

PostgreSQL - amministrazione

PostgreSQL

Gabriele Bartolini

Flavio Casadei Della Chiesa

Luca Ferrari

Marco Tofanari

Associazione Italiana PostgreSQL Users Group

www.itpug.org

Pisa, 8 Maggio 2009



Licenza Creative Commons

Attribuzione

Non commerciale

Condividi allo stesso modo

2.5 Italia

<http://creativecommons.org/licenses/by-nc-sa/2.5/it/>



Scaletta

Installazione

Collegamento tramite openoffice, pgadmin III, ...

Creazione database e utenti

Configurazione per l'accesso al database

Cenni sui parametri di configurazione più utili (postgresql.conf)

Salvataggio ed importazione dati (pg_dump, psql)

Caricamento dati tramite COPY

Esempio di EXPLAIN, ANALYZE, VACUUM



Installazione

```
apt-get install postgresql
```

```
apt-get install postgresql-doc
```

```
apt-get install postgresql-contrib
```

Potrebbe essere necessario inizializzare la directory che conterrà i dati con initdb.

E' ovviamente possibile compilare dai sorgenti, per avere un ambiente ottimizzato.

Il demone PostgreSQL gestisce un cluster di database, ovvero un insieme di più database possono risiedere sullo stesso host e possono essere gestiti dallo stesso processo.



Template0, template1

Appena installato il cluster mette a disposizione un database vuoto, usato come template per la creazione di altri database: template1.

Tutte le caratteristiche di template1 possono essere modificate dall'utente, e saranno riflesse in ogni nuovo database creato.

Il database template0 rappresenta una copia di sicurezza di template1.



Connessione al database: psql

```
File Edit View Scrollback Bookmarks Settings Help
[luca@luca-laptop:~]$ psql -h localhost template1
Welcome to psql 8.3.3, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)

template1=#
```

Connessione al database: pgadmin3

The screenshot shows the pgAdmin 3 interface. On the left, the 'Esploratore degli oggetti' (Object Explorer) pane shows the database structure: Server (8) including FTP, Mammuth, Office Market, SEDE, SEDELDAP, backup, and development; and localhost (localhost:5432) including Database (3) with 'hrpmdb'. Under 'hrpmdb', 'Cataloghi (2)' and 'Schemi (1)' are shown, with 'public' schema containing 'Dominii (0)', 'Funzioni (0)', 'Sequenze (21)', and 'Tabelle (23)'. The 'address' table is selected under 'Tabelle'.

The right pane shows the 'Proprietà' (Properties) tab for the 'address' table. It displays the following properties and values:

Proprietà	Valore
Nome	address
OID	25003
Proprietario	hrpm
Tablespace	pg_default
ACL	{hrpm=arwdxt/hrpm,luca=arwdxt/hrpm}
Chiave primaria	addresspk
Righe (stimate)	547
Fattore riempimento	
Righe (contate)	547
Tabelle ereditate	No
Numero della tabella	0

Below the properties, the 'Riquadro SQL' (SQL Window) shows the SQL command to create the table:

```
-- Table: address
-- DROP TABLE address;

CREATE TABLE address
(
    addresspk serial NOT NULL,
    street character varying(50),
    city character varying(50),
    country character varying(50),
    state character varying(50),
    phone character varying(20),
    zipcode character varying(5),
    fax character varying(20),
    CONSTRAINT address_pkey PRIMARY KEY (addresspk)
)
```

The status bar at the bottom indicates 'Scaricamento dei dettagli di Tabella... Fatto.' and '0,14 sec'.

Connessione con openoffice base 1

Deve essere installato il layer di compatibilità con PostgreSQL (nativo, jdbc od odbc)

▼ Creazione guidata database

Passi

1. Seleziona database
2. Impostazioni di connessione
3. Configura autenticazione utente
4. Salva e procedi

Benvenuti alla Creazione guidata database di OpenOffice.org

Usate la Creazione guidata database per creare un nuovo database, aprire un database esistente o connettervi a un database memorizzato su un server.

Come volete procedere?

☐ Crea un nuovo database

☐ Apri un file di database esistente

Usato recentemente

Apri...

