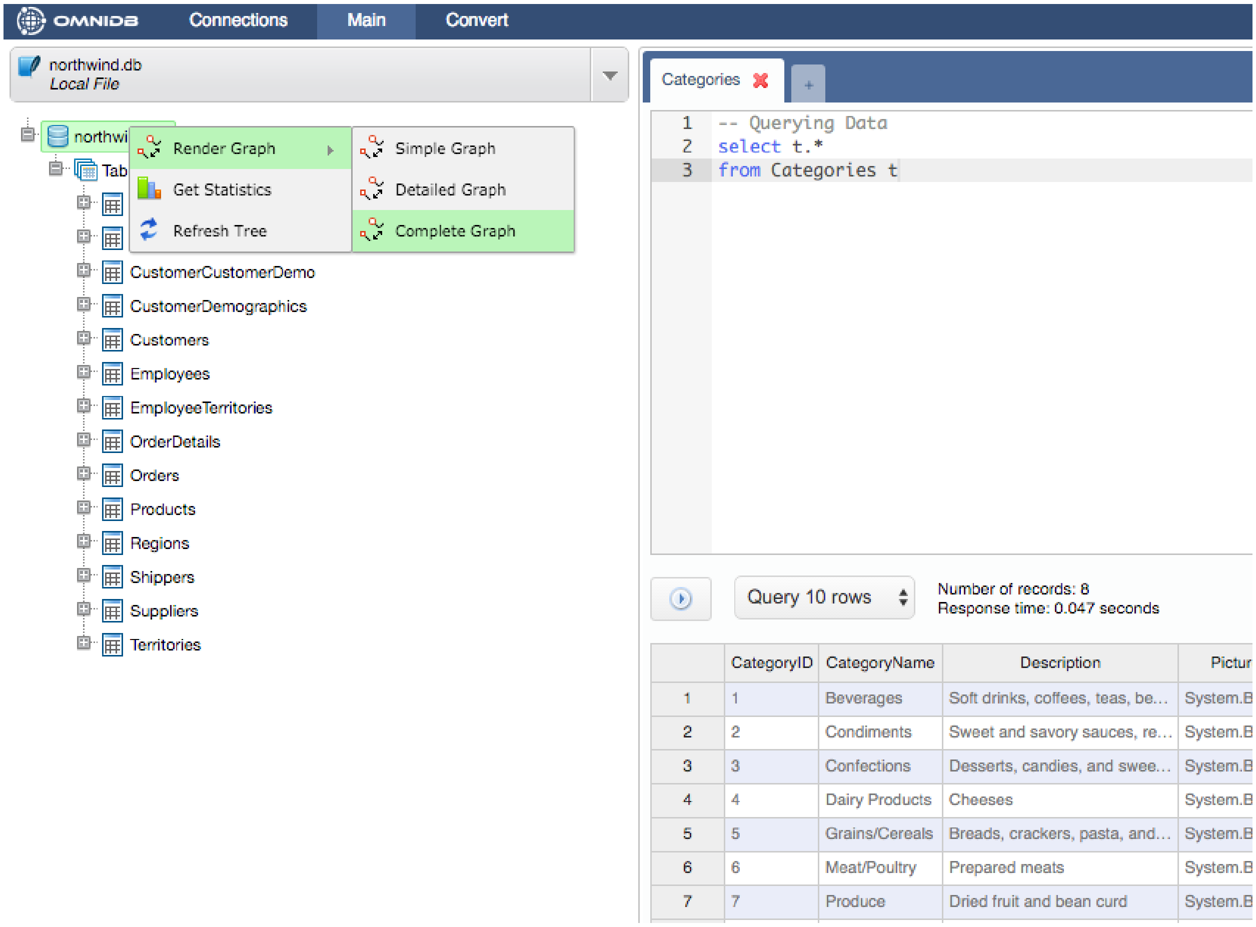
Number of records: 8
Response time: 0.047 seconds

