

Esercitazione su PostGIS

Claudio Rocchini – Istituto Geografico Militare

Nota su questi lucidi

- ▶ Le query SQL da eseguire sono sempre scritte con il carattere **Courier**
- ▶ PowerPoint trasforma talvolta il primo apicetto di una frase in apicetto invero. In SQL bisogna sempre usare l'apicetto diritto (l'apostrofo).
- ▶ Si ricorda che per eseguire una query bisogna:
 - ▶ Aprire pgAdminIII
 - ▶ Connettersi al server giusto
 - ▶ Selezionare un database
 - ▶ Aprire SQL editor
 - ▶ Digitare la query nella finestra di sinistra
 - ▶ Premere il pulsante Play (execute)

▶ 3

PostGIS - Claudio Rocchini - IGM

Introduzione

- ▶ *Postgres* con *PostGIS* è un database server professionale Open Source e Gratuito
- ▶ Il supporto spaziale è veramente completo e fa concorrenza a software commerciali ben più costosi (es. *Oracle*)
- ▶ In questa esercitazione faremo una breve introduzione al supporto spaziale di *Postgres*.
- ▶ **Prerequisiti:**
 - ▶ Cenni sulle basi di dati
 - ▶ SQL
 - ▶ Interfaccia di *Postgres*

▶ 2

PostGIS - Claudio Rocchini - IGM

Supporto Spaziale di Postgres

Vediamo in questa sezione cosa effettivamente PostGIS aggiunge a Postgres

▶ 4

PostGIS - Claudio Rocchini - IGM

Supporto Spaziale a Postgres

- ▶ PostGIS aggiunge il supporto geografico a Postgres
- ▶ Tale supporto si compone di varie parti e si basa principalmente su standard Open GIS Consortium (Oracle invece, che è più vecchio, ha dovuto sviluppare un suo standard).
- ▶ Principali componenti:
 - ▶ Un nuovo tipo di dato: *GEOMETRY*
 - ▶ Tabelle di supporto: *spatial_ref_sys* e *geometry_columns*
 - ▶ Funzioni SQL di supporto (circa 700)
 - ▶ Tool esterni (es. *pgsql2shp* e *shp2pgsql*) che vedremo durante l'importazione di dati
- ▶ **Nota: Postgres non ha di per sé un visualizzatore**

▶ 5

PostGIS - Claudio Rocchini - IGM

spatial_ref_sys: introduzione

- ▶ E' una tabella del sistema spaziale: memorizza l'elenco dei sistemi di riferimento supportati
- ▶ Si basa principalmente sullo standard EPSG (<http://www.epsg.org>)
- ▶ Per ogni sistema previsto è indicato:
 - ▶ il codice numerico
 - ▶ la sorgente (si solito EPSG)
 - ▶ la definizione testuale (equivale al testo contenuto nei file prg)
 - ▶ la definizione proj4
- ▶ **Nota: proj4 è una libreria di trasformazione di coordinate utilizzata da PostGIS**

▶ 7

PostGIS - Claudio Rocchini - IGM

GEOMETRY

- ▶ I dati di una colonna di database sono associati ad un tipo (es. *INTEGER*, *CHARACTER*, *BOOLEAN*)
- ▶ Il supporto spaziale introduce un nuovo tipo di dato: *GEOMETRY*, questo tipo è un tipo di dato ad oggetti (complesso)
- ▶ Questo tipo di dato contiene la geometria di un singolo oggetto geografico e eventualmente il sistema di riferimento associato (codice SRID)
- ▶ Il tipo è multiforme:
 - ▶ può contenere dati a 2, 3 o 4 dimensioni (X,y,z e il campo M)
 - ▶ Vari tipi di geometrie: punti, linee, aree, curve, etc.

▶ 6

PostGIS - Claudio Rocchini - IGM

spatial_ref_sys: contenuto

- ▶ Esempio lo srId 4326 corrisponde a coordinate geografiche WGS84 (GPS)
- ▶ Ecco la definizione testuale:


```

geogcs |
"WGS_84",
DATUM |
    "WGS_1984",
    SPHEROID |
        "WGS84",
        6378137, 298, 257223563,
        AUTHORITY["EPSG","7030"]
    ],
    AUTHORITY["EPSG","6326"]
1,
PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],
UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],
AUTHORITY["EPSG","4326"]

```
- ▶ SRID 32632 e 32633 corrispondono a UTM WGS84 fusi 32 e 33 Nord.
- ▶ SRID 23032 e 23033 corrispondono a UTM ED50 fusi 32 e 33 Nord.