☒ Collega a un database esistente

postgresql

? <<Indietro Avanti >> Fine Annulla



Connessione con openoffice base 2

Creazione guidata database

Passi

1. Seleziona database
2. Impostazioni di connessione
3. Configura autenticazione utente
4. Salva e procedi

postgresql

dbname=itpug host=127.0.0.1

? <<Indietro Avanti >> Fine Annulla



Connessione con openoffice base 3

▼ Creazione guidata database

Passi

1. Seleziona database
2. Impostazioni di connessione
3. Configura autenticazione utente
4. Salva e procedi

Configura l'autenticazione utente

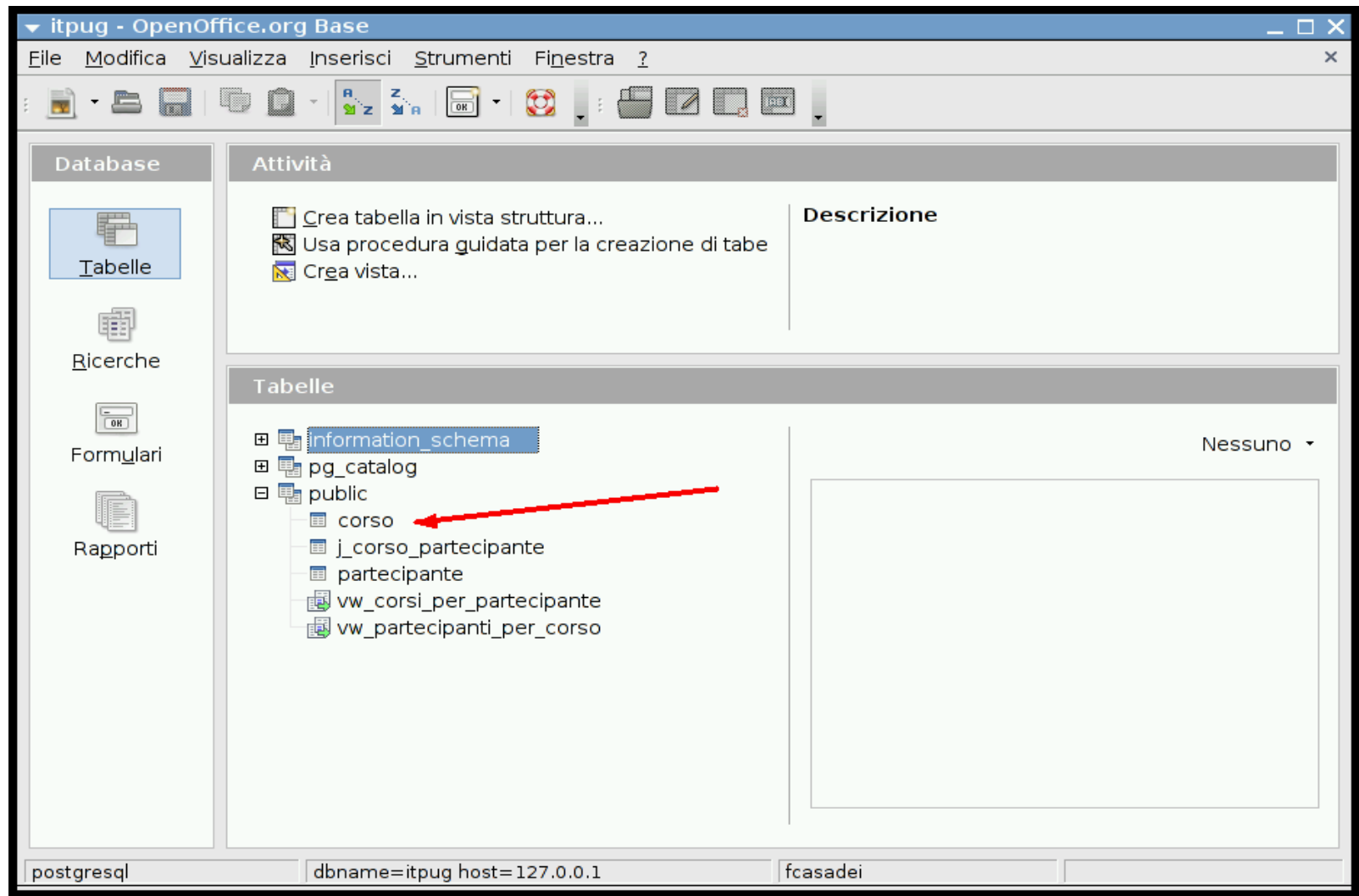
Alcuni database richiedono l'immissione di un nome utente.

