PostGIS – Users and Permissions

In this blog we will explore creating **Users** and applying **Permissions** so that Users can access and edit data within your PostGIS database in a more controlled environment.

Having worked closely with a number of our clients who have made the move to storing their spatial data within a PostGIS Database, this blog aims to illustrate some of the best practice that you should implement to ensure the integrity of your datasets. Many people simply load their spatial data into the ***PUBLIC*** schema of the ***Postgres*** database, and allow access to all tables via the default ***Postgres*** User which is created on first install. However, if you want better governance over these sometimes-sensitive datasets, then you will need to implement a more controlled policy to accessing and, in particular, editing those assets. This blog aims to identify some of the key steps when starting to look at PostGIS database security, with an emphasis on how this allows your users to access data via client applications, such as **QGIS.**

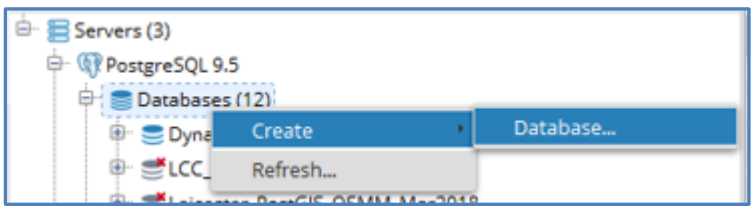
1 - Create Database and Schemas:

Before we look at creating Users and applying permissions for those Users, we should start by ensuring that our spatial datasets are stored within **separate Schemas** in our Database. This will then allow us to define access and edit rights to tables in our database via those Schemas.

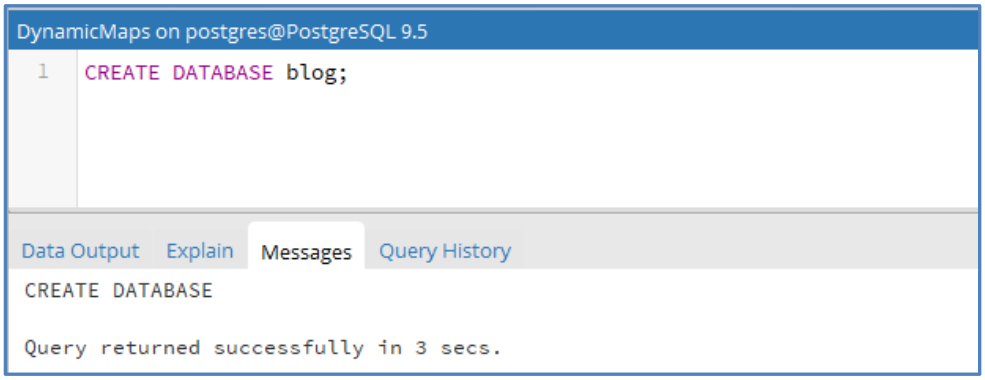
As the **Super User** for your PostGIS database, you have the rights to create Databases, create Roles, Create Users and update the permissions for other Users. Here, as the Super User we will create a new Database called ***'blog'*** which will then have separate Schemas to store our different spatial datasets.

We will be using the **PGAdmin interface** to make changes to our PostGIS database, although the more adventurous can utilise the Psql command line.

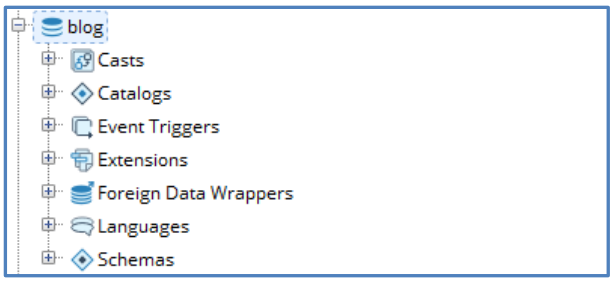
Having logged into the PostGIS Instance as a Super User we can create a new Database by **right clicking** on the PostGIS Instance and choosing **Create > Database.**



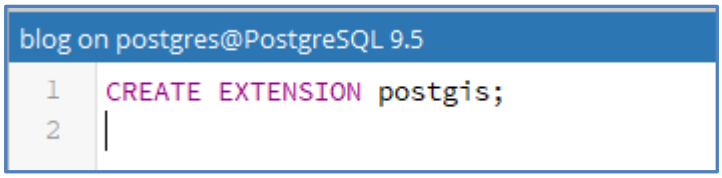
Alternately, you can also use the following command within a query window.



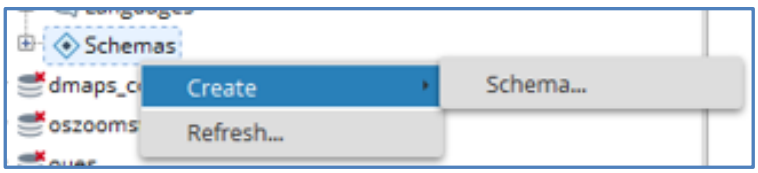
There will now be a new Database within your PostGIS Instance called ***blog.***



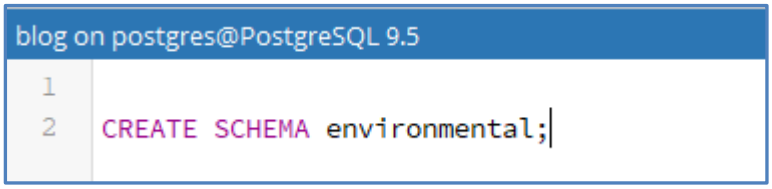
Before we go any further we need to ensure that the new blog database is a spatial database, by running the command to create the **PostGIS extension**.



Within this new database we will now create 2 new SCHEMAS to store our different spatial datasets, one will store **Environmental** data and the other **Points of Interest**. Again, we could use the menus within PGAdmin to create a new Schema.

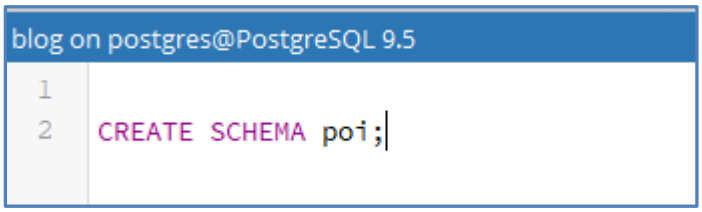


Or within the Query window simply type:

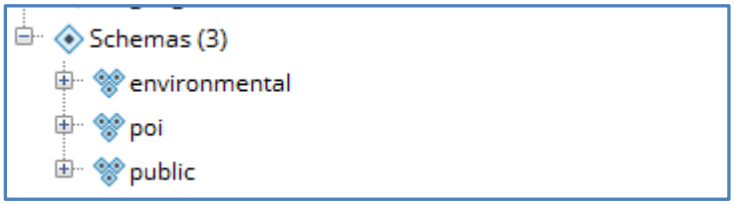


Note – *when using PGAdmin query window the queries will be performed on the Database Instance that is currently connected. So, ensure that you have connected to the new* ***blog*** *database before you run the query, or this new Schema will be created in another database.*

Let's now create the second Schema.