▶ 8

PostGIS - Claudio Rocchini - IGM

spatial_ref_sys: estensione

- ▶ Se un sistema di riferimento non è previsto può essere aggiunto alla tabella, definendo l'opportuna stringa proj4, ovviamente il sistema non sarà standard
- ▶ Ad esempio Fuso Italia, usato per i cataloghi web, (una media fra fuso 32 e fuso 33) può essere definito come:
 - ▶ SRID: 9000;
 - ▶ auth_name: "IGM";
 - ▶ auth_srid: 90000;
 - ▶ proj=tmmerc +lat_0=0 +lon_0=12 +k=0.9985 +x_0=7000000.0 +y_0=0 +ellps=WGS84 +datum=WGS84 +units=m +no_defs"
- ▶ Postgres è in grado di utilizzare correttamente i dati geografici con sistemi definiti dall'utente.

▶ 9

PostGIS - Claudio Rocchini - IGM

geometry_columns

- ▶ Questa è la seconda tabella di sistema
- ▶ Contiene i "metadati" minimi per tutte le colonne geometriche:
 - ▶ schema: nome dello schema che contiene la tabella
 - ▶ table_name: nome della tabella geometrica
 - ▶ geometry_column: nome della colonna di tipo GEOMETRY
 - ▶ coord_dimension: numero dimensioni (2,3, o 4)
 - ▶ srid: codice del sistema di riferimento, può essere -1 = non definito
 - ▶ type: sottotipo geometrico; rappresenta un vincolo ai tipi geometrici contenuti. Può valere GEOMETRY=nessun vincolo

▶ 11

PostGIS - Claudio Rocchini - IGM

Un inciso: Roma40

- ▶ Alcuni sistemi di riferimento non possono essere trattati in modo analitico
- ▶ Ad esempio Roma40, ha bisogno (per una sua eventuale trasformazione), di griglie di trasformazione punto punto, che memorizzano la differenza di coordinate.
- ▶ L'IGM vende queste griglie di trasformazione a blocchi
- ▶ Proj4 (e quindi Postgres) supportano le griglie di trasformazione nel formato standard NAD (nv2)
- ▶ Ad esempio la seguente definizione costruisce un sistema geografico Roma40 (avendo i grigliati opportuni):
 - ▶ +proj=longlat +ellps=WGS84 +to +proj=longlat +ellps=WGS84 +nadgrids=/wgs84_to_roma40.gsb

▶ 10

PostGIS - Claudio Rocchini - IGM

geometry_columns dal sito WEB IGM

oid	table_schema	table_name	geometry_name	coord_dimension	srid	type
1	45694	catello	catello	2	4326	MULTIPOLYGON
2	45698	catello	catello	2	9000	MULTIPOLYGON
3	45702	catello	catello	2	4326	MULTIPOLYGON
4	45696	catello	prodoto_gps	2	9000	MULTIPOLYGON
5	45696	catello	prodoto_label	2	9000	MULTIPOLYGON
6	45713	catello	11000	2	9000	POLYGON
7	45715	catello	1250	2	9000	POLYGON
8	45714	catello	500	2	9000	POLYGON
9	45695	catello	salvata	2	4326	MULTIPOLYGON
10	45704	catello	salvata	2	9000	MULTIPOLYGON
11	45704	catello	salvata_k	2	9000	POINT
12	45706	catello	funco	2	9000	POINT
13	45709	catello	funco2	2	9000	POINT
14	45710	catello	funco3	2	9000	POINT
15	45711	catello	funco4	2	9000	POINT
16	45712	catello	funco5	2	9000	POINT
17	45703	catello	funco6	2	9000	MULTIPOLYGON
18	45700	catello	funco7	2	9000	MULTIPOLYGON
19	45699	catello	funco8	2	9000	MULTIPOLYGON
20	45708	catello	funco9	2	9000	MULTIPOLYGON
21	45702	catello	funco10	2	9000	MULTIPOLYGON
22	45707	catello	funco11	2	9000	MULTIPOLYGON
23	45705	catello	funco12	2	9000	MULTIPOLYGON
24	665133	catello	funco13	2	9000	MULTIPOLYGON