Nome urente

☒ Password necessaria




Connessione con openoffice base 4



Connessione con openoffice base 5

▼ itpug: public.corso

File Modifica Visualizza Strumenti Finestra ?



	corsopk	corsoid	descrizione	requisiti	data	n_ore	materiale	ts	
▶	21	JAVA	Corso avanzato	Corso base	08/05/09	2		2009-	
+	<Campo aut								

Record di dati 1 da 1

Creazione di un nuovo utente

Una volta collegati al database template1 → è possibile creare un nuovo utente tramite CREATE USER

```
template1=# create user linuxday with createdb login password 'linux';  
CREATE ROLE  
template1=# \q  
[luca@fluca-laptop:~]$ psql -h localhost -U linuxday template1  
Password for user linuxday:
```

Da una shell → è possibile creare l'utente usando il comando create_user

```
[luca@fluca-laptop:~]$ createuser -P linuxday  
Enter password for new role:  
Enter it again:  
Shall the new role be a superuser? (y/n) y  
[luca@fluca-laptop:~]$
```



Da “psql” quali opzioni ha create user?

Per avere l'help di ogni comando (informazioni, sintassi, ...)

\h nomecomando

Es \h create user

Il comando \h mostrerà un elenco dei comandi supportati



Creazione di un database: psql

Una volta collegati al database template1 → è possibile creare un nuovo database con CREATE DATABASE

```
[luca@fluca-laptop:~]$ psql -h localhost -U linuxday template1
Password for user linuxday:
Welcome to psql 8.3.3, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)

template1=# create database linuxdaydb;
CREATE DATABASE
template1=# █
```



Creazione di un database: da shell

Da shell → usare lo script createdb

```
[luca@fluca-laptop:~]$ createdb -O linuxday -E UTF8 linuxdaydb
[luca@fluca-laptop:~]$ psql -h localhost -U linuxday linuxdaydb
Password for user linuxday:
Welcome to psql 8.3.3, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)

linuxdaydb=#
```



Creazione di una semplice tabella

Una volta collegati al database → eseguire il comando CREATE TABLE

```
linuxdaydb=# create table talks (  
linuxdaydb(# talkspk serial, -- chiave surrogata  
linuxdaydb(# talksid character varying(20), -- codice del talk  
linuxdaydb(# descrizione text, -- descrizione del talk  
linuxdaydb(# docente character varying(20), -- docente  
linuxdaydb(# PRIMARY KEY (talkspk),  
linuxdaydb(# UNIQUE (talksid)  
linuxdaydb(# );  
NOTICE: CREATE TABLE will create implicit sequence "talks_talkspk_seq" for  
serial column "talks.talkspk"  
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "talks_pkey"  
for table "talks"  
NOTICE: CREATE TABLE / UNIQUE will create implicit index "talks_talksid_key"  
" for table "talks"  
CREATE TABLE  
linuxdaydb=#
```



Popolamento del database

```

linuxdaydb=# insert into talks(talksid, descrizione, docente)
linuxdaydb-# VALUES('pg01', 'Presentazione PostgreSQL', 'Luca Ferrari');
INSERT 0 1
linuxdaydb=# insert into talks(talksid, descrizione, docente)
VALUES('pg02', 'PostgreSQL/JDBC', 'Luca Ferrari');
INSERT 0 1
linuxdaydb=# insert into talks(talksid, descrizione, docente)
VALUES('pg03', 'Partitioning', 'Mr. X');
INSERT 0 1
linuxdaydb=# select * from talks;
talkspk | talksid |      descrizione      | docente
-----+-----+-----+-----
      1 | pg01    | Presentazione PostgreSQL | Luca Ferrari
      2 | pg02    | PostgreSQL/JDBC         | Luca Ferrari
      3 | pg03    | Partitioning             | Mr. X
3 rows)

```



Esecuzione del contenuto di un file

- Non è necessario scrivere comandi sql direttamente in psql, infatti è possibile inserirli in un file di testo per poi farli processare
- Se si è connessi a PostgreSQL tramite il comando psql
 - \i **nomefile.sql**
- Da riga di comando
 - psql -d nomedatabase -f *nomefile.sql*



Esempio di esecuzione da file

```
tofanari@p5q:~/pisa$ cat file.sql
CREATE TABLE j_corso_partecipante (
    j_corso_partecipante_pk integer NOT NULL,
    corsopk integer NOT NULL,
    partecipantepk integer NOT NULL
);
tofanari@p5q:~/pisa$ psql -d test -f file.sql
CREATE TABLE
tofanari@p5q:~/pisa$
```



Introspezione

I comandi \d di psql consentono di effettuare introspezione sulla struttura del database. Ad esempio:

\d fornisce l'elenco degli oggetti nel database

\d nome_tabella fornisce la struttura di una tabella

\df fornisce l'elenco delle stored procedure

\df nome_funzione fornisce il codice sorgente della funzione

\dt elenco tabelle

\di elenco indici

\ds elenco sequences

\dv elenco viste

...