	CategoryID	CategoryName	Description	Picture
1	1	Beverages	Soft drinks, coffees, teas, be…	System.By
2	2	Condiments	Sweet and savory sauces, re…	System.By
3	3	Confections	Desserts, candies, and swee…	System.By
4	4	Dairy Products	Cheeses	System.By
5	5	Grains/Cereals	Breads, crackers, pasta, and…	System.By
6	6	Meat/Poultry	Prepared meats	System.By



9.2.3. Complete Graph

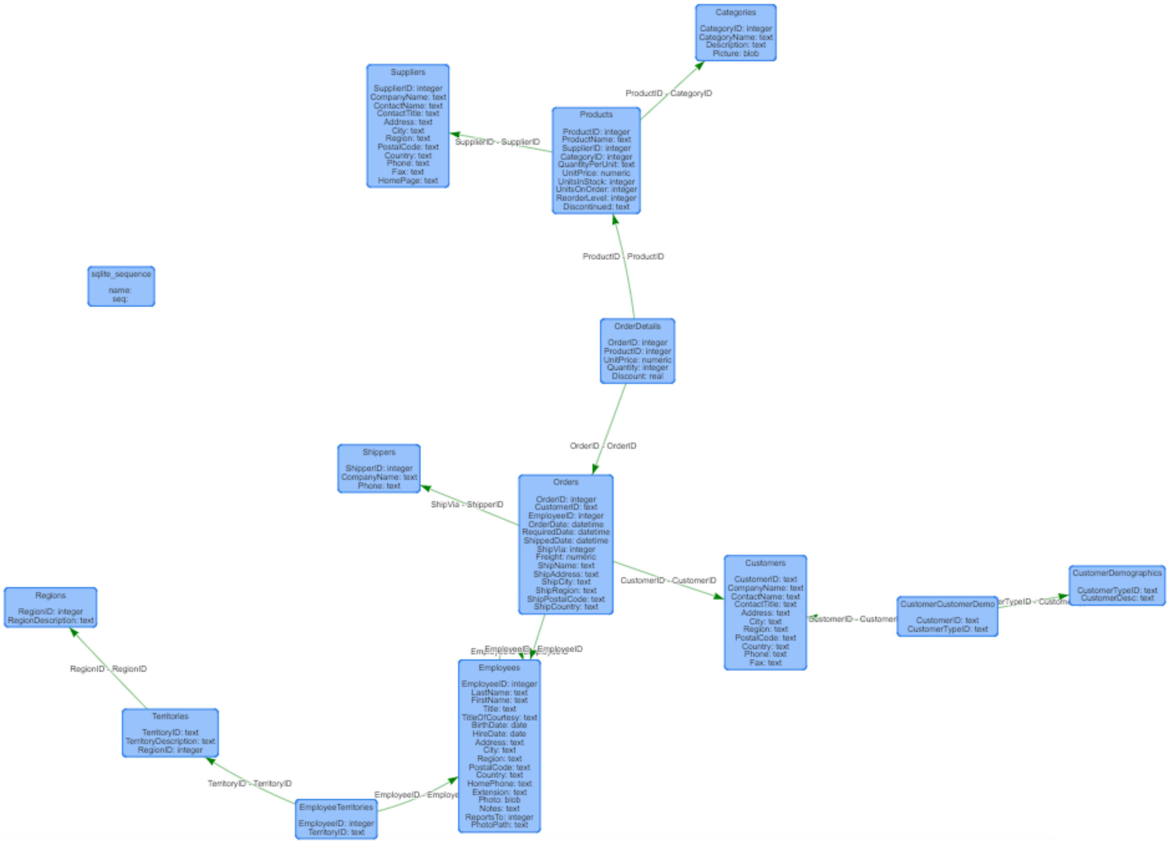
This graph displays tables with all its columns and respective data types. Additionally, edges now are labelled with information about the specific foreign key. To access it just right click the root node of the tree and then select the action *Render Graph > Complete Graph*:



```
-- Querying Data
select t.*
from Categories t
```