▶ 12

PostGIS - Claudio Rocchini - IGM

Note su geometry_columns 1

- ▶ Definire i metadati minimi è fondamentale per il corretto funzionamento di PostGIS (e dei GIS connessi ad esso!)
- ▶ Una tabella può avere più colonne geometriche (corrisponde ad una feature class in cui ogni oggetto a più rappresentazioni: es. città puntuali o areali)
- ▶ Una colonna geometrica:
 - ▶ può contenere dati geometrici di tipo uniforme (scelta consigliata per non mettere in crisi i sistemi GIS), ad esempio punti, in questo caso *geometry_columns.type* contiene il valore POINT
 - ▶ Può contenere dati di tipo misto (punti, linee ed aree), in questo caso *geometry_columns.type* contiene il valore GEOMETRY

▶ 13

PostGIS - Claudio Rocchini - IGM

funzioni

- ▶ PostGIS definisce circa 700 funzioni SQL
- ▶ Vedremo alcune delle funzioni in dettagli durante l'esercitazione. Le funzioni sono divise nei gruppi:
 - ▶ Di Gestione: es. *AddGeometryColumn*
 - ▶ Di Costruzione: es. *ST_MakePoint*
 - ▶ Di accesso: es. *ST_Dimension*
 - ▶ Di modifica: es. *ST_Transform*
 - ▶ Di output: es. *ST_AsGML*
 - ▶ Di relazione e misura: es. *ST_Area* e *ST_Intersects*
 - ▶ Di processamento: es. *ST_Buffer*
 - ▶ Altro: es. *Linear Referencig*, *ST_Xmax*, etc.

▶ 15

PostGIS - Claudio Rocchini - IGM

Note su geometry_columns 2

- ▶ Il sistema di riferimento può essere:
 - ▶ Non indicato: SRID = -1
 - ▶ Indicato in *geometry_columns*, quindi per l'intera colonna (feature)
 - ▶ Indicato dentro il campo *geometry*, quindi per ogni elemento. In questo caso può variare da elemento ad elemento della stessa colonna (cosa sconsigliata)
- ▶ Alcuni sistemi esterni non riconoscono il sistema di riferimento correttamente:
 - ▶ *Shp2pgsql* non è in grado di decodificare il file *prj*
 - ▶ ArcGIS non è in grado di interpretare lo SRID Postgres, il sistema di riferimento deve essere comunicato dall'utente.

▶ 14

PostGIS - Claudio Rocchini - IGM

Lavorare con Postgres

In questa sezione iniziamo a fare qualche esercizio con SQL e i dati geografici

16

PostGIS - Claudio Rocchini - IGM

Valori letterali 1

- ▶ I valori geometrici possono essere specificati in modo letterale, seguendo lo standard OGC WKT:
- ▶ Punto 2D:
 - ▶ 'POINT(0 0)'
- ▶ Polilinea 3D:
 - ▶ 'LINESTRING(0 0 0,1 | 0,1 2 0)'
- ▶ Poligono (con buco):
 - ▶ 'POLYGON((0,0 4,0 4,0 0),(1,2 | 2,2 | 2,1 | 1))'
- ▶ Punto multiplo:
 - ▶ 'MULTIPOINT(0,0,1 2)'
- ▶ Punto con sistema di riferimento:
 - ▶ 'SRID=32632;POINT(0 0)'

▶ 17

PostGIS - Claudio Rocchini - IGM

Valori letterali 2

- ▶ I valori letterali di tipo `GEOMETRY` non possono essere utilizzati così come sono, altrimenti il db li scambia per stringhe, ma:
 - ▶ Si può utilizzare la funzione `ST_GeomFromEWKT` per convertirli in geometria
 - ▶ Si può accodare la dicitura `::GEOMETRY` per informare il sistema che il valore è una geometria.
- ▶ Es. provate ad eseguire la query:


```
SELECT ST_XMax (
  ST_GeomFromText ('LINESTRING (1 3 4, 5 6 7)')
);
```
- ▶ Nella query precedente abbiamo costruito una geometria e poi abbiamo utilizzato la funzione `ST_XMAX` per estrarre il valore x massimo.