\? elenco comandi vari (tra cui l' introspezione)



Introspezione: esempio

```
linuxdaydb=# \d
               List of relations
 Schema |          Name          | Type   | Owner
-----+-----+-----+-----
 public | talks                  | table  | linuxday
 public | talks_talkspk_seq      | sequence | linuxday
(2 rows)

linuxdaydb=# \d talks
                                Table "public.talks"
   Column   |          Type          | Modifiers
-----+-----+-----
 talkspk    | integer                | not null default nextval('talks_talkspk_seq'::regclass)
 talksid    | character varying(20) |
 descrizione | text                   |
 docente    | character varying(20) |

Indexes:
    "talks_pkey" PRIMARY KEY, btree (talkspk)
    "talks_talksid_key" UNIQUE, btree (talksid)

linuxdaydb=#
```



Accesso: pg_hba.conf

Il file pg_hba.conf (host base access) contiene le regole per l'accesso al server PostgreSQL da parte dei client della rete. Occorre specificare il database, la maschera di rete dei client (o l'indirizzo ip) e il metodo di accesso (trust, md5, ssl, ...):

#	TYPE	DATABASE	USER	CIDR-ADDRESS	METHOD
	local	all	all		trust
	host	all	all	127.0.0.1/32	md5
	host	linuxdaydb	linux	192.168.1.0/24	md5
	host	all	all	:::1/128	md5



Postgresql.conf

Il file postgresql.conf contiene parametri fondamentali per la configurazione del server. Alcuni interessanti sono:

```
listen_address = 'localhost' {*, ip address}
```

specifica per quali interfacce il server deve accettare connessioni

```
log_statement = 'none' {none, all, ddl, mod}
```

consente di abilitare il logging dei comandi SQL eseguiti dal server, utile per il debugging o il monitoring dell'attività del server

```
shared_buffers = 24MB {almeno 16k a connessione}
```

indica la memoria disponibile per PostgreSQL in cui conservare le pagine dati

```
work_mem = 1MB
```

è la memoria usata per il sorting (clausole ORDER BY)



Vacuum

Il sistema esegue (manualmente o automaticamente) un comando di utilità, denominato vacuum, che controlla una tabella o un database alla ricerca delle tuple che non sono più visibili da nessuna transazione e le cancella. In questo modo i dati si ricompattano e il database riduce il suo spazio

**Original Heap
With Expired
Rows Identified**

A	A	E	A	A	A	E	A	A	A	A	A
C	C	X	C	C	C	X	C	C	C	C	C
T	T	P	T	T	T	P	T	T	T	T	T
I	I	I	I	I	I	I	I	I	I	I	I
V	V	R	V	V	V	R	V	V	V	V	V
E	E	E	E	E	E	E	E	E	E	E	E

**Move Trailing
Rows Into Expired
Slots**

A	A	A	A	A	A	A	A	A	A	A
C	C	C	C	C	C	C	C	C	C	C
T	T	T	T	T	T	T	T	T	T	T
I	I	I	I	I	I	I	I	I	I	I
V	V	V	V	V	V	V	V	V	V	V
E	E	E	E	E	E	E	E	E	E	E

Truncate File

A	A	A	A	A	A	A	A	A	A		
C	C	C	C	C	C	C	C	C	C		
T	T	T	T	T	T	T	T	T	T		
I	I	I	I	I	I	I	I	I	I		
V	V	V	V	V	V	V	V	V	V		
E	E	E	E	E	E	E	E	E	E		



Vacuum

L'esecuzione di vacuum è un'operazione pesante, perché il sistema deve acquisire un lock esclusivo sull'oggetto.

Vacuum può essere usato per eliminare le tuple espiate (vacuum full), oppure per aggiornare le statistiche di sistema (vacuum analyze).