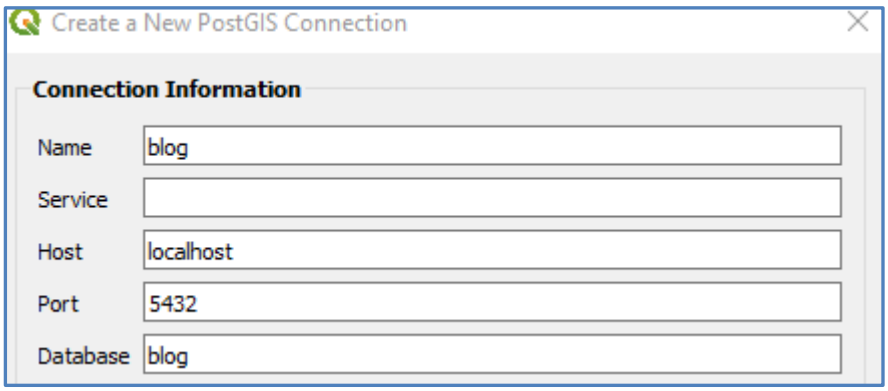


We now have **2 new Schemas** within our **blog** database.

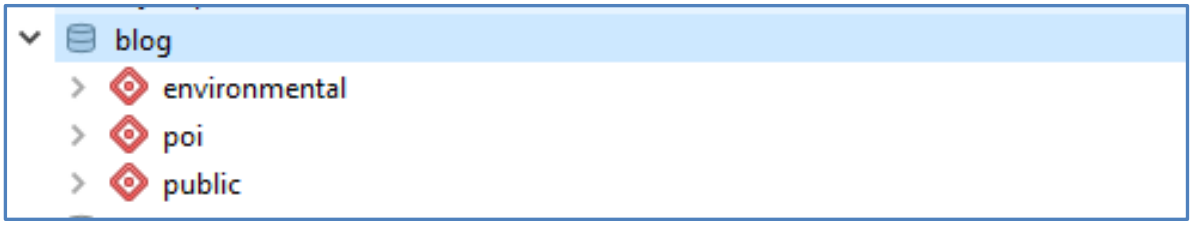


2 - Upload Data into Database:

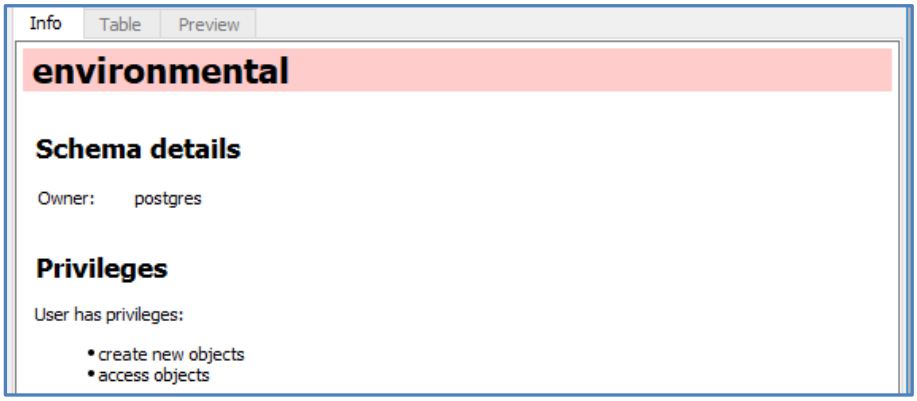
Next, we will use QGIS to connect to our new **blog** database and upload a number of spatial tables, so that we can then explore how to grant access to those tables for different Users. We will undertake all data uploads as the **Super User**.



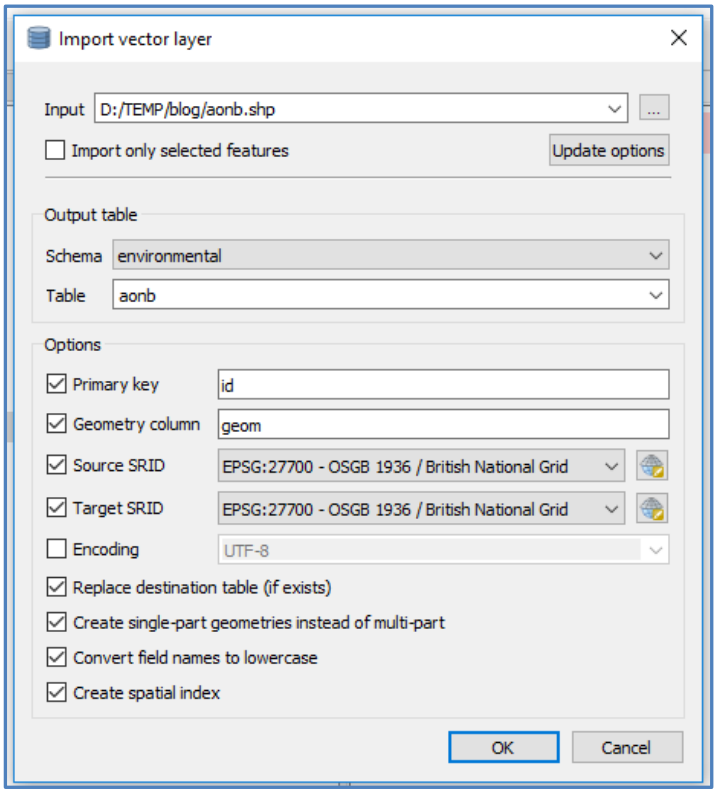
Having created a new **PostGIS Database** connection in QGIS, we can then use the **Database Manager** tool to upload a number of tables into the new **blog** database.



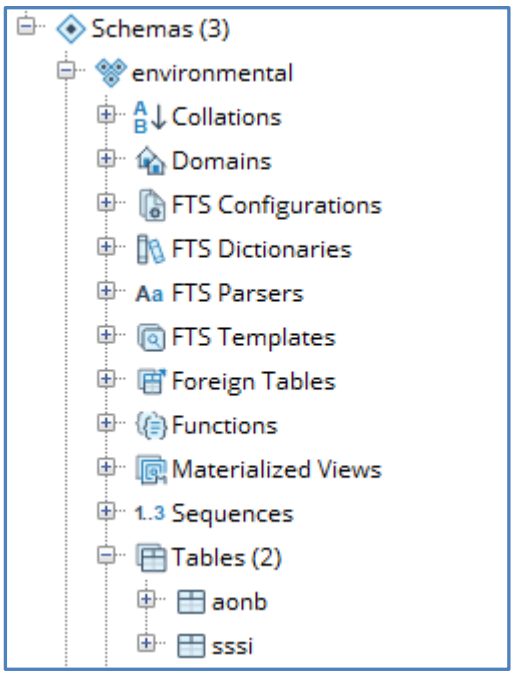
Logging in with the **Super User** postgres account automatically enables me to create new tables and also access/edit data from those tables for both of the 2 new Schemas.

