ESERCITAZIONE

POSTGRESQL e POSTGIS

Luca Lanteri - Rocco Pispico
GFOSS.IT

I dati dell'esercitazione

Dati ISTAT (http://www.istat.it/it/archivio/104317)

- Limiti provinciali del Piemonte (province poligonale)
- Località abitate (localita poligonale)
- Tabella TIPO_LOC Tipologia di località 2001/2011. (tipo_localita.csv)
  Il campo può assumere i seguenti valori:
  - 1. centro abitato
  - 2. nucleo abitato
  - 3. località produttiva
  - 4. case sparse

Da Geoportale Regione Piemonte (http://www.geoportale.piemonte.it/cms/)

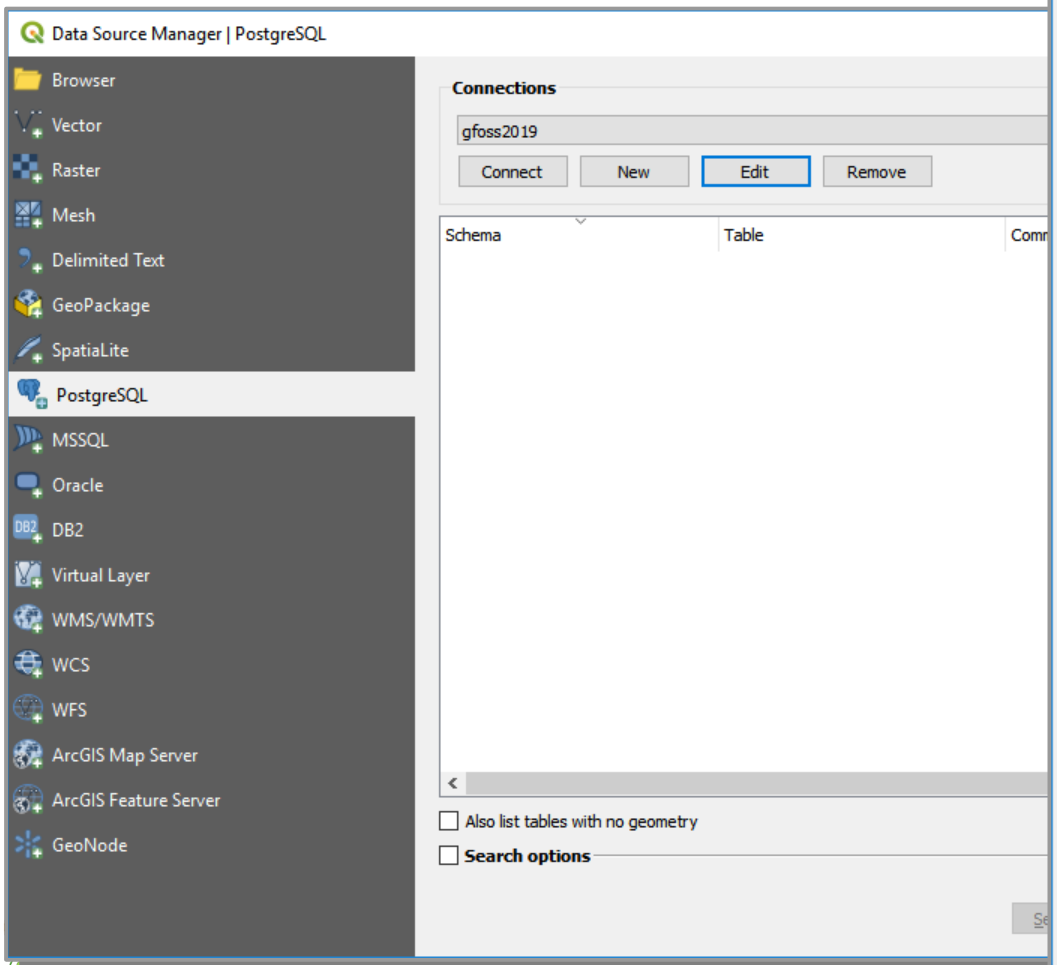
- Limiti amministrativi comunali (comuni poligonale)
- Una selezione della viabilità principale (autostrade lineare)

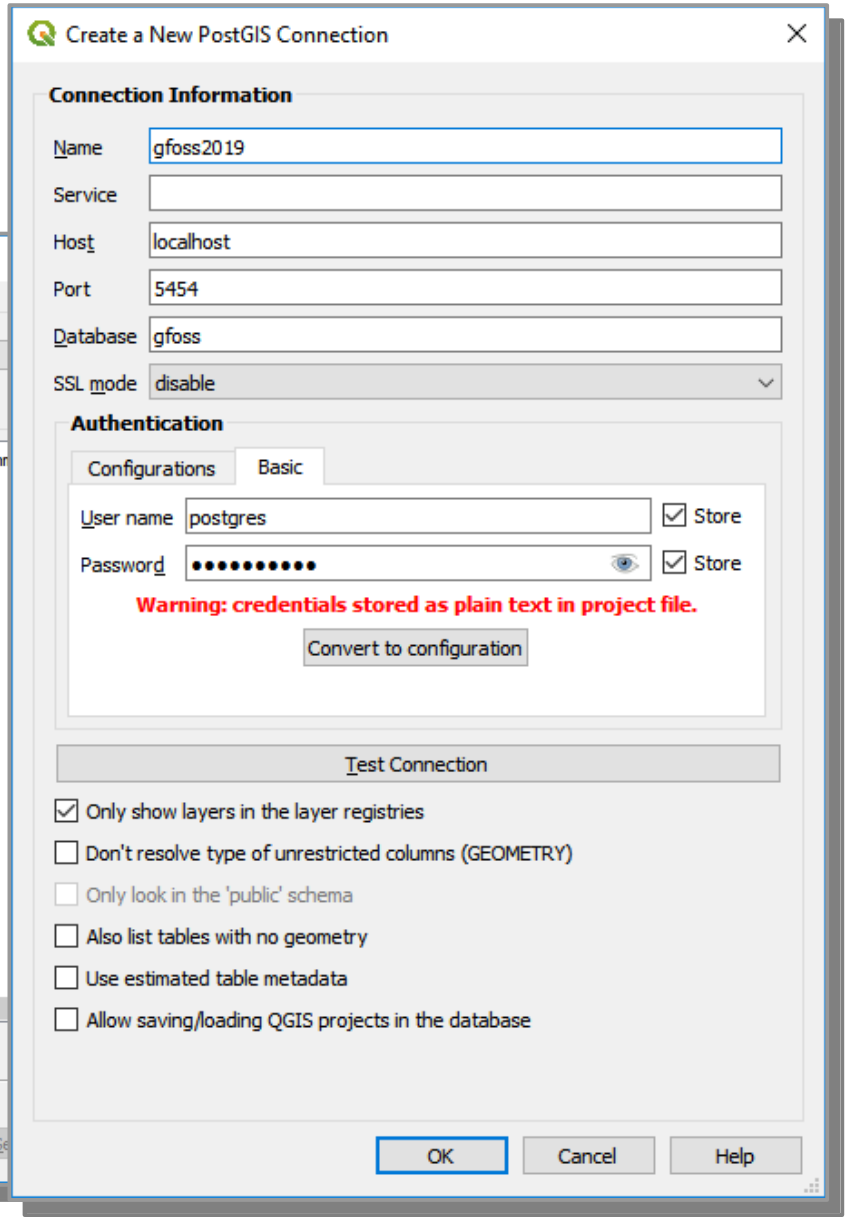
I dati sono disponibili in formato Shapefile e in Geopackage

Connessione al database

Creare una connessione con il database

Dal Data Source Manager scegliere PostgreSQL quindi configurare i parametri di connessione