Number of records: 8
Response time: 0.047 seconds

	CategoryID	CategoryName	Description	Pictur
1	1	Beverages	Soft drinks, coffees, teas, be…	System.B
2	2	Condiments	Sweet and savory sauces, re…	System.B
3	3	Confections	Desserts, candies, and swee…	System.B
4	4	Dairy Products	Cheeses	System.B
5	5	Grains/Cereals	Breads, crackers, pasta, and…	System.B
6	6	Meat/Poultry	Prepared meats	System.B
7	7	Produce	Dried fruit and bean curd	System.B


Categories
CategoryID: integer
CategoryName: text
Description: text
Picture: blob
Suppliers
SupplierID: integer
CompanyName: text
ContactName: text
ContactTitle: text
Address: text
City: text
Region: text
PostalCode: text
Country: text
Phone: text
Fax: text
HomePage: text
ProductID - CategoryID
SupplierID - SupplierID
Products
ProductID: integer
ProductName: text
SupplierID: integer
CategoryID: integer
QuantityPerUnit: text
UnitPrice: numeric
UnitsInStock: integer
UnitsOnOrder: integer
ReorderLevel: integer
Discontinued: text
sqlite_sequence
name:
seq:
ProductID - ProductID
OrderDetails
OrderID: integer
ProductID: integer
UnitPrice: numeric
Quantity: integer
Discount: real
OrderID - OrderID
Shippers
ShipperID: integer
CompanyName: text
Phone: text
ShipVia - ShipperID
Orders
OrderID: integer
CustomerID: text
EmployeeID: integer
OrderDate: datetime
RequiredDate: datetime
ShippedDate: datetime
ShipVia: integer
Freight: numeric
ShipName: text
ShipAddress: text
ShipCity: text
ShipRegion: text
ShipPostalCode: text
ShipCountry: text
CustomerID - CustomerID
Customers
CustomerID: text
CompanyName: text
ContactName: text
ContactTitle: text
Address: text
City: text
Region: text
PostalCode: text
Country: text
Phone: text
Fax: text
CustomerDemographics
CustomerTypeID: text
CustomerDesc: text
CustomerCustomerDemo
CustomerID: text
CustomerTypeID: text
CustomerID - Customer
erTypeID - Custome
Regions
RegionID: integer
RegionDescription: text
EmployeeID - EmployeeID
RegionID - RegionID
Employees
EmployeeID: integer
LastName: text
FirstName: text
Title: text
TitleOfCourtesy: text
BirthDate: date
HireDate: date
Address: text
City: text
Region: text
PostalCode: text
Country: text
HomePhone: text
Extension: text
Photo: blob
Notes: text
ReportsTo: integer
PhotoPath: text
Territories
TerritoryID: text
TerritoryDescription: text
RegionID: integer
TerritoryID - TerritoryID
EmployeeID - Employe
EmployeeTerritories
EmployeeID: integer
TerritoryID: text

10. Schema Conversion

OmniDB allows 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 and the target connection. The current version allows the conversion of tables with all of their structures (columns, restrictions and indexes), along with their data.