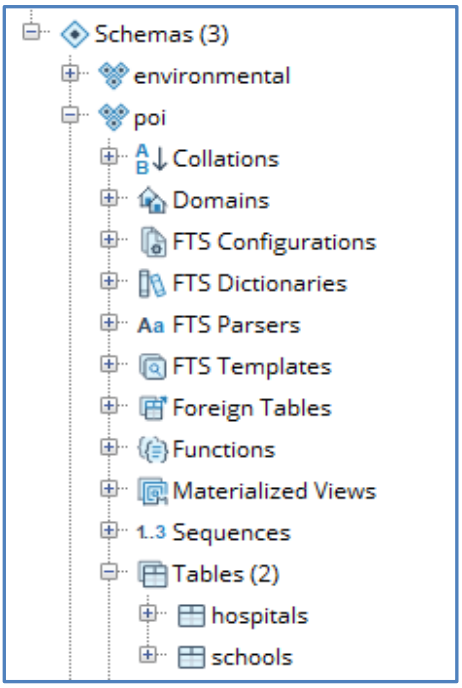


I can also use the **Import Layers** tool to import a number of spatial tables e.g. sssi, aonb, schools and hospitals, choosing to import into the required Schema as appropriate.



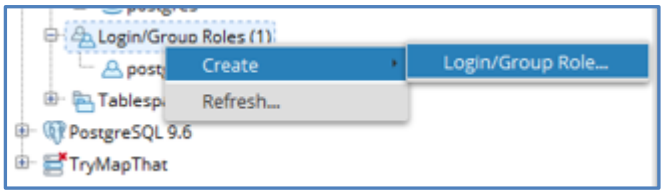
Once the tables have been imported, check within your **blog** database and the tables will have been added into the relevant Schema – **poi** or **environmental**.



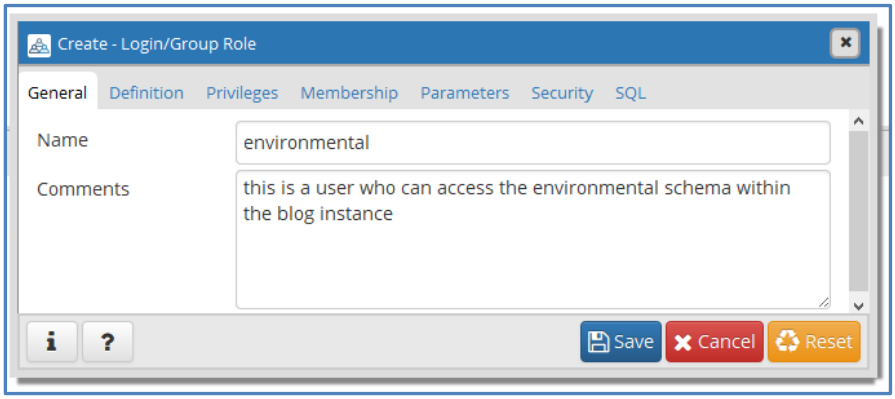


3 - Create Users/Roles:

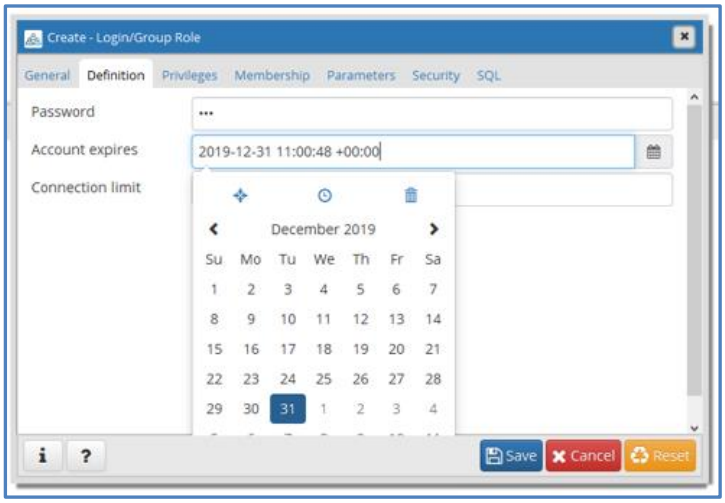
To start testing how our Users will connect to, and access these new spatial datasets, we will first create two new **Users**, one for the **environmenta**l Schema and one for the **poi** Schema. A new User can be created by a Super User from the Login/Group Roles Level, choosing **Create > Login/Group Role.**



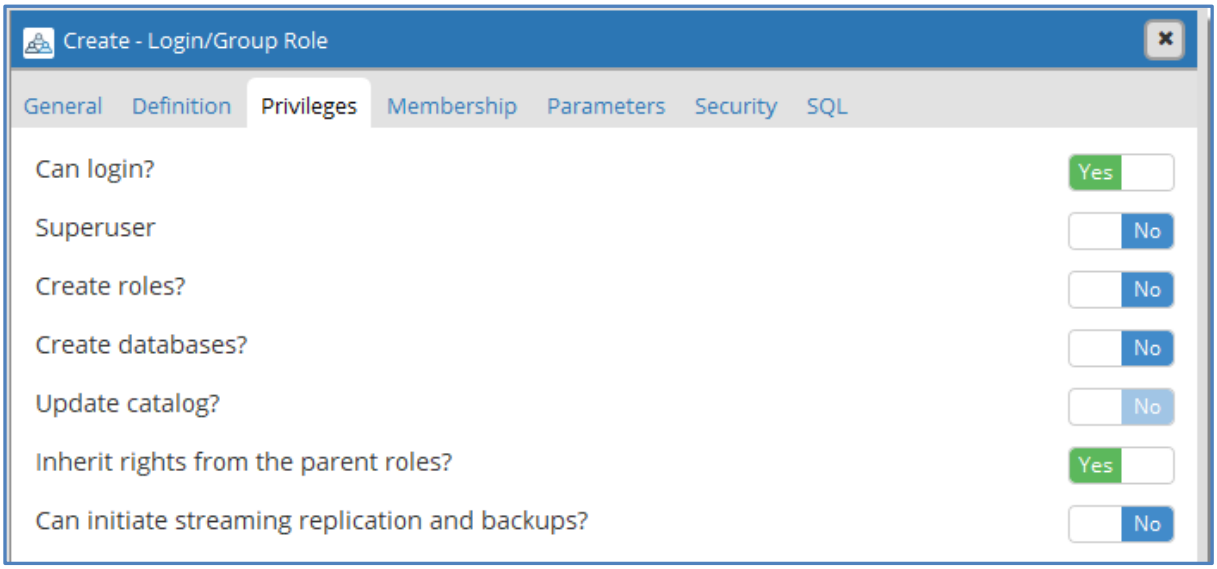
Within the Create – Login/Group Role, specify a **Name** for the new Login and provide a description within the **Comments**.



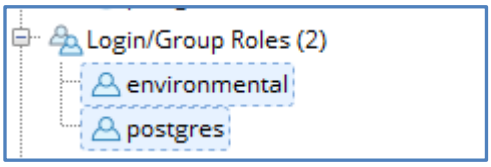
In the **Definition** tab you can specify the **Password** and an **Account expiration** date.