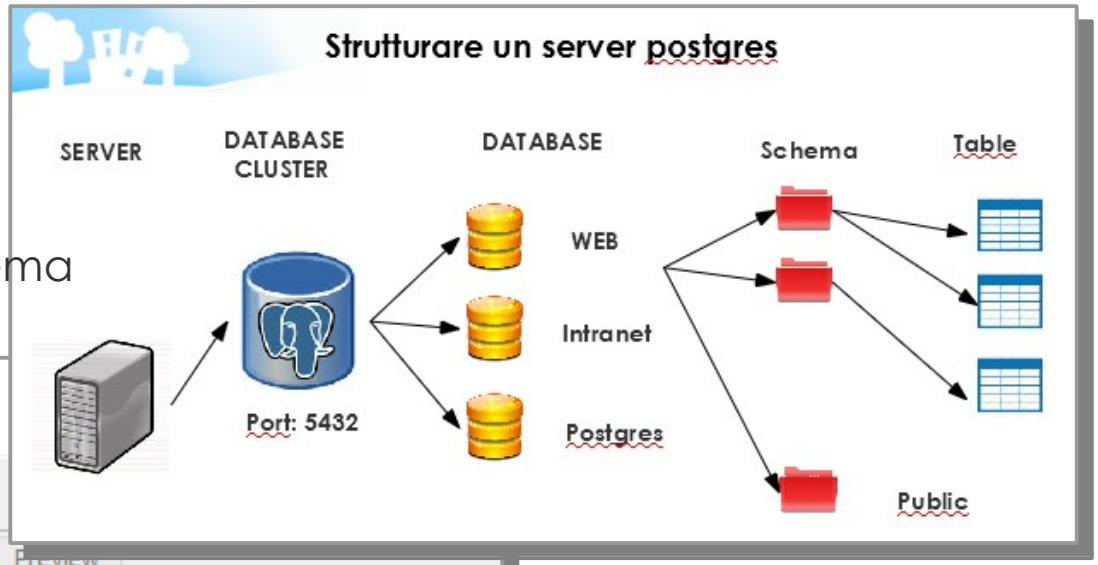


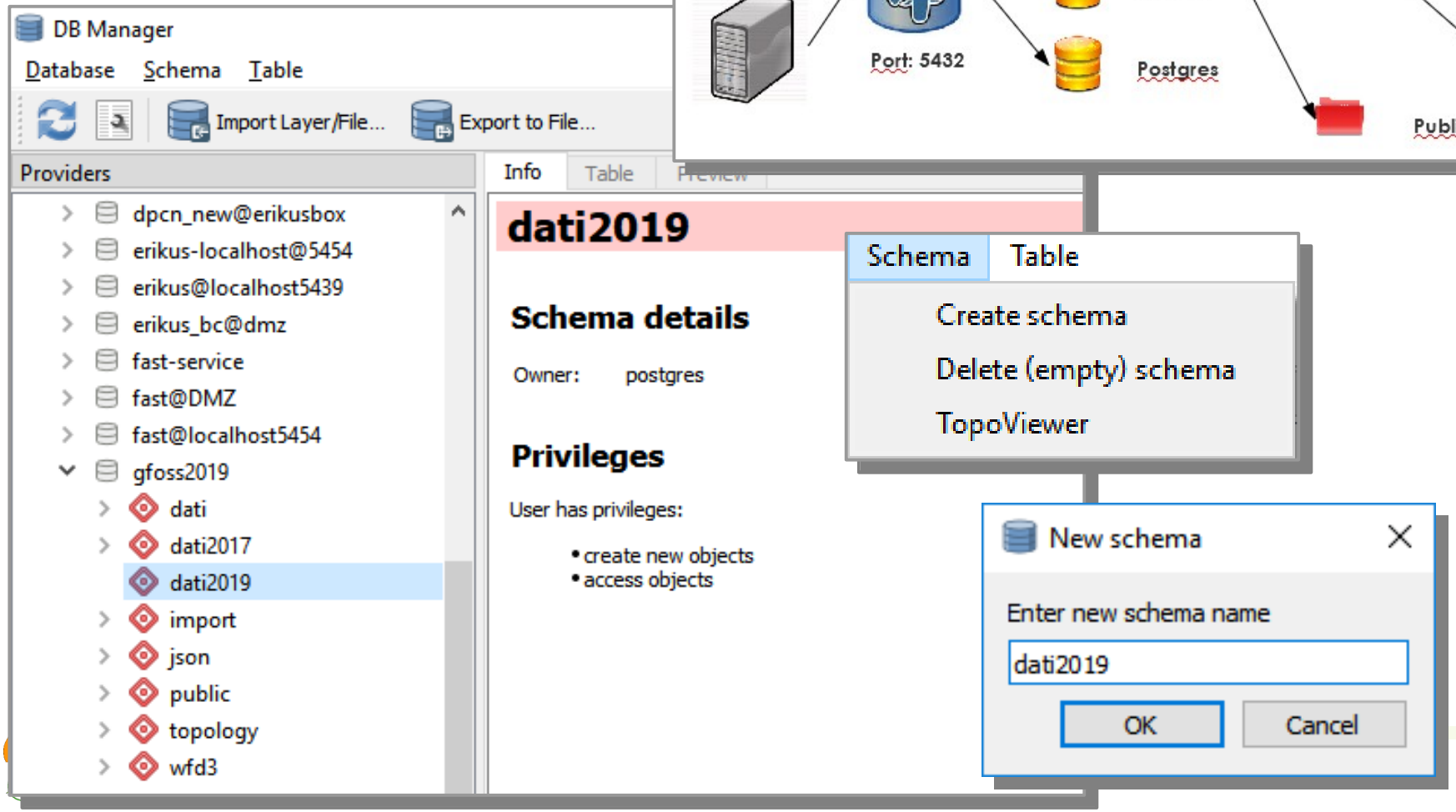


Organizzazione dei dati

Utilizzando DBManager create lo schema dati

Dal menu Schema → Create schema



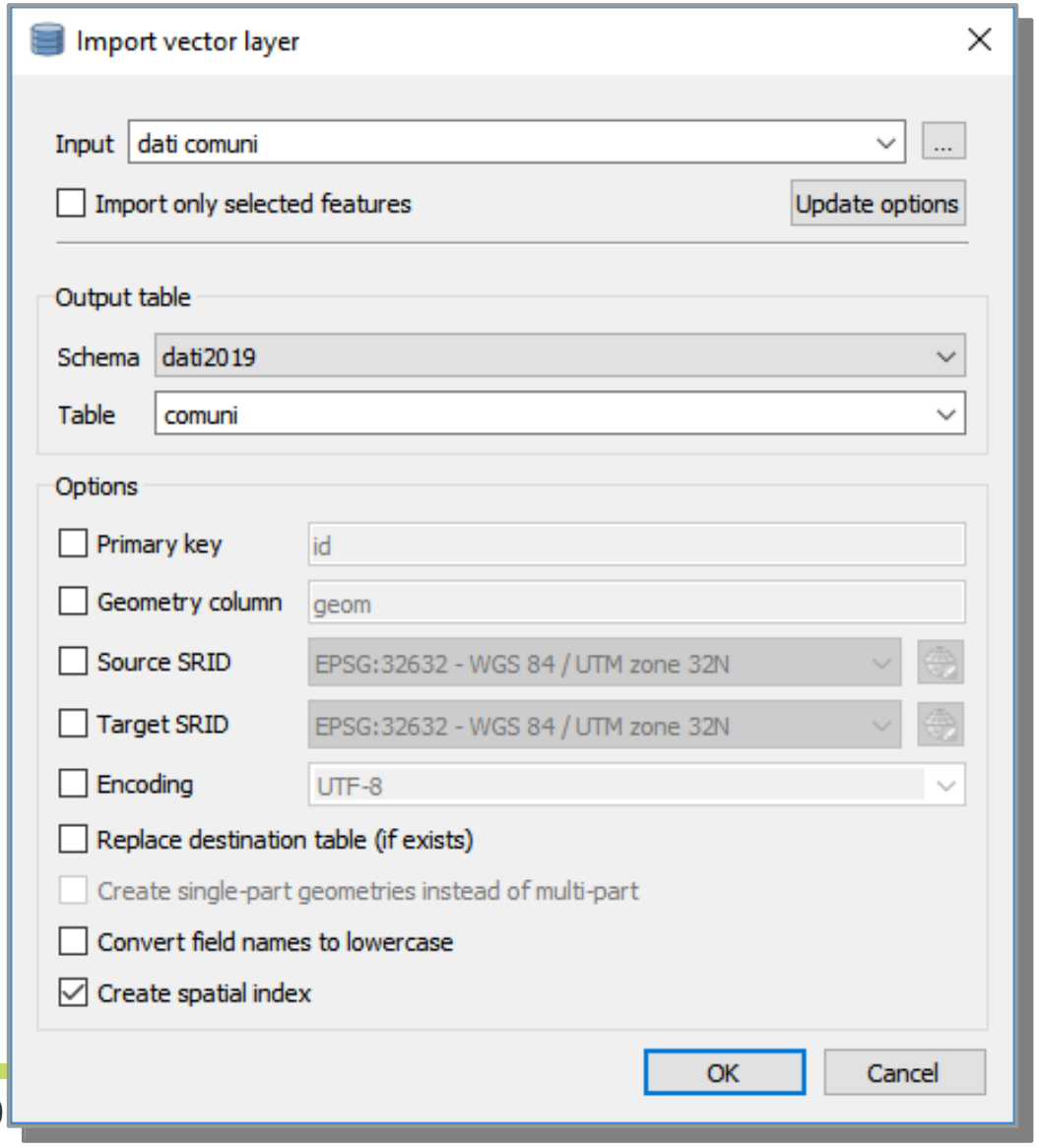


Esercizio 1 – caricare i dati in Postgres con DBManager

Utilizzando il DBManager caricare i layer nello schema

Importare i layer Province e Comuni nello schema dati

NOTA: verificate sistema di riferimento geografico e la presenza di una **primary key** e di un **indice spaziale**