10.1. Data Type Compatibility

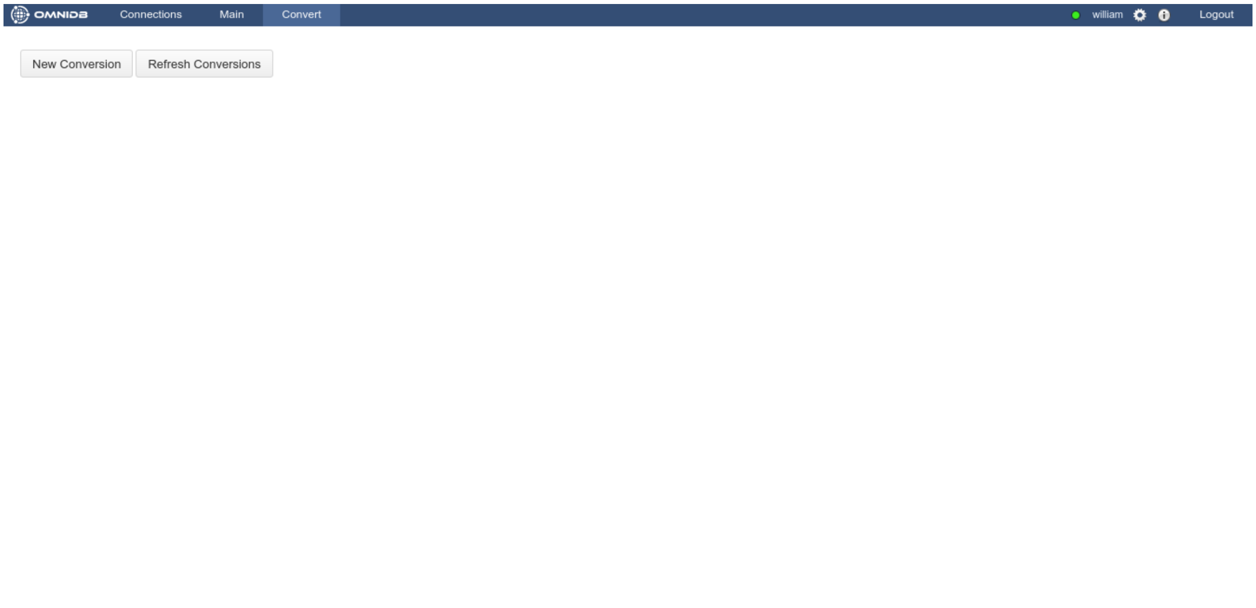
Since each DBMS possess specific data types, which often aren't shared among other technologies, or even when they are, they are found with different names, OmniDB needs to handle these inconsistencies during the conversion.

To achieve this, we utilize a simple data type mapping system. OmniDB has a list of data type categories and a list of every type of data from all technologies, each of them belonging to exactly one category. Besides, each DBMS possess a data type representing each category. With this system, it's possible to map any data type among any DBMS source-target pair, requiring only to to categorize an data type and using the correct representative.

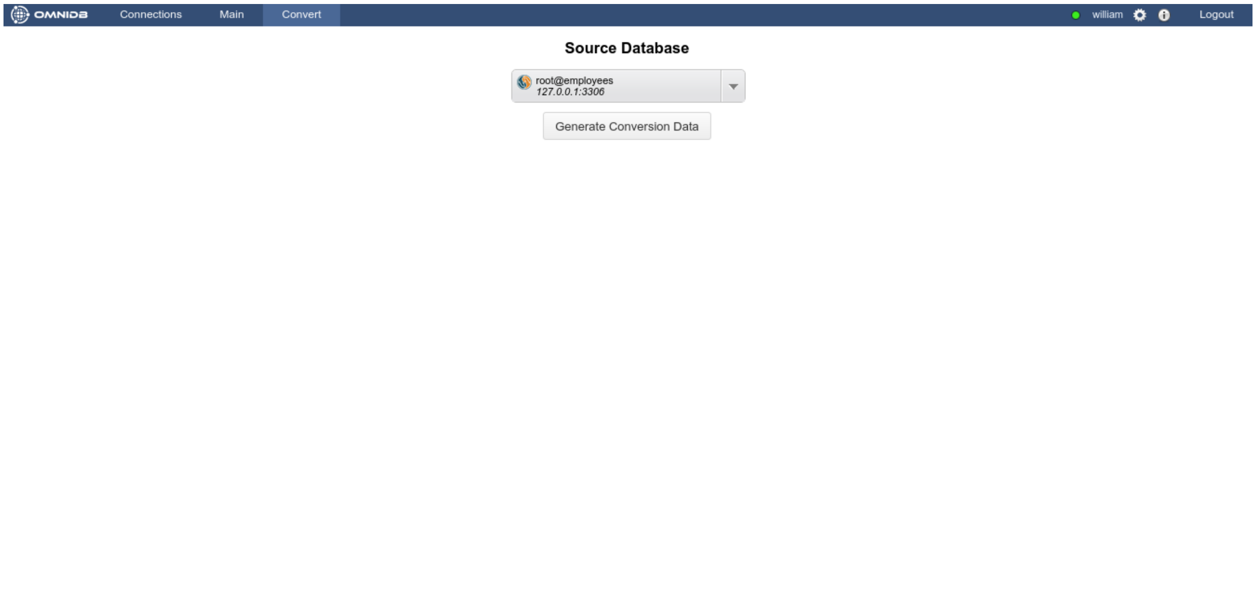
As example, conversion between Oracle and PostgreSQL: Oracle's data type *varchar2* belongs to OmniDB's category *varchar*, and it would use PostgreSQL's representative *character varying*. Thus, every *varchar2* column found on source connection would be created in the target connection with the *character varying* type.