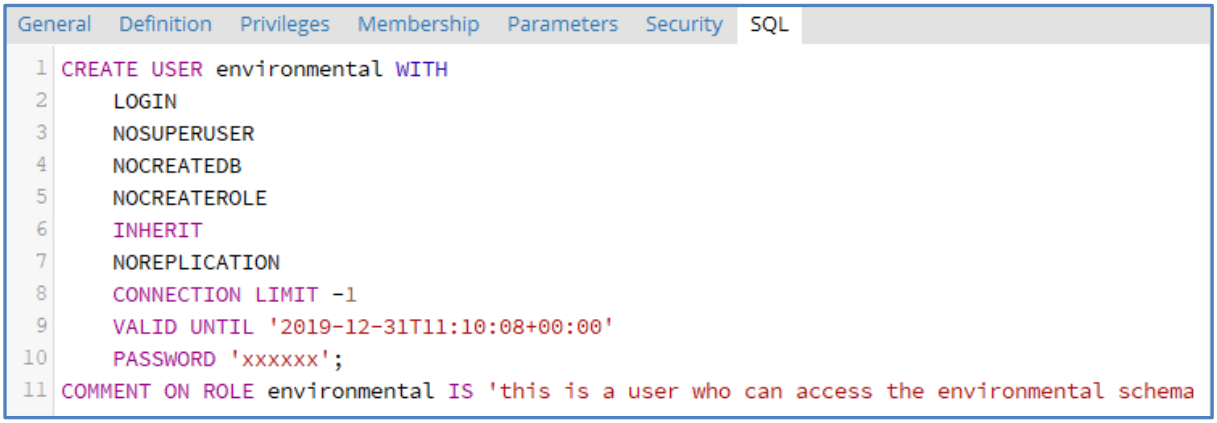
Within the **Privileges** tab you can specify whether the new User can login, create Roles and Databases and whether they have the same privileges as a Super User. We will allow the User to **login only.**



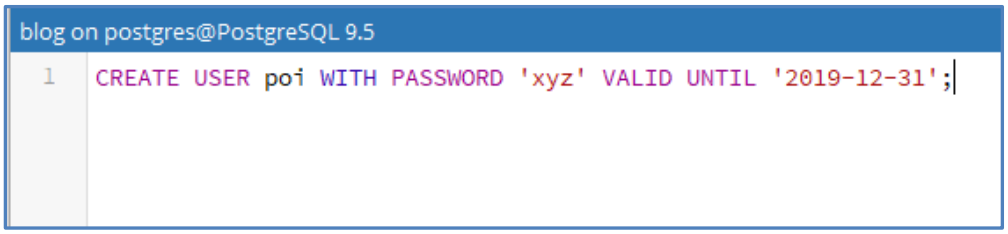
A new **User/Login** has now been created within the PostGIS Instance.



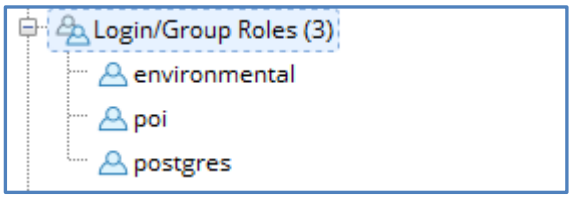
New Users/Logins can also be created using the Query Window which can often be quicker. The SQL used to create the above environmental user was shown in the **SQL** window.



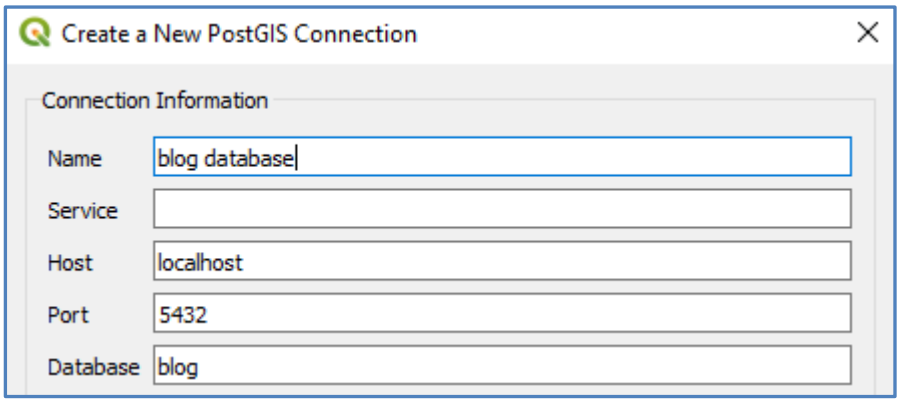
To simplify this SQL, we can simply create the User with a **Name**, specify a **Password** and the **Valid Until** date for the new **poi** User.



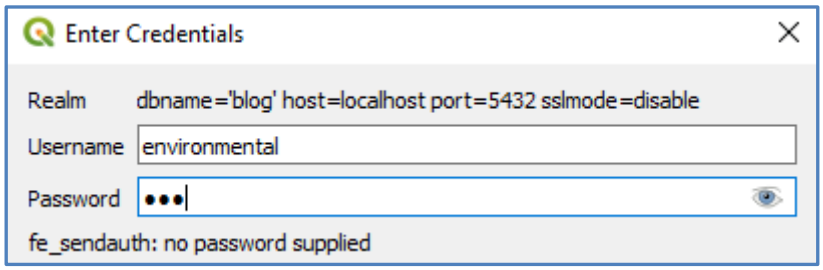
There will now be **3 Users** within the PostGIS Instance.



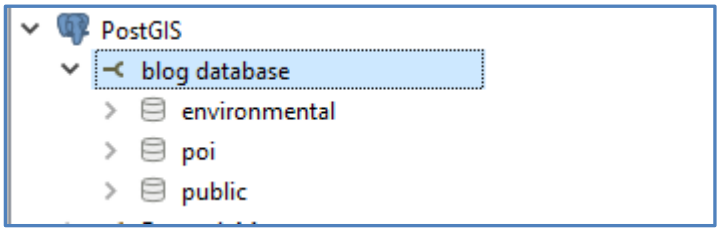
To test these new Users, we can use a new **Database Connection** via **QGIS** to the **blog** database.



Once the Connection is made, choose to login as the new **environmental** User.



The connection will open and currently the environmental user will see that there are 3 Schemas within the blog database, however when they try to click on a Schema to view the Tables the list will be **empty.**

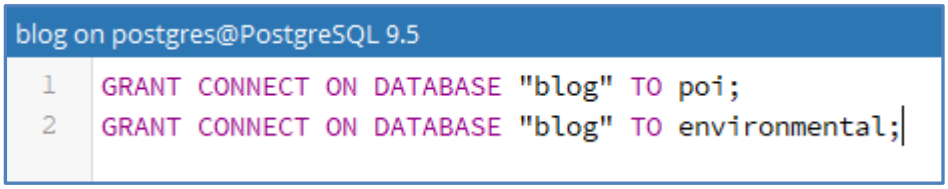


4 - Update Users/Roles to Grant Access and Edit Rights:

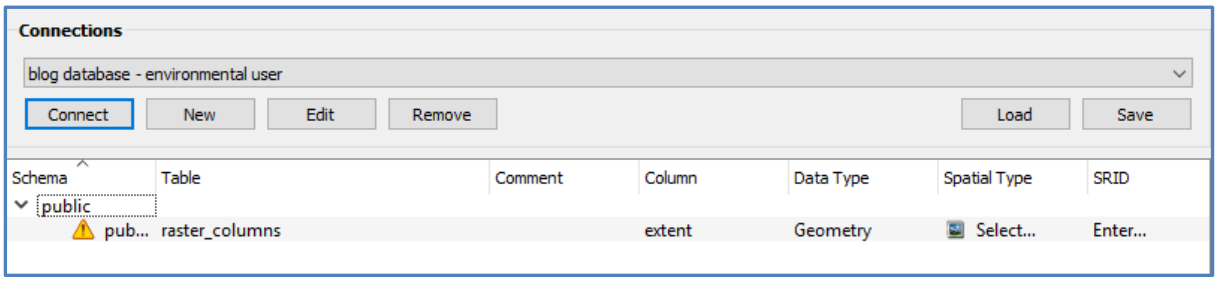
4.1. GRANT CONNECT on DATABASE

Now that we have the new **poi** and **environmental** users we will need to Grant them access to the **blog** database and then to relevant SCHEMA and Tables within those Schemas.