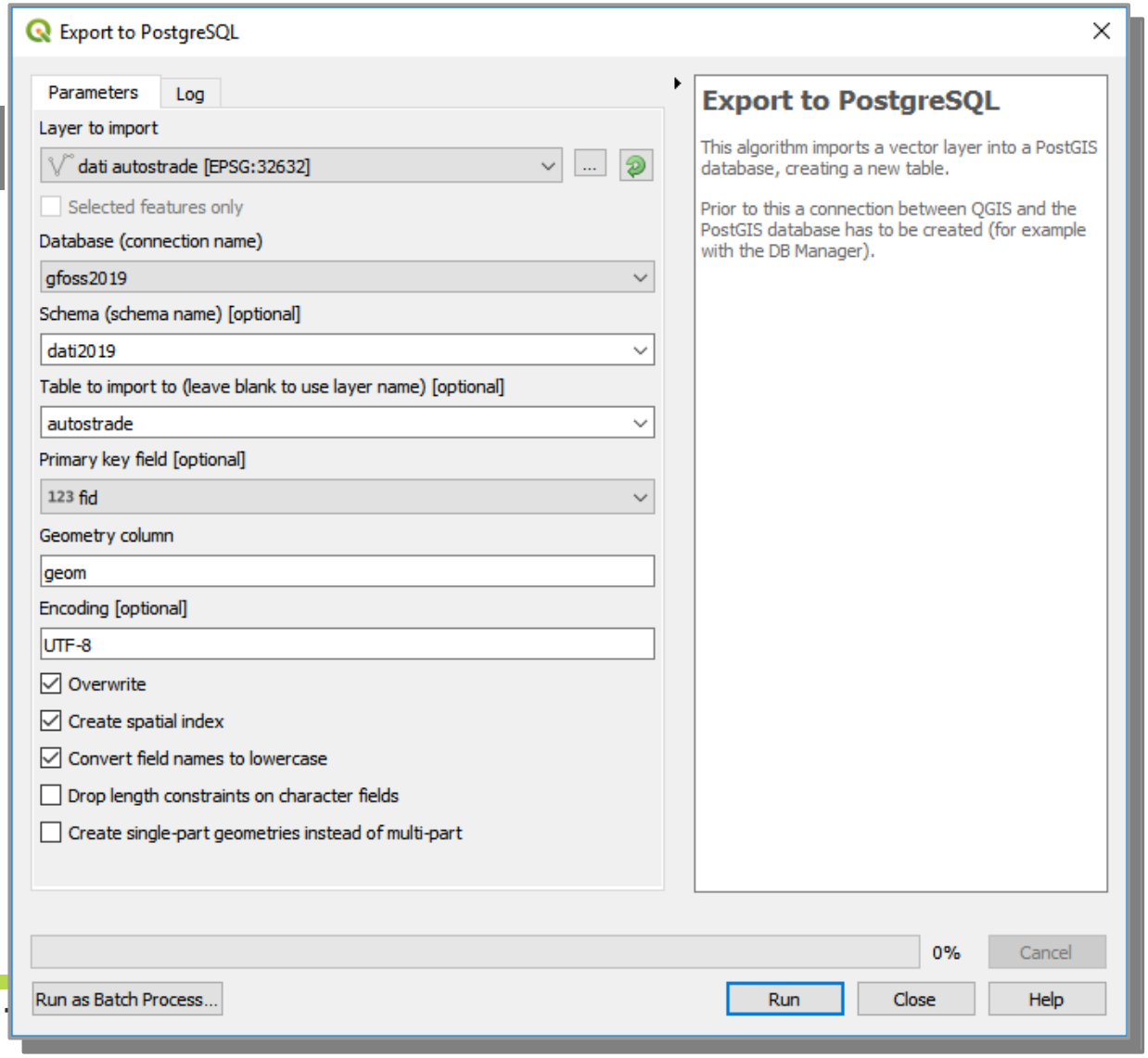
Esercizio 1 – caricare i dati in Postgres con Processing

Con la funzione di Processing

Export to PostgreSQL

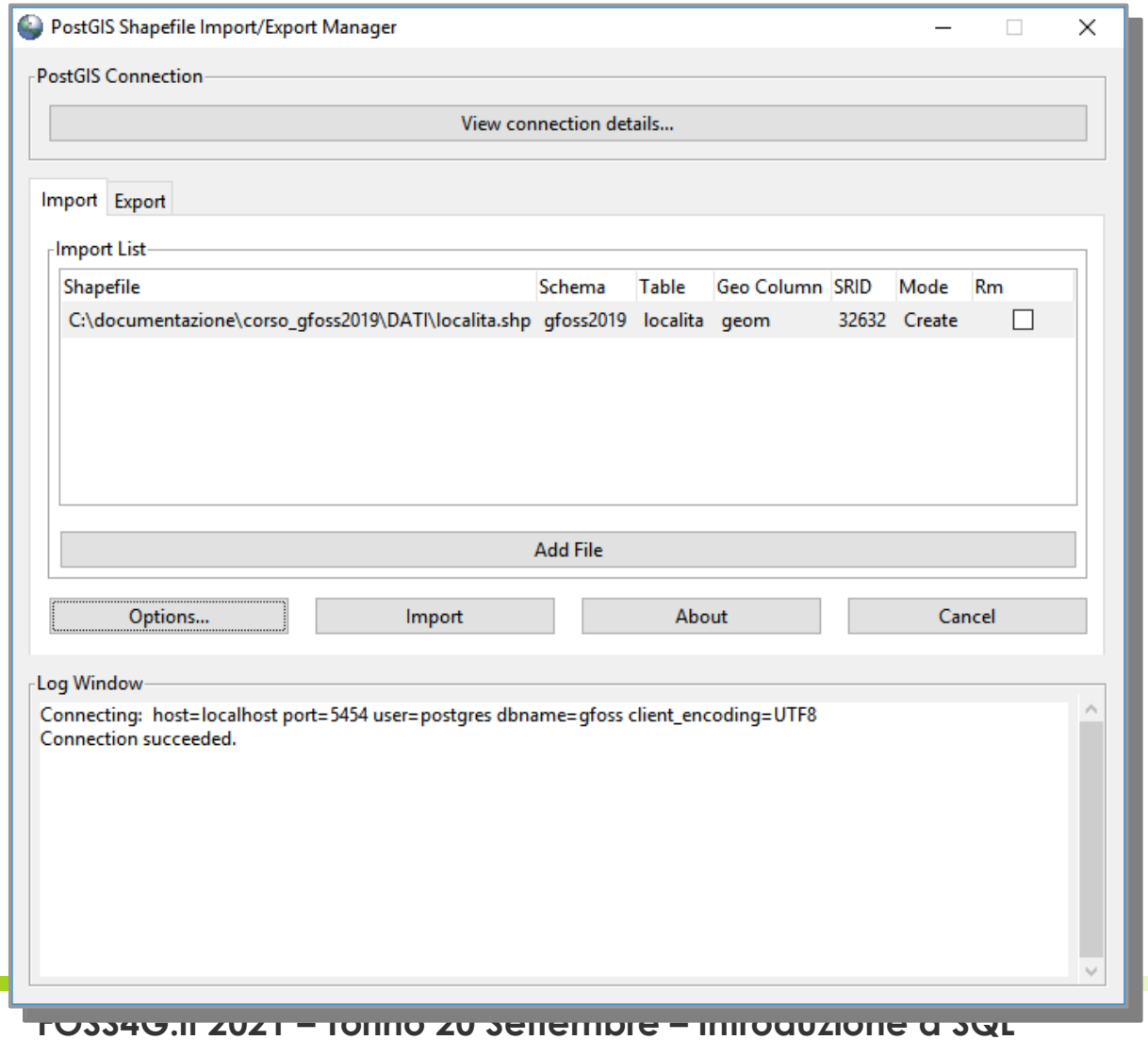
Importare i layer autostrade e localita nello schema dati