▶ 18

PostGIS - Claudio Rocchini - IGM

Valori letterali 3

- ▶ Vogliamo proiettare le coordinate geografiche del centro di Firenze in UTM fuso32N, proviamo:

```
SELECT ST_Transform(
  'SRID=4326;POINT(11.25 43.75) '::GEOMETRY
  ,32632
)
```

- ▶ Abbiamo costruito la geometria di un punto in coord. Geografiche (con l'opzione `::GEOMETRY`) e l'abbiamo trasformato tramite la funzione `ST_Transform` a cui bisogna specificare il sistema di arrivo.
- ▶ La risposta di default è in binario e quindi faticiamo a leggerla. Per leggerla in testo applichiamo la funzione `ST_ASEWKT` a `ST_Transform`.

▶ 19

PostGIS - Claudio Rocchini - IGM

Creazione di una feature class: dati

- ▶ Creiamo adesso una feature a mano
- ▶ Questo vuol dire creare una tabella con una colonna di tipo `geometry` + i metadati associati
- ▶ Creiamo la feature edificio eseguendo:


```
CREATE TABLE edificio
(
  id INTEGER PRIMARY KEY,
  descr CHARACTER VARYING
);
```
- ▶ Nota: il vincolo chiave primaria è stato inserito direttamente dopo la definizione del tipo
- ▶ Nota2: per ora il campo geometrico non c'è. Potevamo inserirlo subito (aggiungendo la colonna "shape `GEOMETRY`") ma preferiamo farlo in un altro modo...

▶ 20

PostGIS - Claudio Rocchini - IGM

Creazione di una feature: geometria

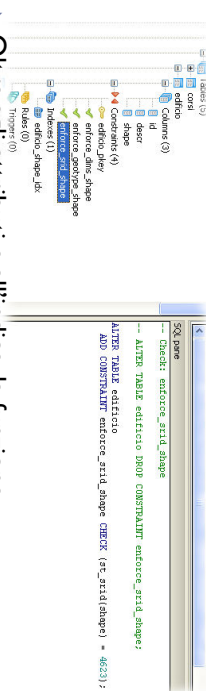
- ▶ Postgres contiene la funzione `AddGeometryColumn` che permette di aggiungere la colonna geometrica, i metadati e alcuni vincoli aggiuntivi in un sol colpo.
- ▶ La funzione ha i seguenti parametri:
 - ▶ Tabella su cui operare
 - ▶ Nome della colonna geometrica
 - ▶ Sistema di riferimento
 - ▶ Tipo geometrico
 - ▶ Numero dimensioni
- ▶ Provate ad eseguire (ormai dovrebbe essere chiaro):


```
SELECT
AddGeometryColumn ('edificio', 'shape', 4623, 'POLYGON', 2);
```

▶ 21

PostGIS - Claudio Rocchini - IGM

Uno sguardo alla nostra tabella

- ▶ Diamo un sguardo alla nostra tabella nell'albero grafico
- 
- ▶ Oltre agli attributi e all'indice, la funzione `AddGeometryColumn` ha aggiunto vincoli (`constraints`) aggiuntivi che bloccano il sistema di riferimento, il numero di dimensioni e il tipo geometrico

▶ 23

PostGIS - Claudio Rocchini - IGM

Creazione dell'indice spaziale

- ▶ Perché il funzionamento delle query spaziali sia veloce è fondamentale creare un indice sulla colonna `shape`
- ▶ La sintassi generica per creare indici è:


```
CREATE INDEX nome_indice ON nome_tabella (colonne)
```
- ▶ Gli indici spaziali però sono diversi da quelli su numeri e parole: in questo caso bisogna specificarne il tipo (che si chiama `gist`): provate quindi ad eseguire:


```
CREATE INDEX edificio_shape_idx
ON edificio USING gist (shape);
```
- ▶ Il nostro indice spaziale è in funzione

▶ 22

PostGIS - Claudio Rocchini - IGM

Uno sguardo ai metadati

- ▶ Selezionate nell'albero grafico la tabella (dello schema `public`) `geometry_column`
- ▶ Visualizzate la tabella e vedrete che questa contiene i metadati minimi della nostra feature:
 - ▶ Nome dello schema
 - ▶ Nome della tabella
 - ▶ Nome del campo geometrico
 - ▶ Dimensioni (2)
 - ▶ Sistema di riferimento (4623=WGS84)
 - ▶ Tipo geometrico

▶ 24

PostGIS - Claudio Rocchini - IGM

4623	public	edificio	shape	2	4623	POLYGON
WGS84	public	edificio	shape	4	WGS84	POLYGON