Firstly, we will **Grant** both Users access to the **blog** database.



Within QGIS if you try and log into the blog database as these Users, you will find that they can still only see the default **Public Schema** and no other Schemas.

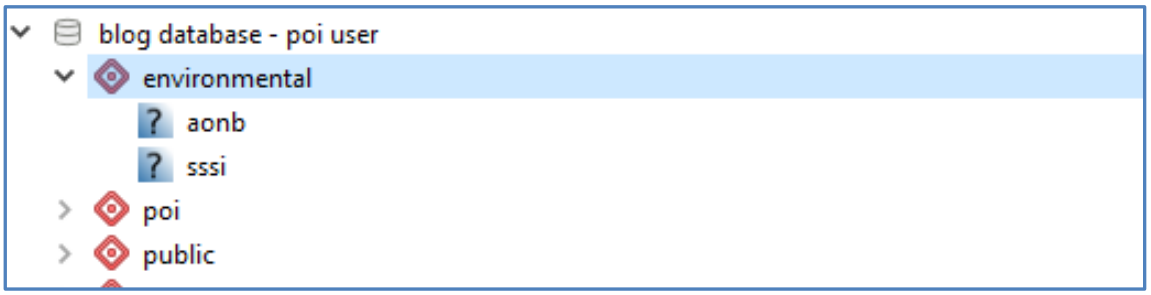


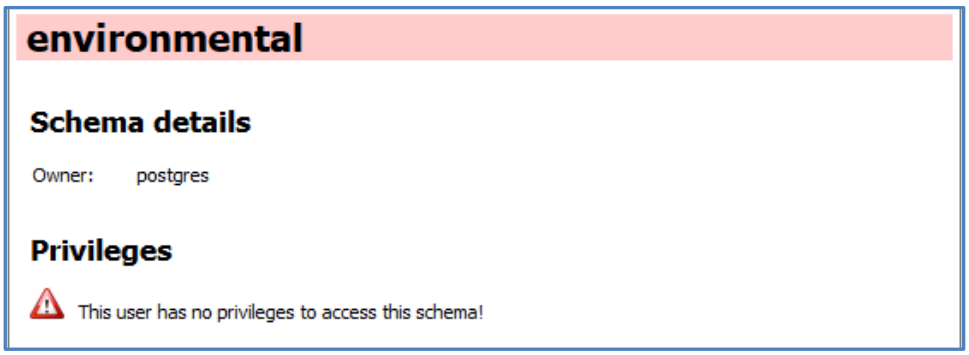
4.2. GRANT USAGE ON SCHEMA

So, next you will need to Grant each User with Access to the relevant **SCHEMAS** – either poi or environmental.

```
GRANT USAGE ON SCHEMA poi TO poi;
GRANT USAGE ON SCHEMA environmental TO environmental;
```

Once granted if you check in QGIS via the **Database Manager** tool you will see that the poi and environmental users still don't have the correct privileges to view Tables within the relevant Schema.





4.3. GRANT SELECT ON TABLE

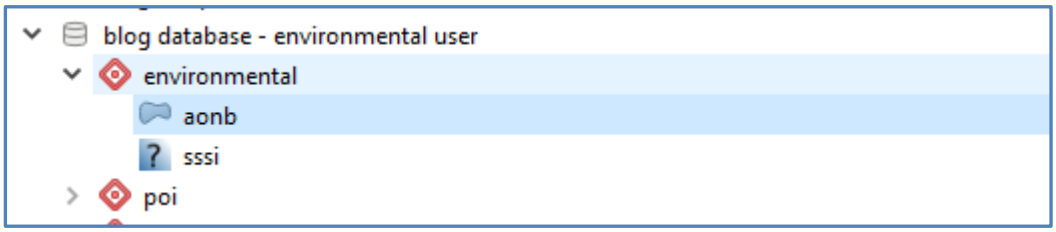
In order to complete these privileges, you will need to Grant the Users with **SELECT** privileges to at least one **Table** in their appropriate Schemas.

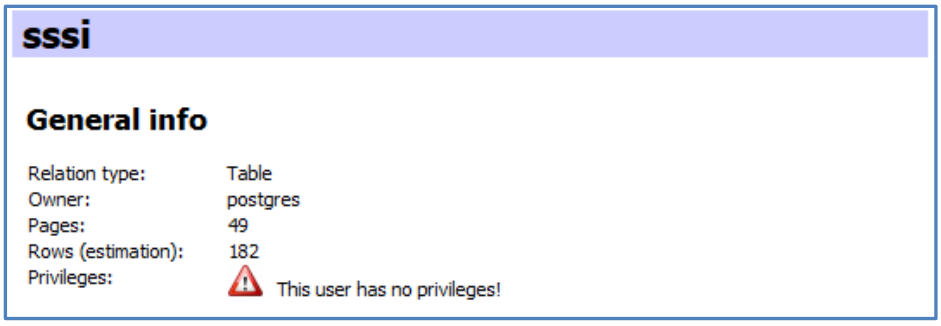
```
14  GRANT SELECT ON environmental.aonb TO environmental;
15  GRANT SELECT ON poi.schools TO poi;
16
17
```

Data Output | Explain | Messages | Query History

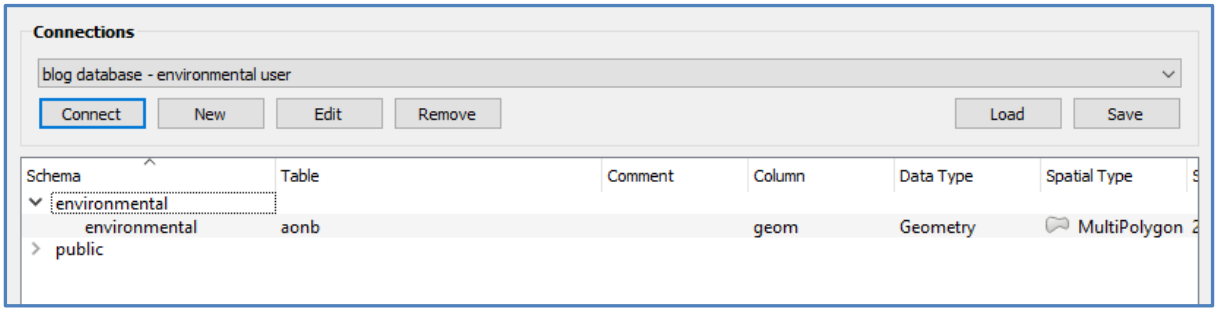
```
GRANT

Query returned successfully in 127 msec.
```

Now within QGIS the environmental User can see the **aonb** Table and has the privileges to SELECT against that Table, but they have no privileges yet to the **sssi** table.