NOTA: verificate lo schema e la presenza di una **primary key** e di un **indice spaziale**



Esercizio 1 – altri metodi per caricare i dati in Postgres

Utilizzando shp2pgsql-gui



Esercizio 1 – altri metodi per caricare i dati in Postgres

Utilizzando shp2pgsql

ESEMPIO:
shp2pgsql -c -D -s 4269 -I shaperoads.shp myschema.roadstable > roads.sql
psql -d roadsdb -f roads.sql

https://postgis.net/docs/using_postgis_dbmanagement.html#shp2pgsql_usage

Utilizzando ogr2ogr

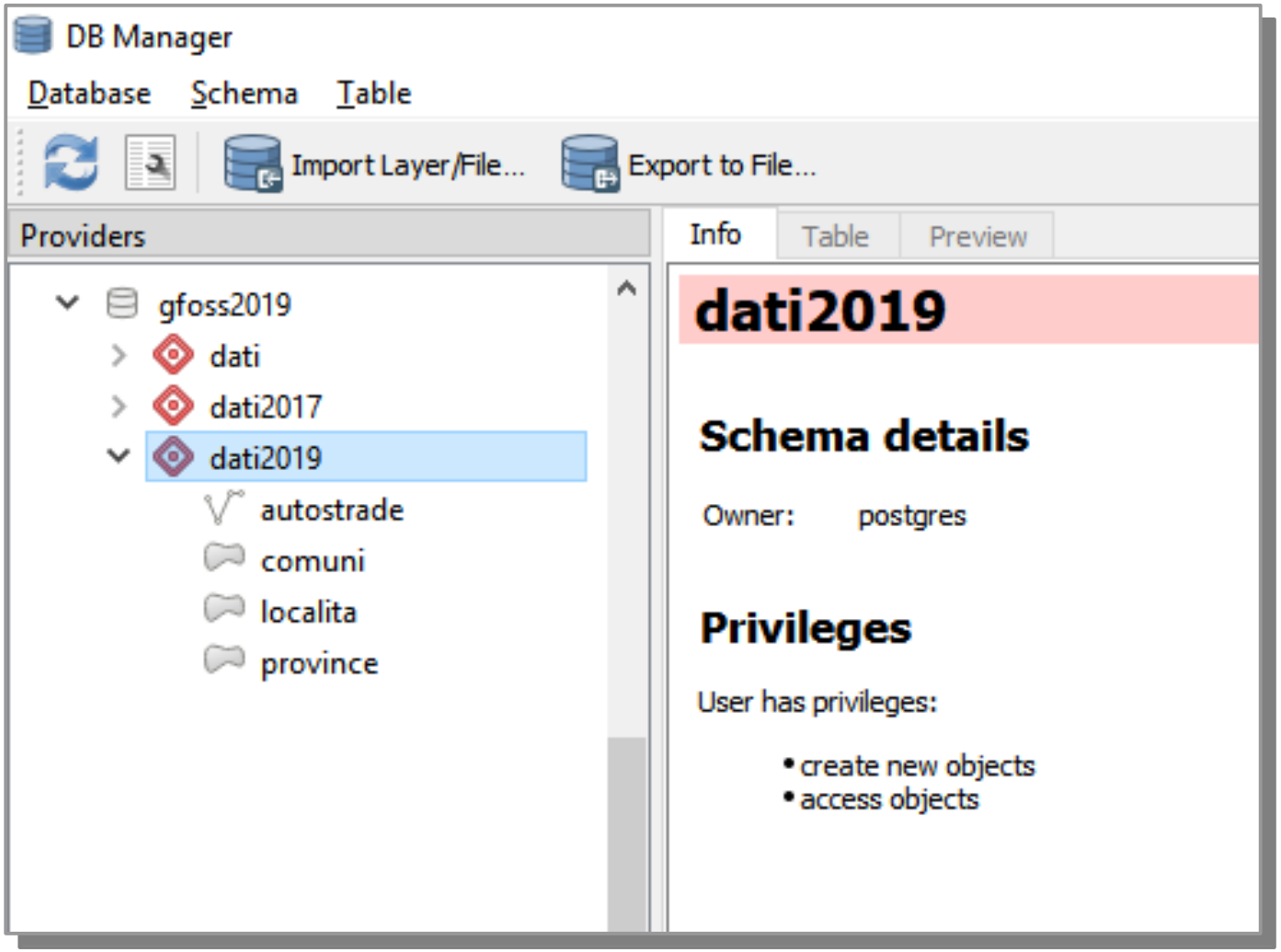
ESEMPIO:
ogr2ogr -f "ESRI Shapefile" mydata.shp PG:"host=myhost user=myloginname dbname=mydbname password=mypassword" "mytable"

https://www.gdal.org/ogr2ogr.html
https://www.bostongis.com/PrinterFriendly.aspx?content_name=ogr_cheatsheet

https://www.gdal.org/ogr_formats.html (42 formati gestiti)

Esercizio 1 – caricare i dati in Postgres con Processing

Alla fine delle operazioni lo schema dovrebbe apparire così



Esercizio 1 – DBManager

Info | Table | Preview

Info | Table | Preview

province

General info

Relation type:	Table
Owner:	postgres
Pages:	1
Rows (estimation):	8
Rows (counted):	8
Privileges:	select, insert, update, delete

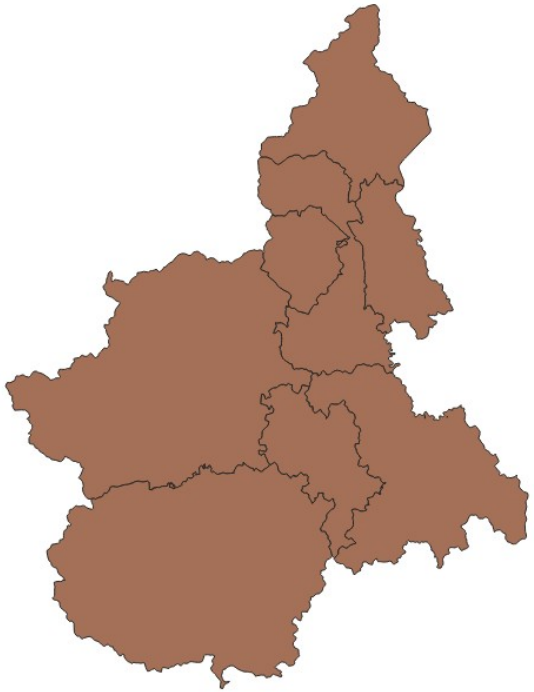
PostGIS

Column:	geom
Geometry:	MULTIPOLYGON
Dimension:	2
Spatial ref:	WGS 84 / UTM zone 32N (32632)
Extent:	(unknown) (find out)

Fields

#	Name	Type	Length	Null	Default
1	id	int4	4	N	nextval('dati2019.province_id_seq'::regclass)
2	geom	geometry (MultiPolygon,32632)		Y	
3	fid	int8	8	Y	
4	COD_REG	int4	4	Y	
5	COD_CM	int8	8	Y	
6	COD_PRO	int4	4	Y	
7	DEN_CMPRO	varchar (50)		Y	
8	PROVINCIA	varchar (50)		Y	
9	SIGLA	varchar (50)		Y	
10	FLAG_CMPRO	int4	4	Y	
11	SHAPE_Leng	float8	8	Y	
12	SHAPE_Area	float8	8	Y	

Constraints



	id	geom	fid	COD_REG	COD_CM	COD_PRO	DEN_CMPRO	PROVINCIA	SIGLA	FLAG_CMPRO	SHAPE_Leng
1	1	MULTIPOLYGON	1	1	0	2	Vercelli	Vercelli	VC	2	418133.4059
2	2	MULTIPOLYGON	2	1	0	3	Novara	Novara	NO	2	250242.386669
3	3	MULTIPOLYGON	3	1	201	1	Torino	Torino	TO	1	539532.779877
4	4	MULTIPOLYGON	4	1	0	4	Cuneo	Cuneo	CN	2	490394.849316
5	5	MULTIPOLYGON	5	1	0	6	Alessandria	Alessandria	AL	2	474880.850716
6	6	MULTIPOLYGON	6	1	0	5	Asti	Asti	AT	2	315561.373435
7	7	MULTIPOLYGON	7	1	0	103	Verbano-Cusio-...	Verbano-Cusio-...	VB	2	298908.472016
8	8	MULTIPOLYGON	8	1	0	96	Biella	Biella	BI	2	158026.165493