Popolamento della tabella

- ▶ Creiamo adesso qualche oggetto geografico, provate ad eseguire:


```
INSERT INTO edificio
VALUES
(1, 'Ospedale',
'SRID=4623;POLYGON((6 42, 8 42, 8 43, 6 43, 6 42))'::GEOMETRY
);
```
- ▶ I primi due valore sono la chiave primaria e la descrizione, il terzo è la geometria (un poligono rettangolare) del giusto sistema di riferimento
- ▶ Nota: le aree vanno "chiuso" vale a dire che l'ultimo vertice deve coincidere con il primo (ovvero i quadrati hanno 5 vertici...)

▶ 25

PostGIS - Claudio Rocchini - IGM

Semplici analisi sulle nostre feature

- ▶ Proviamo adesso a tabellare le aree dei nostri edifici con la query:


```
SELECT id,descr,
ST_Area(shape)
FROM edificio;
```
- ▶ L'area viene misurata in gradi quadrati... proviamo invece a proiettare queste coordinate nel fuso 32:


```
SELECT id,descr,
ST_Area( ST_Transform(shape, 32632) )/10000
FROM edificio;
```
- ▶ In questo caso il risultato è in ettari
- ▶ Nei semplici esempi fino a qui visti, le funzioni vengono applicate ad ogni singolo oggetto

▶ 27

PostGIS - Claudio Rocchini - IGM

Ancora altri dati

- ▶ Divertiamoci anche a creare un secondo oggetto:


```
INSERT INTO edificio
VALUES
(2, 'Industria',
'SRID=4623;POLYGON((10 43, 13 43, 13 46, 10 46, 10 43), (11 44, 12 44, 12 45, 11 45, 11 44))'::GEOMETRY);
```
- ▶ Nota: la riga della geometria qui sopra va scritta tutta di seguito (PowerPoint invece va a capo)
- ▶ La geometria della nostra industria è un rettangolo con un buco nel centro: la seconda serie di coordinate definisce il buco centrale.

▶ 26

PostGIS - Claudio Rocchini - IGM

Semplici funzioni aggreganti

- ▶ Vediamo ora qualche esempio di funzione aggregante. Provate ad eseguire:


```
SELECT ST_Extent(shape) FROM edificio;
```
- ▶ Il risultato è:


```
"BOX(6 42,13 46)"
```
- ▶ La funzione `ST_Extent` calcola l'estensione massima dell'unione degli elementi della tabella. Un altro esempio:


```
SELECT St_Asewkt(ST_Union(shape) )
FROM edificio;
```
- ▶ La funzione `ST_Union` calcola l'unione di tutte le geometrie

▶ 28

PostGIS - Claudio Rocchini - IGM

Connessione con QGIS (oppure UDIG)

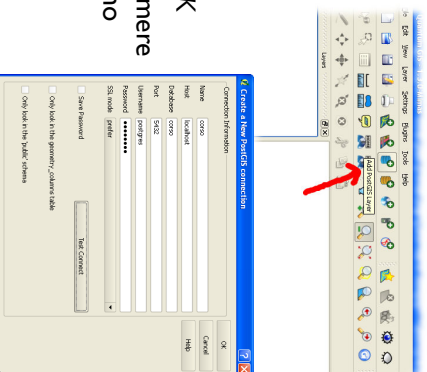
Ed ora un piacevole intermezzo: visualizzazione di dati tramite QGIS

29

PostGIS - Claudio Rocchini - IGM

Procedura di visualizzazione con QGIS 1

- ▶ Lanciare QGIS
- ▶ Premere il pulsante: **Add PostGIS Layer**
- ▶ Nel dialogo che si apre premere il tasto **new**
- ▶ Inserire i valori corretti di connessione e premere **OK**
- ▶ Nel dialogo principale premere il tasto **Connect**: appariranno tutte le feature disponibili



▶ 31

PostGIS - Claudio Rocchini - IGM

Visualizzare i dati: introduzione

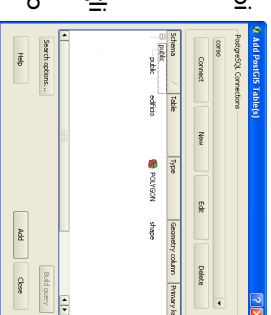
- ▶ Questa è un'esercitazione su Postgres-PostGIS
- ▶ Questo software non ha di per sé un visualizzatore grafico.
- ▶ E' difficile però continuare senza visualizzare i nostri dati.
- ▶ Introduciamo quindi i concetti minimi per poter visualizzare i dati con QGIS
- ▶ Una descrizione di QGIS esula dagli scopi di questa dispensa.