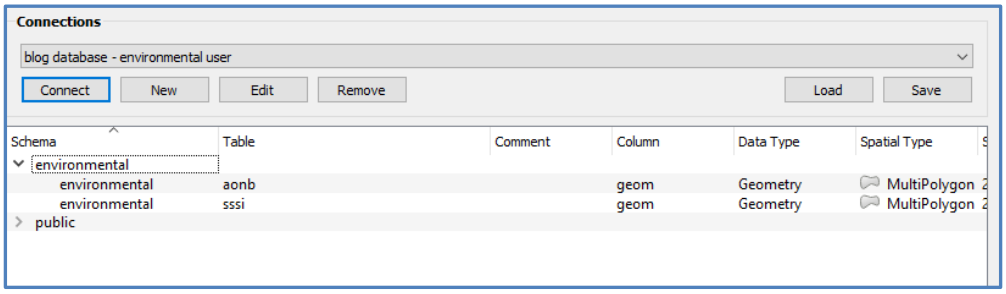
This means when trying to **Add a Layer** in QGIS from PostGIS, the environmental User can now add the **aonb** table but cannot add any other tables.



Complete the **Grant on SELECT** to each User to all the tables in their SCHEMA.

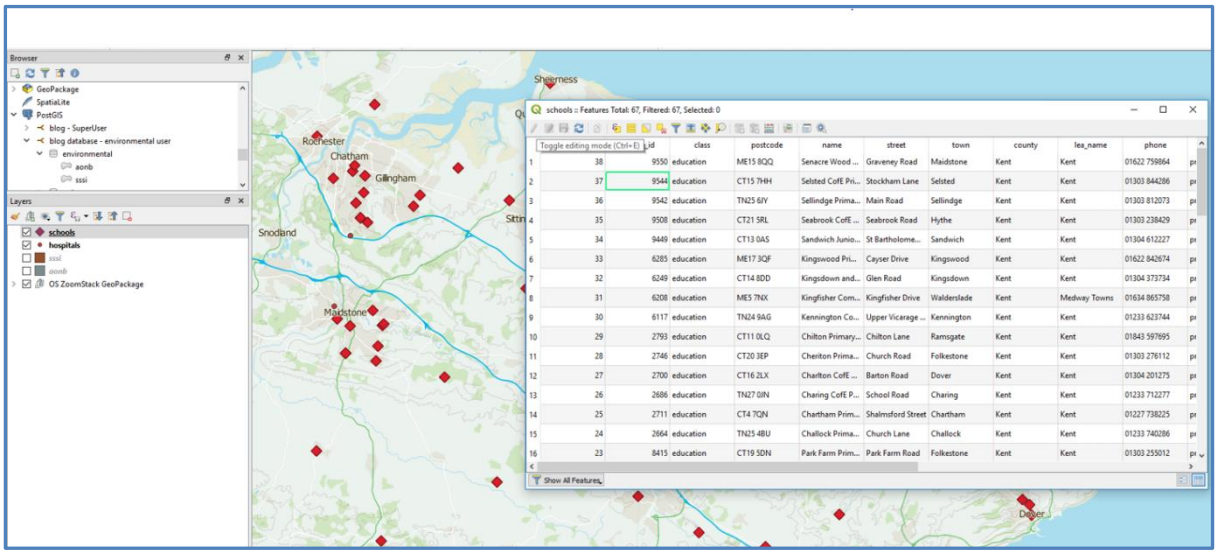
```
GRANT SELECT ON environmental.sssi TO environmental;
GRANT SELECT ON poi.hospitals TO poi;
```

Users will now be able to **ADD** (Select from) all the tables in their relevant Schema and load those into QGIS.



4.4. GRANT UPDATE ON TABLES

While the 2 new Users can now load and select from Tables in the relevant Schema, they don't currently have **Edit, Delete** or **Insert** Privileges. For example, if we logged into PostGIS as the POI user and viewed the attributes for the Schools Table, we can't edit the attributes or toggle the Layer into Edit mode. The Users Role is currently **Read Only** for these Tables.



To allow our POI User to edit the Schools Table we will run the **Grant on UPDATE** command for the Schools Table.

```
GRANT UPDATE ON poi.schools TO poi;
```

Now within QGIS if we **re-connect** to the Schools Table, we can toggle the **Edit Mode** to on and edit the **Attributes** of a record e.g. change a **School Name**.

	id	feature_id	class	postcode	name
1	1	1100	education	TN23 4QR	Updated Ashford Oaks Community Primary School
2	2	10176	education	CT21 5QE	St Augustine's Catholic Primary School
3	3	10602	education	CT9 4BU	St Gregory's Catholic Primary School
4	4	10246	education	ME5 8PU	St Benedict's Roman Catholic Primary School

Pressing **Save**, will then commit those edits back to the PostGIS database and update the Schools Table in the POI Schema.

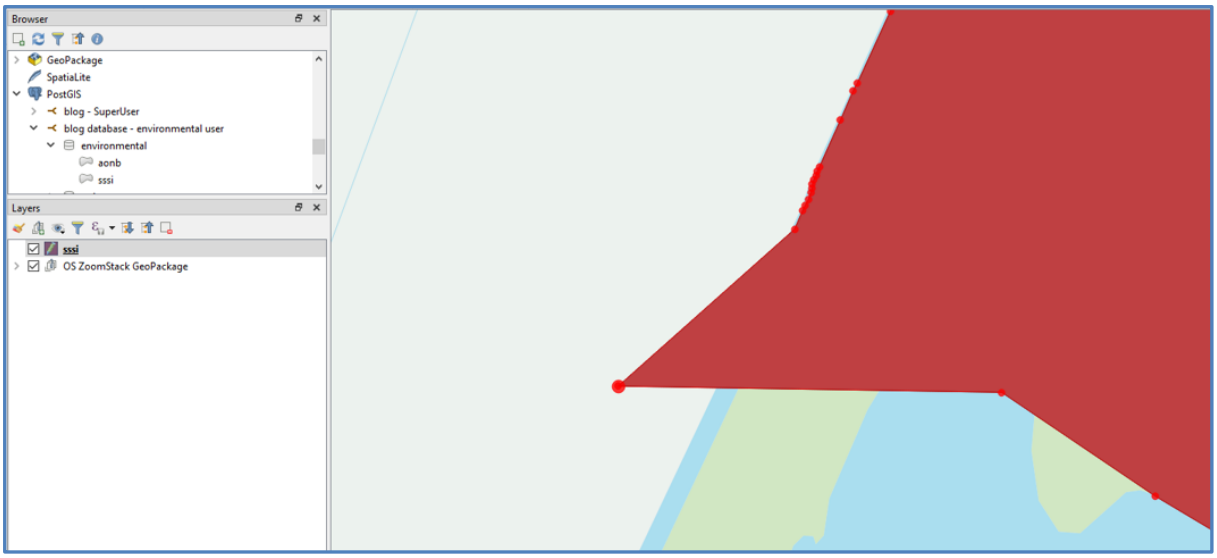
Data Output | Explain | Messages | Query History

	id [PK] integer	geom geometry	feature_id bigint	class character varying (254)	postcode character varying (254)	name character varying (254)
1	1	010100002...	1100	education	TN23 4QR	Updated Ashford Oaks Com...
2	2	010100002...	10176	education	CT21 5QE	St Augustine's Catholic Prim...
3	3	010100002...	10602	education	CT9 4BU	St Gregory's Catholic Prima...
4	4	010100002...	10246	education	ME5 8PU	St Benedict's Roman Catholi...
5	5	010100002...	10396	education	CT8 8EB	St Crispin's Community Pri...