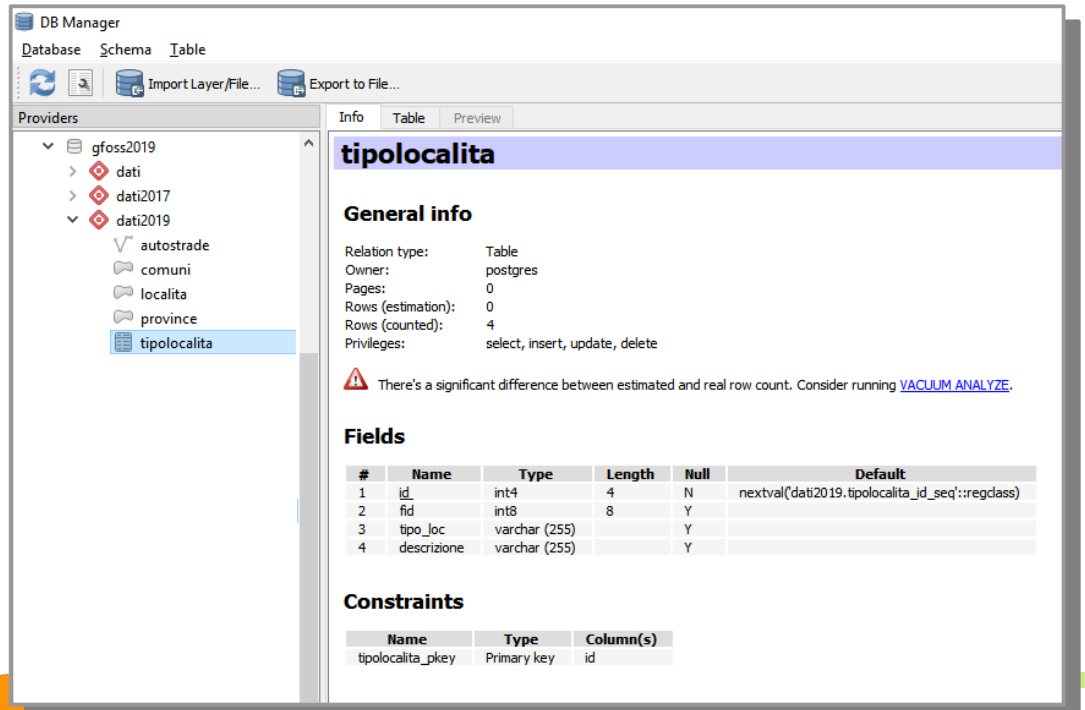
Esercizio 2 – caricare una tabella non geografica

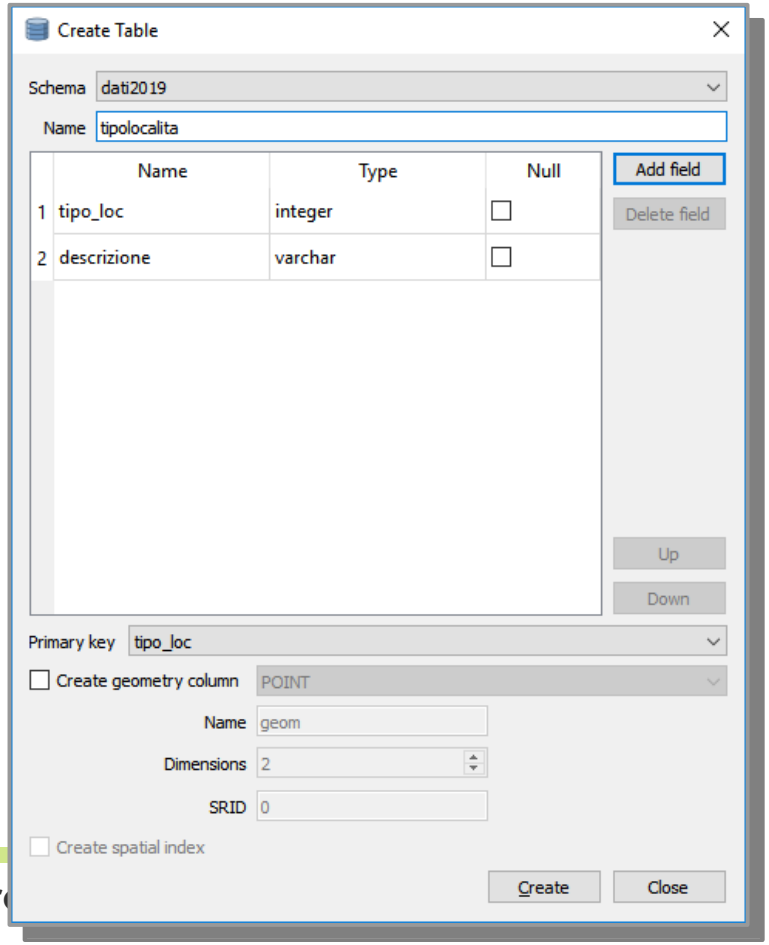
1) Utilizzando DBManager oppure Processing oppure shp2pgsql-gui caricare la tabella

tipo_localita.csv nello schema dati

2) strada alternativa crearla attraverso DBManager → Table → Create Table

Alla fine delle operazioni lo schema dovrebbe apparire così





Esercizio 3 – prima select

1) select che unisce alcuni dati della tavola localita e la decodifica del tipo località dalla tavola tipo_localita non geografica

```
SELECT
  tl.descrizione,
  localita.gid,
  localita.geom,
  localita.loc2011,
  localita.pro_com,
  localita.tipo_loc,
  localita.denominazi as nome
FROM
  dati2019.localita,
  dati2019.tipolocalita as tl
WHERE
  localita.tipo_loc = tl.tipo_loc;
```

Indicare i campi delle tavole che si desidera visualizzare.
È possibile impostare un alias al nome del campo.

Tavole da cui trarranno i dati.
È possibile impostare un alias della tavola

Indicare quali campi permettono l'unione (JOIN) tra le due tavole

2) limitare in fase di test

Durante la fase di test è consigliato impostare un limite massimo di risultati con la clausola: **Limit <numero di valori desiderati>** ad esempio **Limit 10**

Esercizio 3 – prima select

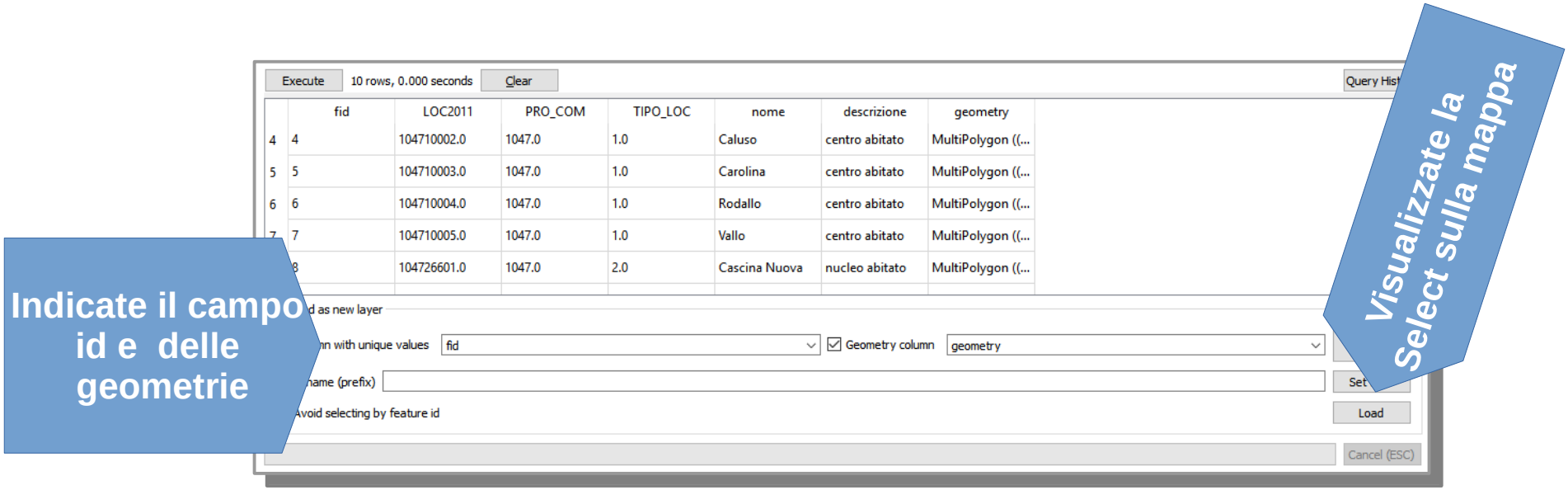
3) ha dato un errore?