▶ 30

PostGIS - Claudio Rocchini - IGM

Procedura di visualizzazione con QGIS 2

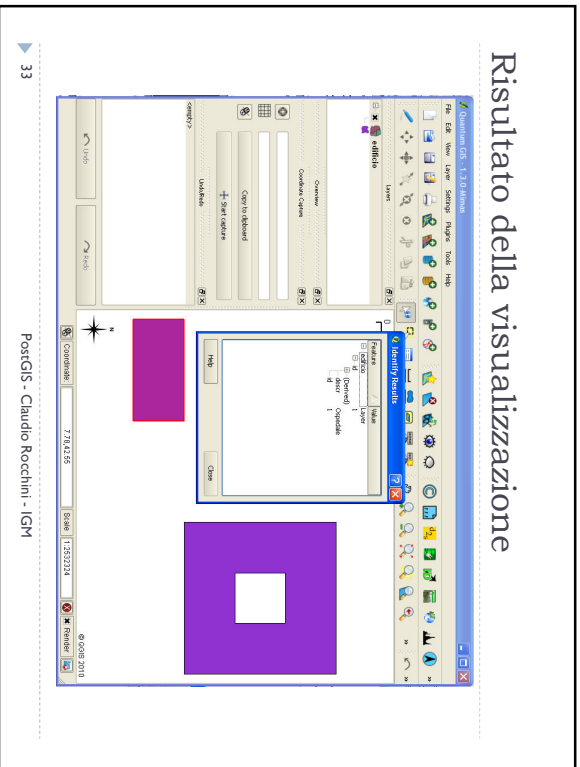
- ▶ Selezionare la feature public – edifici e premere il tasto **Add**
- ▶ Se tutto va bene apparirà nella mappa l'elenco degli oggetti da noi manualmente creati (vedi prossimo lucido)
- ▶ Selezionando lo strumento **Identify Feature (I)** è possibile anche interrogare gli attributi degli oggetti.
- ▶ Possiamo mantenere QGIS aperto mentre continuiamo a lavorare con Postgres



▶ 32

PostGIS - Claudio Rocchini - IGM

Risultato della visualizzazione



Importazione di dati Esterni (shape)

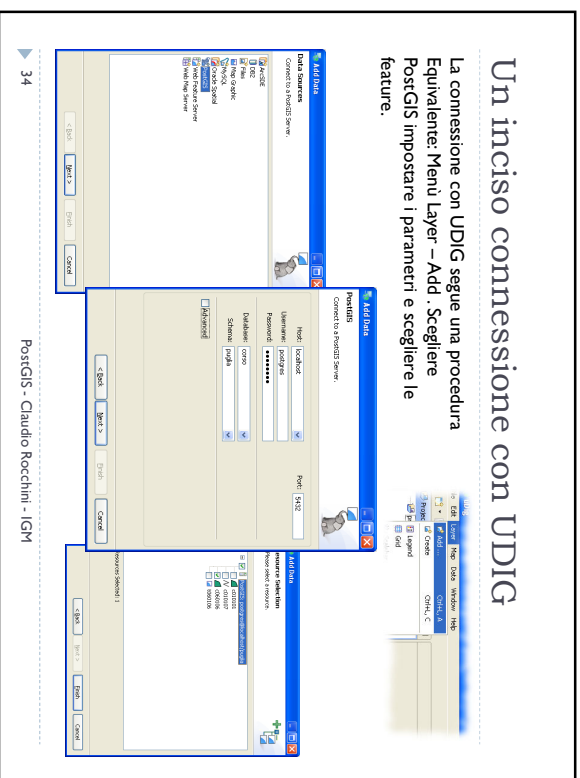
In questa sezione si importano dei dati esterni nel formato shape

35

PostGIS - Claudio Rocchini - IGM

Un inciso connessione con UDIG

La connessione con UDIG segue una procedura Equivalente: Menu Layer – Add . Scegliere PostGIS impostare i parametri e scegliere le feature.