Running the **GRANT UPDATE On** to both Users to each of their Tables will allow them to make **attribute Edits** as required.

```
GRANT UPDATE ON poi.hospitals TO poi;
GRANT UPDATE ON environmental.aonb TO environmental;
GRANT UPDATE ON environmental.sssi TO environmental;
```

Using the **GRANT UPDATE** privilege, a User can also **edit** the **existing geometry** of their spatial datasets e.g. **move** points and **reshape** polylines and polygons.

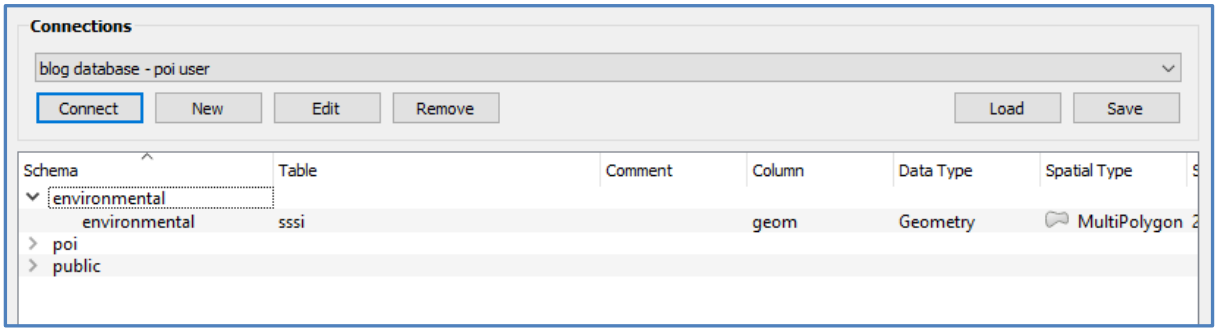


4.5. GRANT ACCESS to Multiple SCHEMAS

You may wish the POI User to now Access Tables from the Environmental Schema but not edit them. To do this, simply **GRANT USAGE** and then **GRANT SELECT** on the Tables that you wish to Share across Users.

```
GRANT USAGE ON SCHEMA environmental TO poi;
GRANT SELECT ON environmental.sssi TO poi;
```

If we now re-connect via QGIS as the **POI** User, we can now access and open the **sssi** Table.



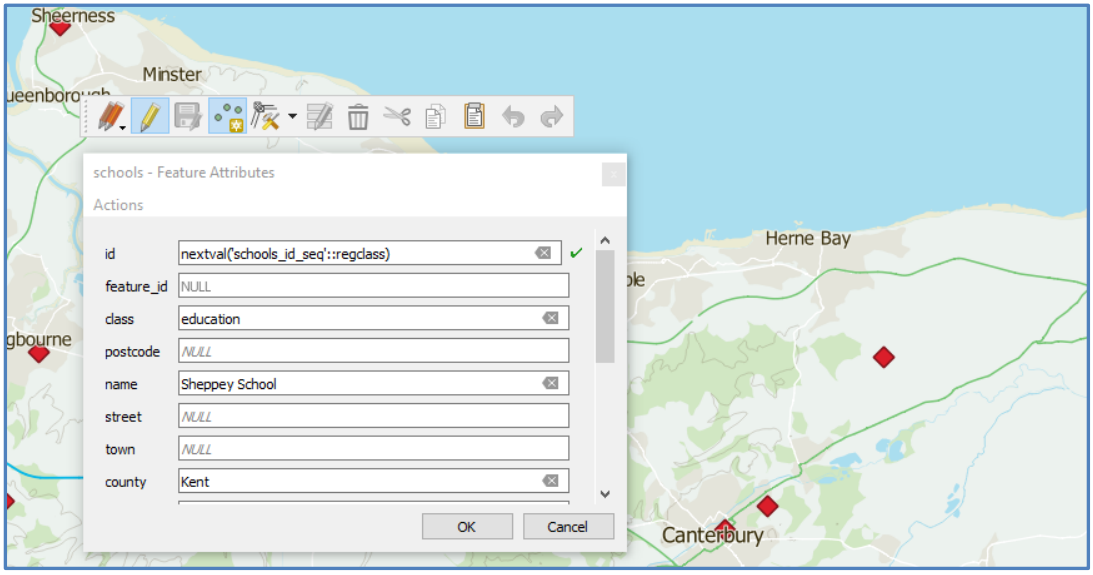
The POI User will only have **Read Only** Privileges though:

4.6. GRANT INSERT Privileges

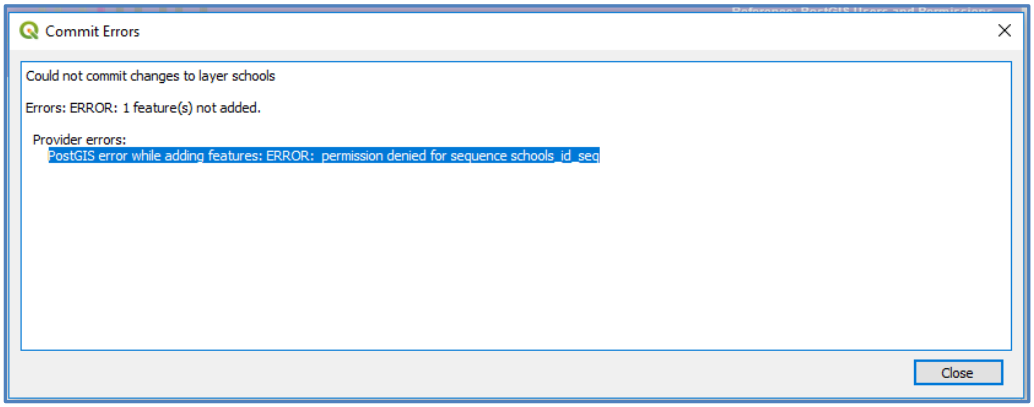
Now that we can edit attributes you may wish a User to be able to **Insert records** into an existing Table. To do this we can run the **GRANT INSERT On** command.

```
GRANT INSERT ON poi.schools TO poi;
```

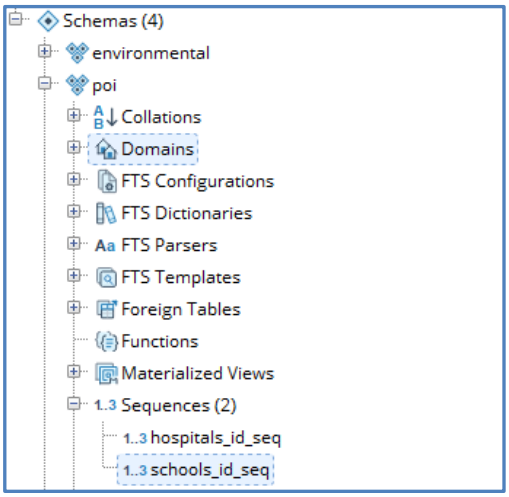
If we now re-connect via QGIS as the POI User, we can now **Insert (Add)** a new record into the Schools Table and save that insertion into the PostGIS database.