Sostituire la clausola where con la seguente:

localita.tipo_loc::integer = tipo_localita.tipo_loc::integer;

4) caricare i dati nella vista con DBManager

Abilitate ***Load as new layer*** e visualizzate il risultato sulla mappa



Esercizio 4 – prima vista

1) come nell'esercizio precedente creiamo una query che prenda dati da due tavole e con qualche trasformazione

```
SELECT
  l.gid,
  ST_Centroid(l.geom) as geom,
  l.loc2011 as codice_localita,
  l.pro_com as codice_comune,
  l.tipo_loc,
  tl.descrizione as desc_tipo_localita,
  l.denominazi as nome_localita,
  l.popres as popolazione_residente,
  l.maschi as maschi,
  l.popres - l.maschi as femmine,
  l.famiglie as numero_famiglie,
  l.abitazioni as numero_abitazioni,
  l.edifici as numero_edifici,
  l.shape_area as superficie_in_mq
FROM
  dati2019.localita l,
  dati2019.tipolocalita tl
WHERE
  l.tipo_loc::integer = tl.tipo_loc::integer
limit 10;
```

Trasformiamo il layer poligonale in elementi puntuali con la funzione ST_Centroid

Colonna femmine = popres - maschi

https://it.wikipedia.org/wiki/Vista_(basi_di_dati)

Esercizio 4 – prima vista

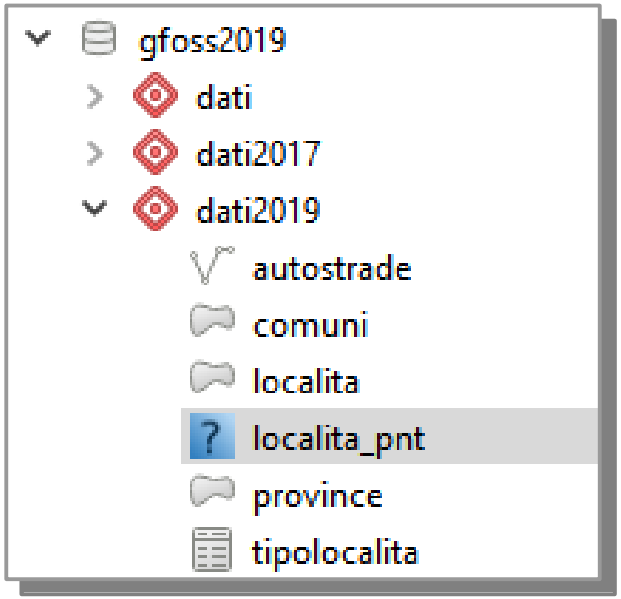
2) creare la vista

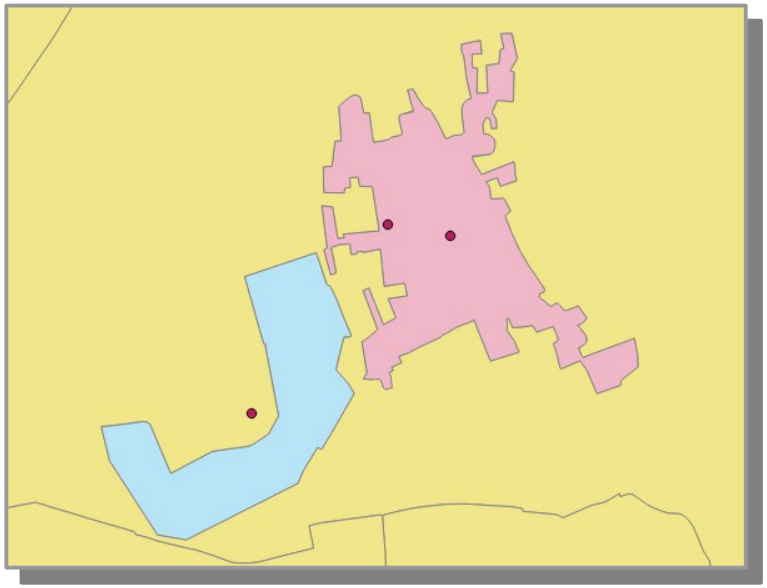
Aggiungere: **CREATE OR REPLACE VIEW dati2019.localita_pnt AS** prima della SELECT

Controllare che **LIMIT 10** sia stato rimosso

3) caricare la vista in QGIS

Va tutto bene? Ci sono anomalie?





Esercizio 4 – prima vista

4) creare una nuova vista

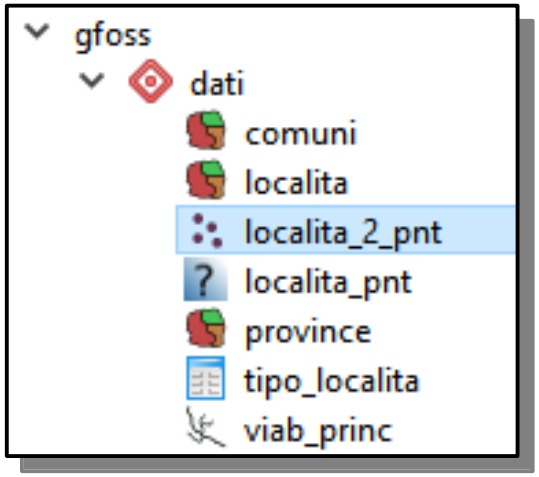
Sostituire la prima parte con la seguente:

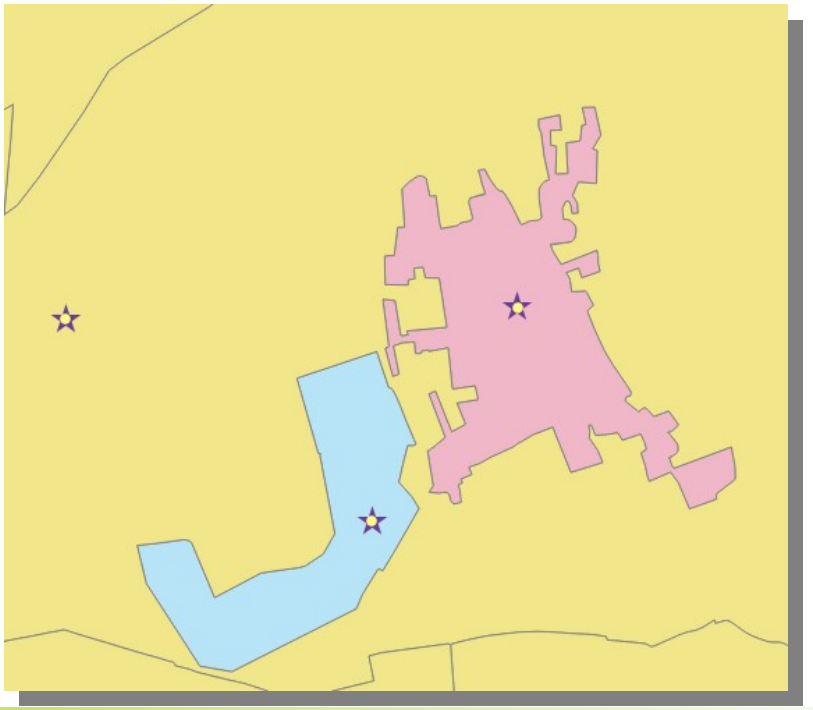
```
CREATE OR REPLACE VIEW dati2019.localita_2_pnt AS

SELECT
  l.gid,
  ST_pointonsurface(l.geom)::geometry(Point,32632) AS geom,
```

5) caricare la nuova vista in QGIS

Va meglio?