Dati di Esempio

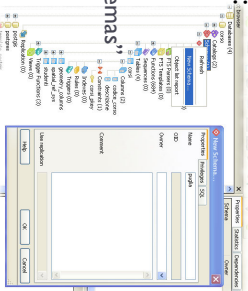
- ▶ Come dati di esempio, utilizzeremo il DB topografico della regione Puglia, in particolare i dati del foglio 383 e solo le feature C010101 (area stradale), C010107 (elemento stradale) e C060106 (Coltura Agricola).
- ▶ In particolare l'ultima feature contiene un attributo multivalore
- ▶ L'importazione dei file shape avverrà attraverso lo strumento "shp2pgsql", un tool di PostGIS. Lo stesso strumento può importare tabelle di dati puri (non geografici).

36

PostGIS - Claudio Rocchini - IGM

Preparazione all'importazione

- ▶ Importeremo i nostri file shape nel database Postgres di lavoro
- ▶ I dati però non verranno caricati nella schema principale, ma in un nuovo schema denominato "puglia".
- ▶ Creazione dello schema "puglia":
 - ▶ Aprite pgAdmin III
 - ▶ Selezionate il server da utilizzare
 - ▶ Selezionate il database "corso"
 - ▶ Clicca col bottone destro su "Schemas"
 - ▶ Selezionate "New Schema"
 - ▶ Digitate "puglia" come nome
 - ▶ Premete OK



▶ 37

PostGIS - Claudio Rocchini - IGM

Procedura di importazione

- ▶ Creare la cartella c:\lavoro
- ▶ Copiarci dentro i file shape e dbf da importare
- ▶ Aprire una shell DOS (ad esempio premendo Start – Esegui, digitando cmd e premendo OK)
- ▶ Digitare il comando:
 - ▶ cd \lavoro
- ▶ Digitare il comando:
 - ▶ shp2pgsql -s 32633 -c -g shape -I C010101_POL.shp puglia.C010101 > C010101.sql
- ▶ Viene creato il file c010101.sql



▶ 39

PostGIS - Claudio Rocchini - IGM

Funzionamento di shp2pgsql

- ▶ Shp2pgsql è un software dos (a riga di comando) funziona digitando il comando in una finestra dos, con le opportune opzioni
- USO: shp2pgsql [<options>] [<shapefile>] [<schema>] [<table>]
- OPTIONS:
 - s <srld> Seleziona lo SRID.
 - C Cancella e ricrea le tabelle
 - a Appende i dati
 - c Crea nuove tabelle e dati
 - p Crea le tabelle senza dati
 - g <geometry_column> specifica il nome del campo geometrico
 - I Crea l'indice spaziale
 - W <encoding> Specifica la codifica carattere
 - n Importa solo i dbf

▶ 38

PostGIS - Claudio Rocchini - IGM

Analisi dei parametri

- ▶ -s **32633** : seleziona il sistema di riferimento (32633 = UTM Fuso 32N, WGS84, il sistema utilizzato dalla Puglia), il sistema va indicato perché Postgres non riesce a leggere i file prj
- ▶ -c : crea tabella e carica i dati
- ▶ -g **shape** : seleziona "shape" come nome della colonna geometrica
- ▶ -I : crea anche l'indice spaziale
- ▶ **C010101_POL.shp** : il file da importare
- ▶ **puglia.C010101** : nome della tabella da creare (nello schema "puglia")

▶ 40

PostGIS - Claudio Rocchini - IGM

Importazione delle altre feature

- ▶ Si continua con il comando:
 - ▶ `shp2pgsql -s 32633 -c -g shape -I C010107_LIN.shp puglia.C010107 > C010107.sql`
- ▶ Quindi con il comando:
 - ▶ `shp2pgsql -s 32633 -c -g shape -I C060106_POL.shp puglia.C060106 > C060106.sql`
- ▶ Infine rimane da importare la tabella di soli dati T060106: questa non è una feature geografica ma una tabella pura, utilizzare per implementare un attributo multivalore. In questo caso va aggiunto l'opzione `-n` al posto dell'opzione `-I` (che non serve). Digitate il comando:
 - ▶ `shp2pgsql -s 32633 -c -g shape -n T060106_TY.dbf puglia.T060106 > T060106.sql`

▶ 41

PostGIS - Claudio Rocchini - IGM

Finestra DOS durante l'importazione

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versione 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\claudio>cd \lavoro