10.2. Making Conversions

To convert schemas through OmniDB's interface, the user has to enter the *Conversions* window through the *Convert* link in the top screen's main menu.

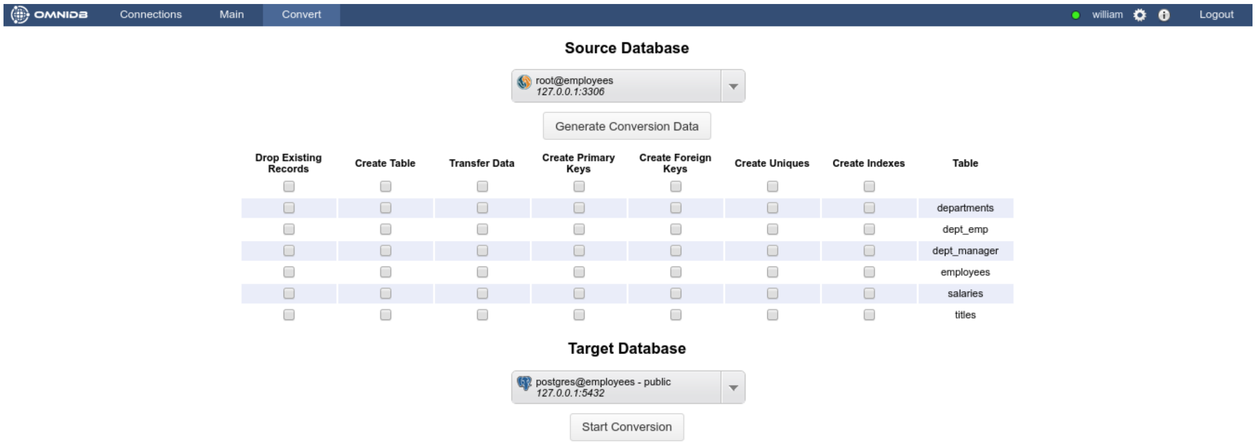


This interface displays a conversion list created in a grid shape, as well as actions to update the grid and to create a new conversion. Clicking on the *New Conversion* button leads the user to the following screen:



Source Database
root@employees
127.0.0.1:3306

In this interface, the user first chooses the connection of origin and then clicks on *Generate Conversion Data*, which will display a list with all the tables included in the connection and a number of checkboxes that allows him to configure what must be converted:



Source Database

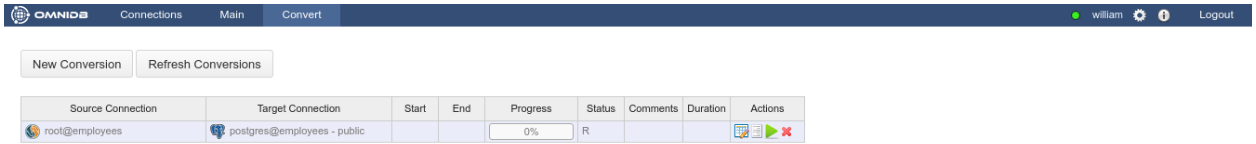
root@employees
127.0.0.1:3306

Drop Existing Records	Create Table	Transfer Data	Create Primary Keys	Create Foreign Keys	Create Uniques	Create Indexes	Table
□	□	□	□	□	□	□	
□	□	□	□	□	□	□	departments
□	□	□	□	□	□	□	dept_emp
□	□	□	□	□	□	□	dept_manager
□	□	□	□	□	□	□	employees
□	□	□	□	□	□	□	salaries
□	□	□	□	□	□	□	titles

Target Database

postgres@employees - public
127.0.0.1:5432

Once the user selects the structures chosen for conversion, they can click on *Create Conversion*, creating all the necessary configuration to realize a conversion (still, without actually initiating it).