Esercizio 5 – il campo geometria

1) cosa c'è nel campo della geometria

```
select
 gid,
 geom,
 st_astext(geom)
from
  dati2019.localita_2_pnt
Limit 10
```

	id integer	geom geometry(Point,32632)	st_astext text
1	1	0101000020787F00007A09C04AA4781741AA957381B9	POINT(384553.072998188 5010150.02267973)
2	2	0101000020787F00005A6F203DB87E1741B0B451C1E9	POINT(384942.059694042 5010343.02061193)
3	3	0101000020787F00002711C3F5C739194130E87F8FA5	POINT(413297.990001934 5015190.24218182)
4	6	0101000020787F00009DCCA62D98211941926CB80F31	POINT(411750.044581601 5014724.24563135)
5	20	0101000020787F0000B26CFB8762991A41F0DF0C20D2	POINT(435800.632795046 5048136.50078581)
6	589	0101000020787F0000416112BD88A91641595F1F4A73	POINT(371298.184640426 4958669.15816482)
7	4	0101000020787F0000179AAF05E33A1941B2A5B6B8D3	POINT(413368.755552681 5017422.8861479)
8	5	0101000020787F00006C1797AE7F3A19411B7E14016E	POINT(413343.920498243 5011896.01687577)
9	66	0101000020787F000099F76367DC941641BBA07BEBED	POINT(369975.100967282 4999095.67942065)
10	7	0101000020787F00009A6EE57B6324194113386DBFC8	POINT(411928.870992401 5013282.9910412)

2) geometria lineare

```
select  fid, geom,  st_astext(geom)
from   dati2019.autostrade limit 10
```

Geom è in formato WKB
ST_Astext(geom) è in formato WKT

	id integer	geom geometry(LineString,4326)	st_astext text
1	1	0102000020E6100000020000000276E707DB3E8214063	LINESTRING(8.9544944 44.6137907,8.9551518 44.6131613)
2	2	0102000020E6100000050000000D33252EFA9841E40E3	LINESTRING(7.6295545 44.8362476,7.6297393 44.8363953,7.629925
3	3	0102000020E610000004000000EFD57DB6C4E421407A	LINESTRING(8.9468133 44.6340522,8.9465717 44.6338984,8.945587
4	4	0102000020E6100000030000000038D73043E38520409C	LINESTRING(8.2614995 46.0229861,8.2613573 46.0229288,8.261238
5	5	0102000020E610000006000000 7F8FB05EFBE421405C	LINESTRING(8.9472303 44.619496,8.9469877 44.6192808,8.9468711
6	6	0102000020E610000004000000B9DE365321862040C9	LINESTRING(8.261973 46.0231614,8.2617489 46.0230814,8.2615619
7	7	0102000020E610000016000000AB01EF891A911E404D	LINESTRING(7.6417028 44.8437602,7.6417122 44.8437165,7.641736
8	8	0102000020E61000000E000000F874D080D5E4214050	LINESTRING(8.9469414 44.6184442,8.9472024 44.6180776,8.948305
9	9	0102000020E6100000310000002DC7D056CA8B204062	LINESTRING(8.2730281 46.0232463,8.2729292 46.023243,8.272898
10	10	0102000020E6100000050000000DA7EE8386DE8214050	LINESTRING(8.9539583 44.6401032,8.9531346 44.6396265,8.952649

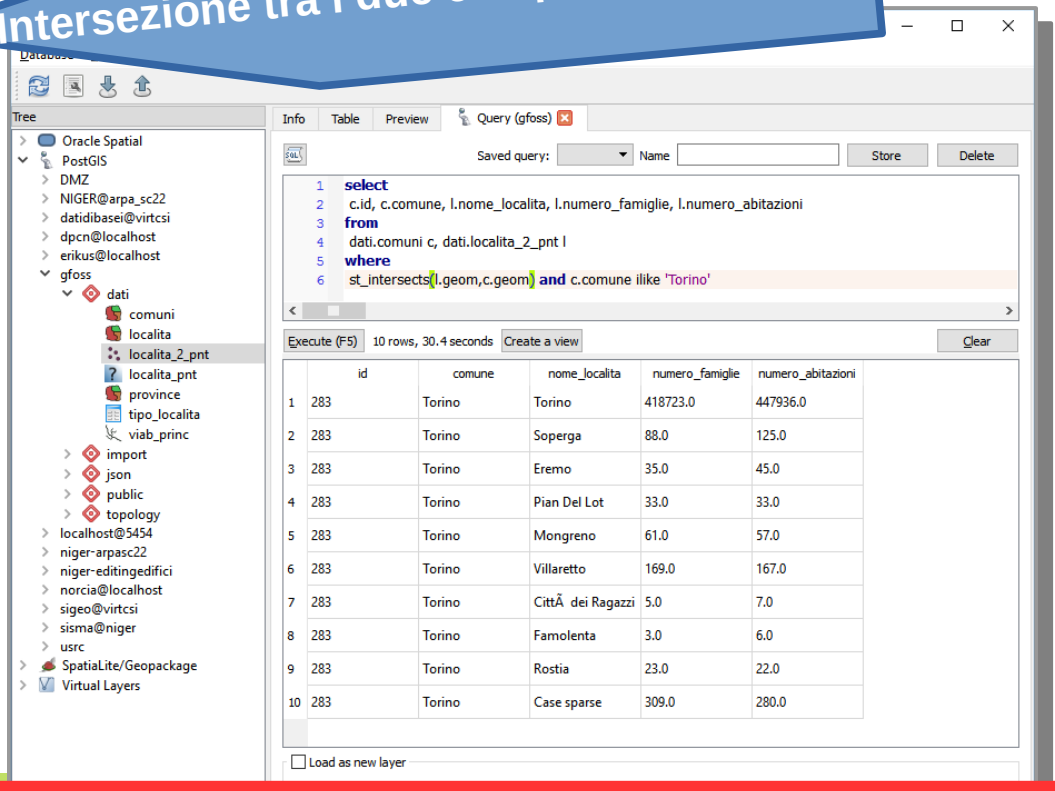
Esercizio 6 – Select con intersect

1) intersezione tra layer poligonale e puntuale

```
select
      c.id,      c.comune_nom,      l.nome_localita,      l.numero_famiglie,
l.numero_abitazioni
from
 dati2019.comuni c, dati2019.localita_2_pnt l
where
 st_intersects(l.geom,c.geom)
 and c.comune_nom ilike 'Torino';
```

Intersezione tra i due campi geometria

Limitiamo il set di dati al comune di Torino



http://postgis.net/docs/reference.html#Spatial_Relationships_Measurements

Esercizio 7 – Select con group by

1) quante frazioni ci sono per ogni comune?

```
select
 c.comune_nom, count(*)
from
  dati2019.comuni c, dati2019.localita l
where
l.geom && c.geom
and st_intersects(c.geom,st_pointonsurface(l.geom))
and c.cod_pro = 5
group by c.comune_nom;
```

ST_Intersects controlla l'effettiva intersezione tra geometrie
l'operatore && verifica il map extent delle geometrie
Unire le due voci rende più veloce la query

La clausola GROUP BY indica su quale (o quali) campi aggregare il conteggio

	comune	count
1	Pino d'Asti	2
2	Cortazzone	10
3	Isola d'Asti	10
4	Coazzolo	2
5	Piea	5
6	Aramengo	10
7	Tigliole	28

Esercizio 7 – Select con group by e order by

2) **mettiamo in ordine crescente**

Aggiungere all'ultima riga: **order by c.comune**

	comune	count
1	Agliano Terme	15
2	Albugnano	5
3	Antignano	8
4	Aramengo	10
5	Asti	73
6	Azzano d'Asti	3
7	Baldichieri d'Asti	8

3) **oppure in ordine decrescente**

order by c.comune **desc**

```
1  select
2   c.comune, count(*)
3  from
4    dati.comuni c, dati.localita l
5  where
6  l.geom && c.geom
7  and st_intersects(c.geom,st_pointonsurface(l.geom)) and c.cod_pro = 5
8  group by c.comune
9  order by c.comune
```

Execute (F5) 118 rows, 17.3 seconds Create a view Clear

	comune	count
1	Agliano Terme	15
2	Albugnano	5
3	Antignano	8
4	Aramengo	10
5	Asti	73
6	Azzano d'Asti	3
7	Baldichieri d'Asti	8

	comune	count
1	Vinchio	4
2	Villanova d'Asti	16
3	Villafranca d'Asti	11
4	Villa San Secondo	3
5	Vigliano d'Asti	9
6	Viarigi	12
7	Viale	3

Esercizio 8 – Select con group by – seconda parte

1) quante persone abitano in ogni comune?

```
select
 c.comune_nom, sum(l.popolazione_residente) as pop_residente,
 sum(l.maschi) as tot_maschi,
 sum(l.femmine) as tot_femmine
from
  dati2019.comuni c, dati2019.localita_2_pnt l
where
 l.geom && c.geom
 and st_intersects(c.geom,st_pointonsurface(l.geom)) and c.cod_pro = 5
group by c.comune_nom
order by sum(l.popolazione_residente) desc
```