However, the INSERT may fail, because the new record is using a **SEQUENCE** on the ID field (Primary Key) to insert the next unique ID.



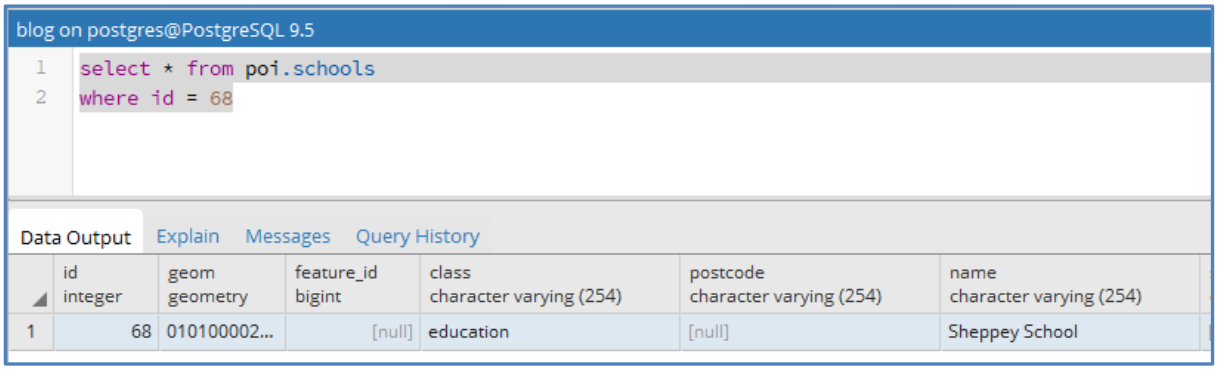
We haven't yet Granted any Privileges to the POI User to be able to Access any **SEQUENCES.**



To resolve this, we can **GRANT Usage** and **Select** on all the **Sequences** in the POI Schema to the POI User.

```
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA poi TO poi;
```

Now if we **Insert (Add)** the new School again via QGIS, the record is Saved, and the new record inserted into the Schools Table within your PostGIS database.



Complete the steps by allowing the Environmental User to Access all **Sequences** in their Schema.

```
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA environmental TO environmental;
```

And then Grant the **INSERT ON** to all the other spatial tables as required.

```
GRANT INSERT ON poi.hospitals TO poi;
GRANT INSERT ON environmental.sssi TO environmental;
GRANT INSERT ON environmental.aonb TO environmental;
```

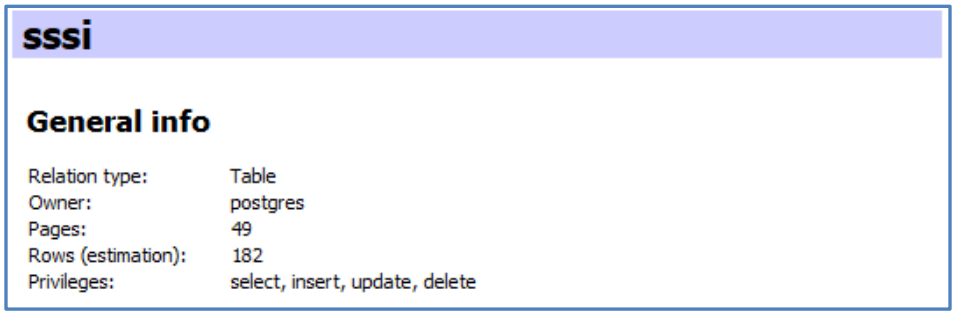
4.7. GRANT DELETE Privileges

The final privilege that we will activate is the option for a User to **Delete records** from their spatial tables. Currently because we have only granted – SELECT, UPDATE and INSERT, the Users cannot **Delete features** – and in QGIS the option will be greyed out.

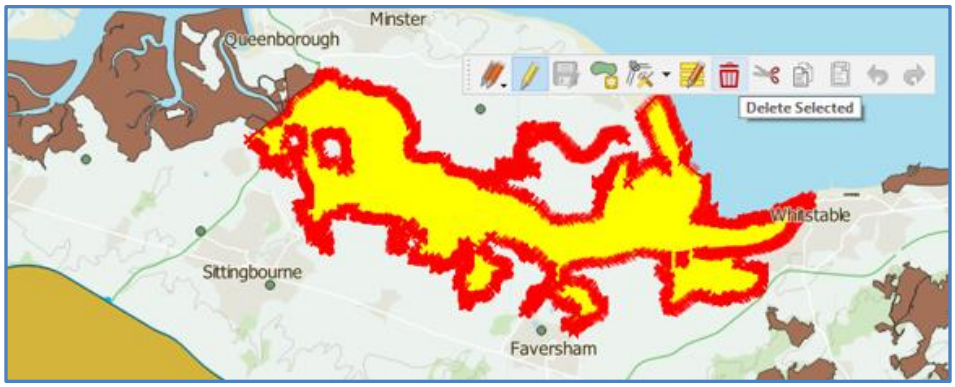
To enable this option, we will simply run the **GRANT DELETE On** statement:

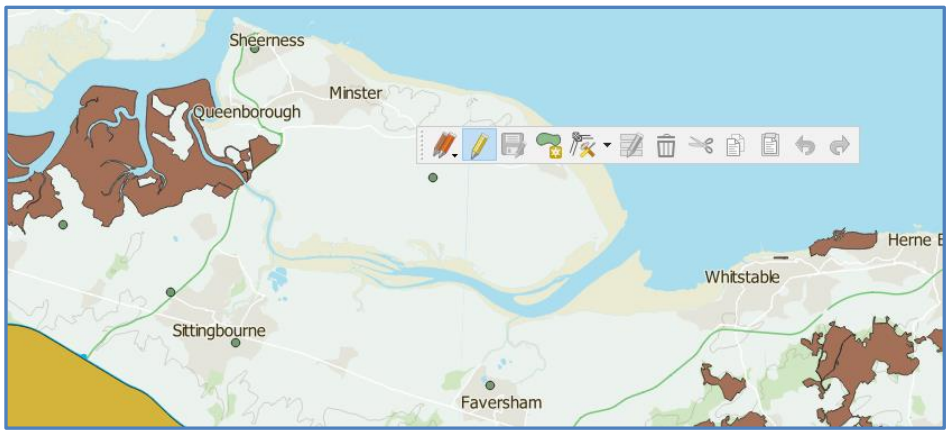
```
GRANT DELETE ON poi.schools TO poi;
GRANT DELETE ON poi.hospitals TO poi;
GRANT DELETE ON environmental.sssi TO environmental;
GRANT DELETE ON environmental.aonb TO environmental;
```

Checking the connection properties in the Database Manager in QGIS, now shows that the Environmental User has all **4 Privileges** that they need – including **DELETE.**



Now if we re-connect via QQGIS, the Users have the privileges to **DELETE** records from their spatial tables as required.





Having done quite a bit of googling to find these options, there is definitely more that can be done to configure your access rights to Tables in your PostGIS database. So, keep an eye out for **Part 2** where we will likely explore:

- Revoking Privileges
- Options for Remote Access to your PostGIS Instance