Source Connection	Target Connection	Start	End	Progress	Status	Comments	Duration	Actions
root@employees	postgres@employees - public			0%	R			

The conversions grid now possess the record that we just created. This record has actions that allow us to manipulate and monitor the conversion. The first action leads the user to a screen that details the conversion of each table.

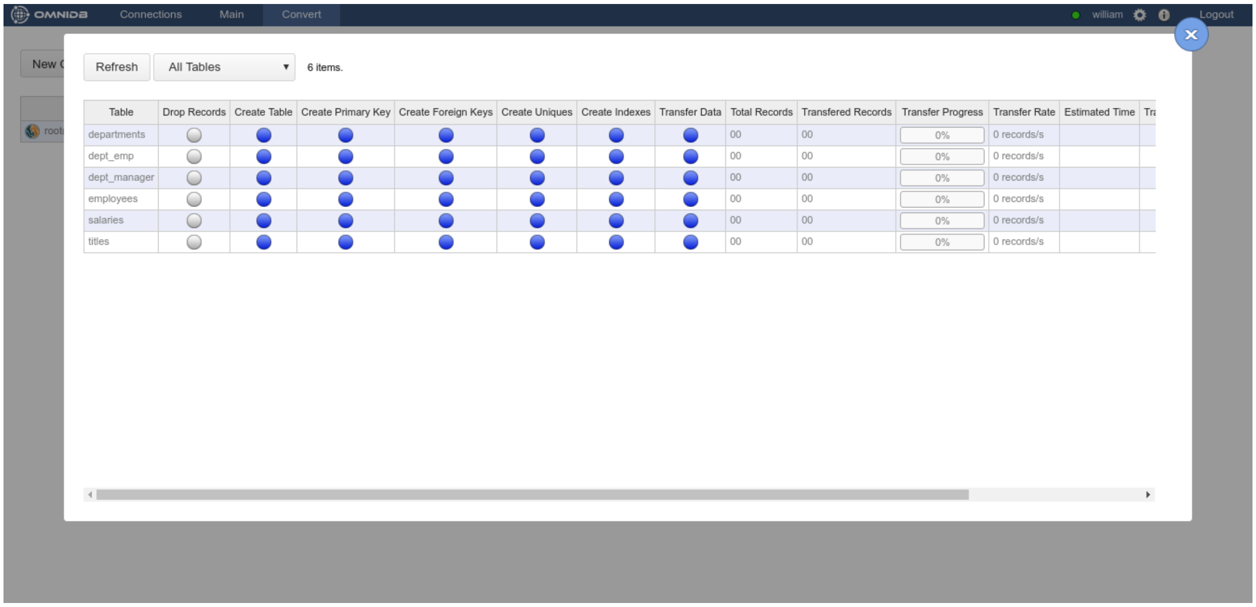
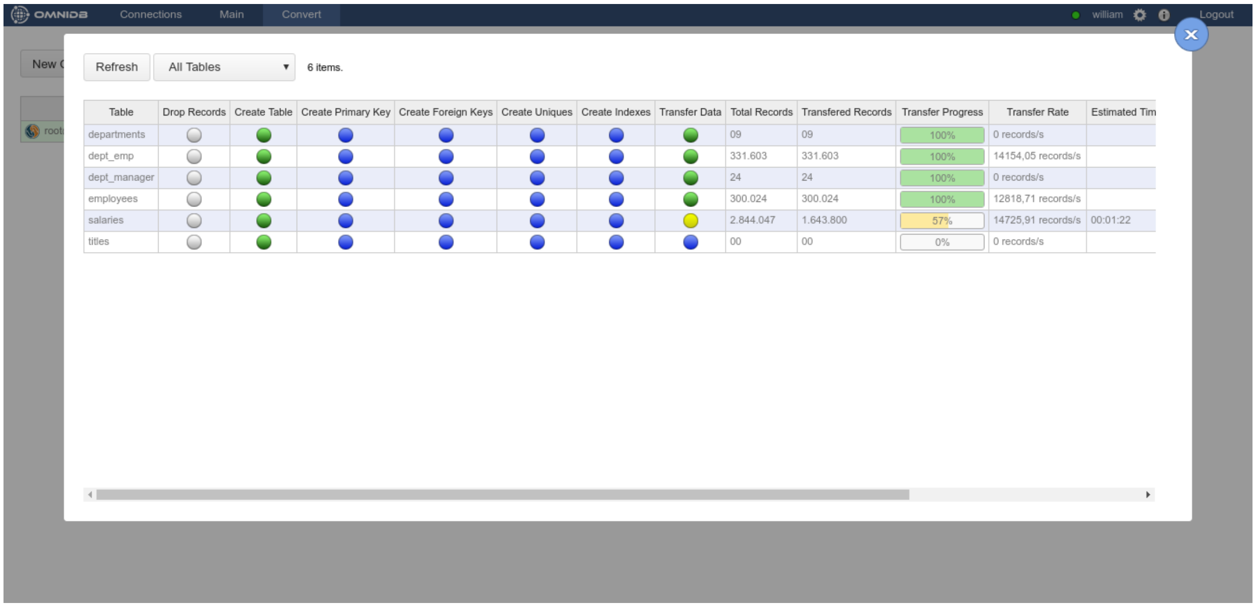