Ogni volta che vacuum esegue, lo xmin di tutte le tuple visibili viene resettato (freeze) così da prevenire problemi ad un successivo wrap-around.

Può essere abilitato di default nel file postgresql.conf:

```
autovacuum = on
```



Vacuum: esecuzione

```
linuxdaydb=# vacuum full verbose talks;  
INFO: vacuuming "public.talks"  
INFO: "talks": found 0 removable, 8912896 nonremovable row versions in 171013 pages  
DETAIL: 0 dead row versions cannot be removed yet.  
Nonremovable row versions range from 56 to 68 bytes long.  
There were 12275442 unused item pointers.  
Total free space (including removable row versions) is 741655488 bytes.  
62508 pages are or will become empty, including 0 at the end of the table.  
112052 pages containing 740292984 free bytes are potential move destinations.  
CPU 1.95s/2.98u sec elapsed 38.70 sec.  
INFO: index "talks_pkey" now contains 8912896 row versions in 44786 pages  
DETAIL: 0 index row versions were removed.  
0 index pages have been deleted, 0 are currently reusable.  
CPU 0.29s/1.04u sec elapsed 6.36 sec.  
INFO: index "docente_idx" now contains 8912896 row versions in 85154 pages  
DETAIL: 0 index row versions were removed.  
48912 index pages have been deleted, 48912 are currently reusable.  
CPU 0.66s/1.34u sec elapsed 12.30 sec.  
INFO: index "durata_idx" now contains 8912896 row versions in 19561 pages  
DETAIL: 0 index row versions were removed.  
0 index pages have been deleted, 0 are currently reusable.  
CPU 0.10s/1.00u sec elapsed 3.33 sec.
```



Utilities

- `pg_dump`: effettua un dump di un database on-line, producendo un file SQL (eventualmente compresso) che contiene i comandi per la creazione del database, dello schema e dei dati
- `pg_restore`: ricostruisce un database, uno schema o i dati partendo da un file di dump
- `vacuumdb`: effettua il vacuum su un database
- `reindex`: ricostruisce gli indici
- `copy, \copy`: bulk loading di grosse moli di dati (il default usato da `pg_dump`)
- `initdb`: inizializza una directory da usare per il cluster (crea le cartelle `pg_clog`, `pg_xlog`, `base`, ...)



EXPLAIN

Explain è un comando SQL che consente di analizzare le scelte dell'ottimizzatore per l'esecuzione di una query.

Explain si basa sulle statistiche raccolte durante il funzionamento del database; le statistiche possono essere aggiornate anche eseguendo explain analyze.

Leggere l'output di explain è un'arte!



EXPLAIN: primo esempio

```
linuxdaydb=# explain select * from talks;  
               QUERY PLAN  
-----  
Seq Scan on talks (cost=0.00..14.40 rows=440 width=152)  
(1 row)  
  
linuxdaydb=#
```

Il piano di esecuzione indica:

Una scansione sequenziale sulla tabella (Seq Scan)

La dimensione delle tuple ritornate (152 byte)

Il numero di righe ritornate (440 !)

Il costo iniziale (0.00) e finale del piano (14.40)

C'è qualche cosa che non va!



EXPLAIN: aggiornare le statistiche

La tabella *talks* è stata appena creata, e le sue statistiche non sono aggiornate: l'ottimizzatore ipotizza che ci siano 440 righe al suo interno! Rifacendo il piano di esecuzione con le statistiche aggiornate si ha un valore migliore:

```
linuxdaydb=# explain analyze select * from talks;
                                QUERY PLAN
-----
Seq Scan on talks (cost=0.00..14.40 rows=440 width=152) (actual time=0.010
..0.015 rows=3 loops=1)
Total runtime: 0.068 ms
(2 rows)

linuxdaydb=#
```



EXPLAIN: con indici

```
linuxdaydb=# explain select * from talks where durata > 0 and durata < 0.5 order by talksid;  
               QUERY PLAN  
-----  
Sort  (cost=598388.77..602122.28 rows=1493405 width=37)  
  Sort Key: talksid  
    -> Bitmap Heap Scan on talks  (cost=73308.00..322724.78 rows=1493405 width=37)  
        Recheck Cond: (durata > 0)  
        Filter: ((durata)::numeric < 0.5)  
        -> Bitmap Index Scan on durata_idx  (cost=0.00..72934.65 rows=4480216 width=0)  
            Index Cond: (durata > 0)  
(7 rows)
```