C:\lavoro>shp2pgsql -s 32633 -c -g shape -I C010107_POL.shp puglia.C010107 > C010107.sql
Shapefile type: PolygonZ
Postgis type: MULTIPOLYGON(41)

C:\lavoro>shp2pgsql -s 32633 -c -g shape -I C010107_LIN.shp puglia.C010107 > C010107.sql
Shapefile type: LineZ
Postgis type: MULTILINESTRING(41)

C:\lavoro>shp2pgsql -s 32633 -c -g shape -I C060106_POL.shp puglia.C060106 > C060106.sql
Shapefile type: PolygonZ
Postgis type: MULTIPOLYGON(41)

C:\lavoro>shp2pgsql -s 32633 -c -g shape -I -n T060106_TY.dbf puglia.T060106 > T060106.sql
C:\lavoro>

```

▶ 43

PostGIS - Claudio Rocchini - IGM

Uno sguardo ai file prodotti

- ▶ Lo strumento utilizzato produce dei comandi sql


```

SET STANDARD_CONFORMING_STRINGS TO ON;
BEGIN;
CREATE TABLE "puglia"."C010101" (gid serial PRIMARY KEY,
"file_id" int4,
"codice_in" varchar(50),
"fonte" varchar(8),
"rilevo" varchar(8),
"tipo_elab" varchar(8),
"data_note" varchar(254),
"livello" int2,
"scala" varchar(10),
"ac_vel_zon" varchar(8),
"ac_vel_fon" varchar(8),
"ac_vel_sed" varchar(8),
"ac_vel_liv" varchar(8),
"shape_leng" numeric,
"shape_area" numeric);
SELECT AddGeometryColumn('puglia','C010101','shape','32633','MULTIPOLYGON',4);
INSERT INTO "puglia"."C010101" ("file_id","codice_in","fonte","rilevo" ...

```

▶ 42

PostGIS - Claudio Rocchini - IGM

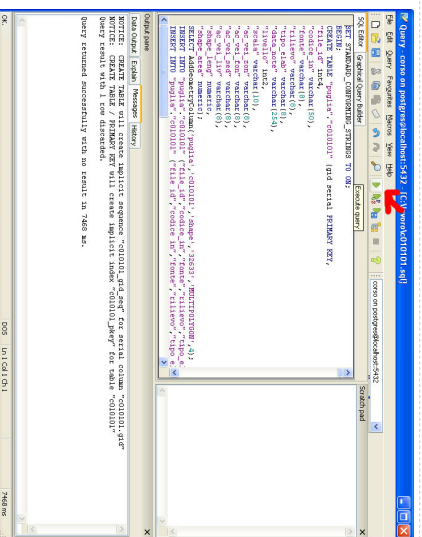
Esecuzione dei file SQL

- ▶ Adesso bisogna eseguire i file SQL prodotti.
- ▶ Un modo può essere quello di utilizzare il comando DOS "psql", ma noi invece utilizzeremo l'interfaccia grafica.
- ▶ Da pgAdmin III, lanciate la finestra SQL (Execute)
- ▶ Nella finestra SQL Editor, Selezionate il menù File – Open ... quindi selezionate il file "C010101.sql"
- ▶ Premete il pulsante play (Execute) e attendente l'esecuzione dei comandi SQL.
- ▶ Se tutto va bene il messaggio finisce con la frase: "Query returned successfully with no result in xxx ms."

▶ 44

PostGIS - Claudio Rocchini - IGM

Schermata di esecuzione dell'SQL



▶ 45

PostGIS - Claudio Rocchini - IGM

Esecuzione degli altri file

- ▶ Ripetete il ciclo di operazioni (Open – Execute) con gli altri file: C060106.sql, C010107.sql e T060106.sql.
- ▶ Attendente pazientemente il risultato di ogni esecuzione.
- ▶ Il caricamento dei dati è finito.
- ▶ Potete dare uno sguardo ai metadati: nell'interfaccia grafica selezionate la tabella `public.geometry_columns`, quindi premete il pulsante "View data" (Tabella):

oid	table_catalog	table_name	geometry_name	coord_dimension	srid	type
17249	public	publ	shape	4	32653	MULTIPOLYGON
17443	public	publ	shape	4	32653	MULTILINESTRING
17479	public	publ	shape	4	32653	MULTIPOLYGON

▶ 46

PostGIS - Claudio Rocchini - IGM