Table	Drop Records	Create Table	Create Primary Key	Create Foreign Keys	Create Uniques	Create Indexes	Transfer Data	Total Records	Transfered Records	Transfer Progress	Transfer Rate	Estimated Time	Tra
departments	○	●	●	●	●	●	●	00	00	0%	0 records/s		
dept_emp	○	●	●	●	●	●	●	00	00	0%	0 records/s		
dept_manager	○	●	●	●	●	●	●	00	00	0%	0 records/s		
employees	○	●	●	●	●	●	●	00	00	0%	0 records/s		
salaries	○	●	●	●	●	●	●	00	00	0%	0 records/s		
titles	○	●	●	●	●	●	●	00	00	0%	0 records/s		

The columns *Drop Records*, *Create Table*, *Create Primary Key*, *Create Foreign Keys*, *Create Uniques*, *Create Indexes* and *Transfer Data* have images representing the state of conversion for each structure. The states are:

- Grey sphere: structure not converted;
- Blue sphere: structure still not converted;
- Yellow sphere: structure under conversion;
- Green sphere: structure successfully converted;
- Red sphere: error during conversion.

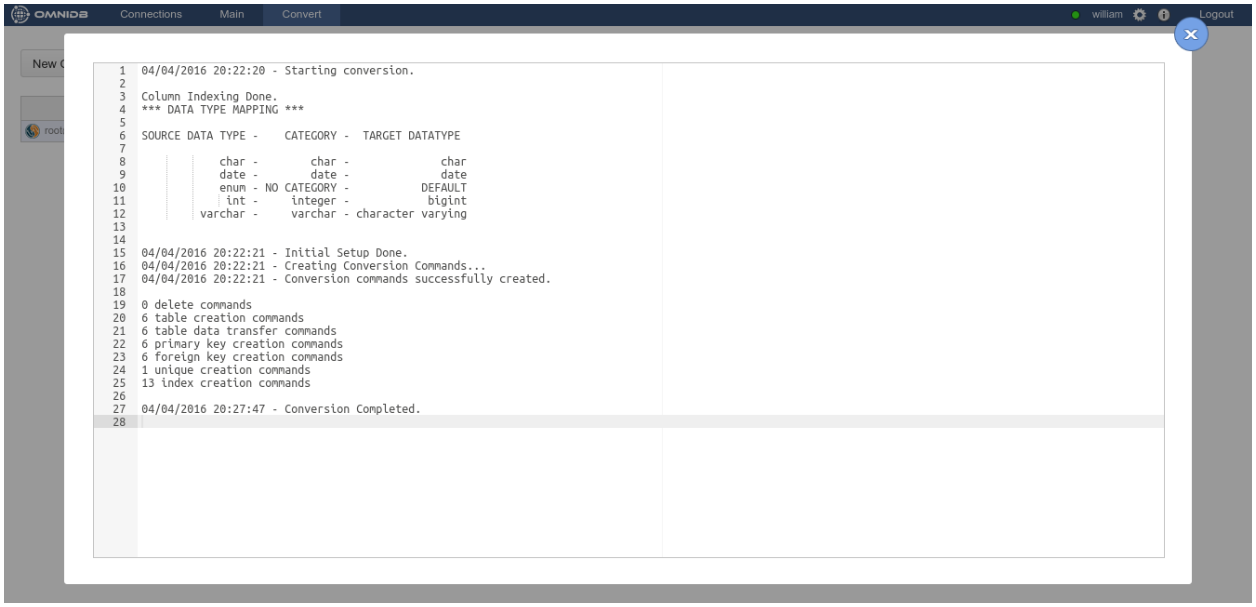
Besides, this interface allows the user to monitor the data transfer process for each table, displaying information such as: total record count to be converted, amount of successfully transfered records, transfer speed (in record/second), estimated transfer duration time, among others.



6 items.

Table	Drop Records	Create Table	Create Primary Key	Create Foreign Keys	Create Uniques	Create Indexes	Transfer Data	Total Records	Transfered Records	Transfer Progress	Transfer Rate	Estimated Tim
departments	○	🟢	🔵	🔵	🔵	🔵	🟢	09	09	100%	0 records/s	
dept_emp	○	🟢	🔵	🔵	🔵	🔵	🟢	331.603	331.603	100%	14154,05 records/s	
dept_manager	○	🟢	🔵	🔵	🔵	🔵	🟢	24	24	100%	0 records/s	
employees	○	🟢	🔵	🔵	🔵	🔵	🟢	300.024	300.024	100%	12818,71 records/s	