Somma dei campi

Ordinamento per popolazione in ordine decrescente

	comune	pop_residente	tot_maschi	tot_femmine
1	Asti	73926.0	34974.0	38952.0
2	Canelli	10569.0	5080.0	5489.0
3	Nizza Monferrato	10355.0	4972.0	5383.0
4	San Damiano d'...	8373.0	4153.0	4220.0
5	Costigliole d'Asti	5969.0	2937.0	3032.0
6	Villanova d'Asti	5774.0	2818.0	2956.0
7	Castagnole dell...	3784.0	1847.0	1937.0

Esercizio 9 – Select con group by – seconda parte

2) superficie della provincia?

```
select
   p.provincia,   round((sum(ST_area(c.geom))/1000000)::numeric,2)   as
sup_kmq
from
  dati2019.comuni c, dati2019.province p
where
p.geom && c.geom
and st_intersects(c.geom,p.geom)
group by p.provincia
order by p.provincia
```

	provincia	sup_kmq
2	Alessandria	4214.75
3	Asti	2549.43
4	Biella	1366.42
5	Cuneo	7702.89
6	Novara	1798.54
7	Verbano-Cusio...	2724.72
8	Vercelli	3364.55

Somma delle superfici ST_Area(geom) restituisce la superficie del poligono

Raggruppamento e ordinamento per provincia

Esercizio 10 – Select con group by – terza parte

1) **lunghezza delle strade per comune?**

```
select
 c.comune_nom, el_str_sed, count(*) as numero_archi
,sum(st_length(st_intersection(c.geom,v.geom))/1000) as lun_km
from
  dati2019.comuni c, dati2019.autostrade v
where
v.geom && c.geom
and st_intersects(c.geom,v.geom)
group by c.comune_nom,el_str_sed
order by c.comune_nom,el_str_sed
```

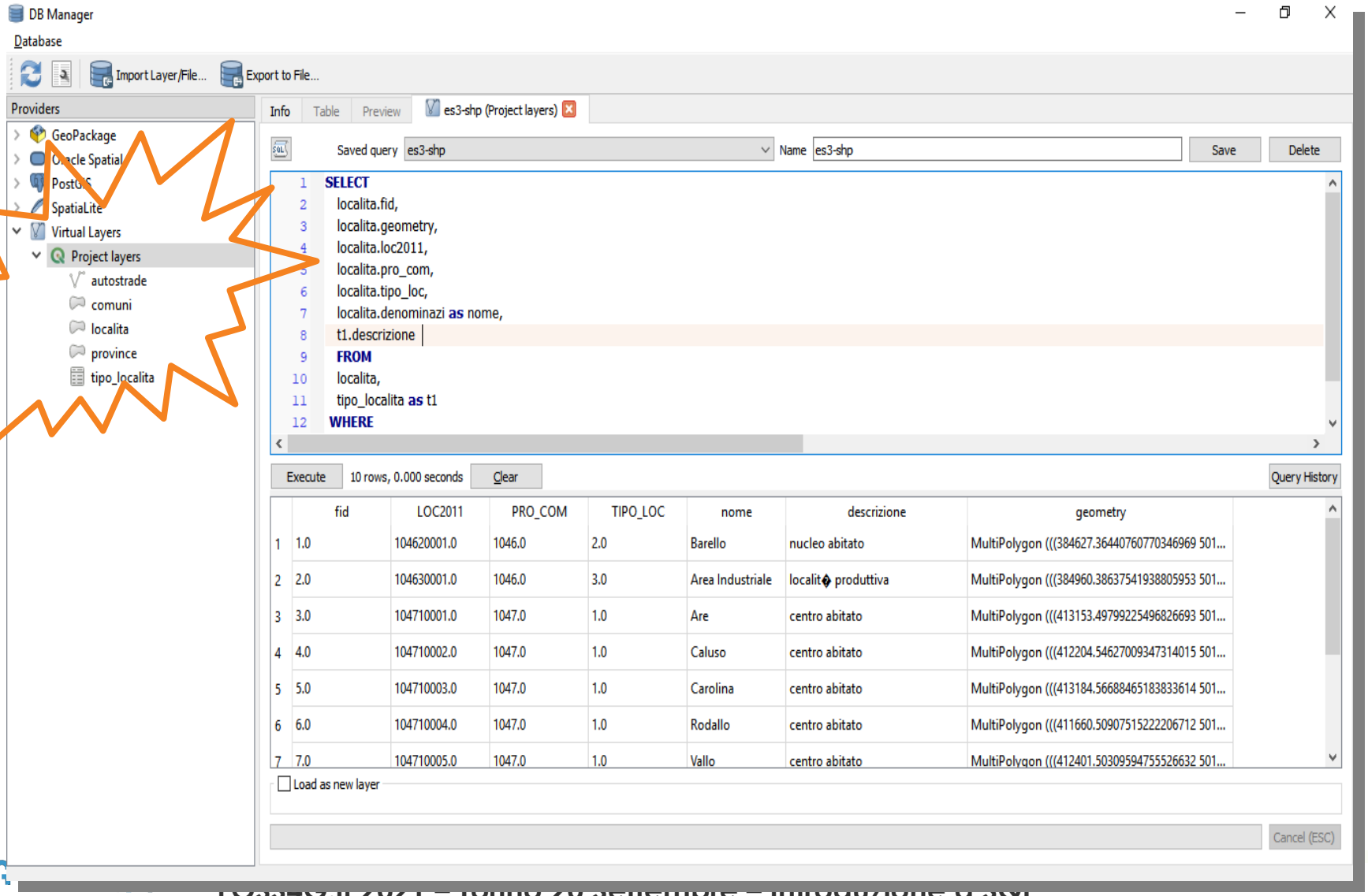
Somma delle lunghezze ST_Length(geom) restituisce la lunghezza della ST_Intersection tra le geometrie dei comuni e della viabilità

Airasca	a raso	12	9.72314799541377
Airasca	su ponte/viadotto/cavalcavia	4	0.147465576363129
Alba	a raso	6	1.36465934714501
Albiano d'Ivrea	a raso	8	4.23342987621825
ALESSANDRIA	a raso	76	62.9304977441036
ALESSANDRIA	su ponte/viadotto/cavalcavia	26	8.34765495540765
Alice Castello	a raso	30	23.9838634383668
Alice Castello	in galleria		
Alice Castello	su ponte/viadotto/cavalcavia		
Alpignano	a raso		
Arquata Scrivia	a raso	8	

Raggruppamento e ordinamento per comune e per tipologia di sede stradale

Virtual Layers – Project layers

Utilizzare le query in formato SQL per **tutti** i layer presenti nel progetto con pochi adattamenti. Non è veloce quanto PostgreSQL.



Esercizio 11 – Distanze

Trovare i distributori più vicini in linea d'aria alle sedi di Arpa Piemonte.

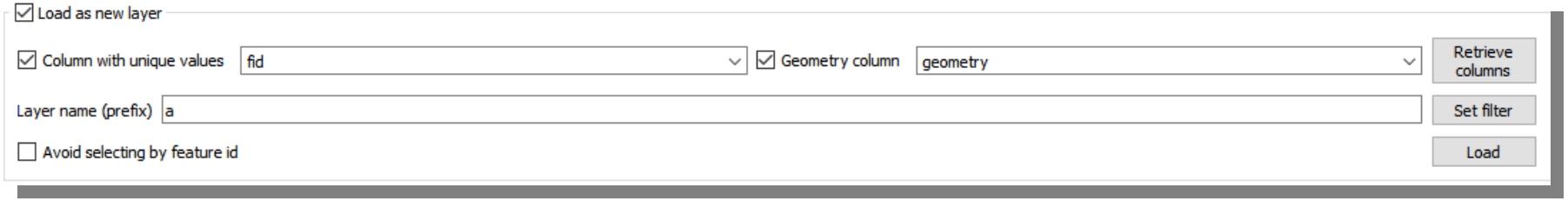
1) Create un nuovo progetto

2) Caricate i layer: puntivendita e sediarpa presenti nel file es11.gpkg

3) in DBManager impostate questa select:

select s.fid*1000+p.fid id, s.fid, s.sede, s.indirizzo, p.fid, p.super, p.gasolio, p.gpl, p.self, p.ditta, st_distance(s.geometry, p.geometry) d,
st_geomfromtext('LINESTRING (' || st_x(s.geometry) || ' ' || st_y(s.geometry) || ',' || st_x(p.geometry) || ' ' || st_y(p.geometry) || ')',32632) geometry
from "es11 sediarpa" s, "es11 puntivendita" p
where st_distance(s.geometry, p.geometry) <= 3000

4) caricare il layer




i sogni
del Signor
ROSSI