salaries	○	🟢	🔵	🔵	🔵	🔵	🟡	2.844.047	1.643.800	57%	14725,91 records/s	00:01:22
titles	○	🟢	🔵	🔵	🔵	🔵	🔵	00	00	0%	0 records/s	

The second action in the conversion grid displays a text log detailing the conversion. In this log, it's possible to see the time in which it started, how did OmniDB map the data types between the DBMSs, eventual errors that might have ocurred during the conversion (detailing the commands that were utilized and which errors ocurred) and the time that the conversion finished.



```
04/04/2016 20:22:20 - Starting conversion.

Column Indexing Done.
*** DATA TYPE MAPPING ***

SOURCE DATA TYPE -    CATEGORY -  TARGET DATATYPE

            char -        char -              char
            date -        date -              date
            enum - NO CATEGORY -           DEFAULT
             int -     integer -            bigint
         varchar -     varchar - character varying


04/04/2016 20:22:21 - Initial Setup Done.
04/04/2016 20:22:21 - Creating Conversion Commands...
04/04/2016 20:22:21 - Conversion commands successfully created.

0 delete commands
6 table creation commands
6 table data transfer commands
6 primary key creation commands
6 foreign key creation commands
1 unique creation commands
13 index creation commands

04/04/2016 20:27:47 - Conversion Completed.

```

The third action (green arrow) actually starts the conversion. It's important to notice that the conversion is an independent process, initiated when the user clicks this action. This process runs on background and doesn't prevent simultaneous use of OmniDB, which means that the user can navigate the tool and manipulate it without affecting the conversion.

Once initiated, the conversion will show the user a new action, allowing him to cancel the conversion.