

# **PosgreSQL DBA Base**

**Federico Campoli**

## **PostgreSQL DBA Base**

Federico Campoli

Prima Edizione

Pubblicato 2007

Copyright © 2007 PGHost di Federico Campoli

Opera rilasciata sotto licenza Creative Commons - **Attribuzione - Non commerciale - Non opere derivate 2.5**

### **Tu sei libero:**

- di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera

### **Alle seguenti condizioni:**

- **Attribuzione.** Devi attribuire la paternità dell'opera nei modi indicati dall'autore o da chi ti ha dato l'opera in licenza.
- **Non commerciale.** Non puoi usare quest'opera per fini commerciali.
- **Non opere derivate.** Non puoi alterare o trasformare quest'opera, né usarla per crearne un'altra.

Ogni volta che usi o distribuisce quest'opera, devi farlo secondo i termini di questa licenza, che va comunicata con chiarezza. In ogni caso, puoi concordare col titolare dei diritti d'autore utilizzi di quest'opera non consentiti da questa licenza. Questa licenza non riduce o elimina in alcun modo i diritti morali dell'autore.

# Sommario

<b>1. Funzionalità PostgreSQL</b> .....	<b>1</b>
1.1. Caratteristiche del DBMS .....	1
1.1.1. Catalogo di sistema.....	1
1.1.2. Concorrenza MVCC.....	1
1.1.3. Transazionale.....	2
1.1.4. Funzionalita' di subquery .....	2
1.1.5. Supporto per piu' linguaggi procedurali.....	3
1.1.6. Trigger .....	3
1.1.7. Viste.....	3
1.1.8. Funzioni con librerie dinamiche scritte in C .....	3
1.1.9. Tablespace .....	3
1.2. Numeri limite .....	4
<b>2. Installazione di PostgreSQL</b> .....	<b>5</b>
2.1. ....	5
2.1.1. Installazione su Windows .....	5
2.1.2. Installazione su GNU/LINUX .....	5
<b>3. Amministrazione di PostgreSQL</b> .....	<b>8</b>
3.1. ....	8
3.1.1. Windows .....	8
3.1.2. GNU/LINUX .....	8
<b>4. L'accesso al database</b> .....	<b>9</b>
4.1. Il file pg_hba.conf .....	9
4.2. Le tipologie di connessione.....	9
4.3. Le modalita' di autenticazione .....	9
4.4. Il client testuale psql .....	10
4.4.1. Esempio di restore da dump testuale .....	10

# Lista delle Tabelle

1-1. Tabelle di sistema .....	1
1-2. Livelli di isolamento delle transazioni .....	2
4-1. Struttura file pg_hba.conf .....	9

# Capitolo 1. Funzionalità PostgreSQL

## 1.1. Caratteristiche del DBMS

PostgreSQL è un completo Database management system le cui caratteristiche sono paragonabili a prodotti commerciali di alto livello. In questo primo capitolo effettueremo una carrellata delle sue caratteristiche salienti.

### 1.1.1. Catalogo di sistema

PostgreSQL per operare sugli oggetti che compongono il database cluster adoperata una serie di tabelle e viste che compongono quello che viene chiamato catalogo di sistema. Interrogando tali tabelle e' possibile avere le informazioni piu' disparate riguardo allo stato del cluster.

Di seguito sono elencate le tabelle di sistema piu' rilevanti:

**Tabella 1-1. Tabelle di sistema**

Nome	Tipologia	Dati contenuti
pg_class	tabella	La tabella di sistema pg_class elenca le tabelle e tutto cio' che ha similitudini con le tabelle inclusi indici , sequenze viste, tipi dato composti e tabelle TOAST
pg_database	tabella	La tabella pg_database contiene le informazioni relative ai database del cluster.
pg_tablespace	tabella	La tabella di sistema pg_tablespace contiene informazioni relative alle tablespace presenti nel cluster.
pg_roles	vista	Vista di sistema con password oscurate della tabella pg_authid che contiene i dati degli user presenti nel cluster database.
pg_locks	tabella	Fornisce informazioni sui lock presenti nel cluster database.

## 1.1.2. Concorrenza MVCC

La consistenza dei dati viene garantita dal modello MVCC (Multiversion Concurrency Control) che garantisce *il lock più fine di quello di riga*. Questo significa che ogni query (e quindi ogni transazione) vede sempre una fotografia dei dati come erano al momento del lancio della query indipendentemente dallo stato dei dati su disco. L'approccio MVCC, evita l'uso di lock espliciti su gli oggetti della query e garantisce buone prestazioni in ambienti multiutente. I lock prodotti da MVCC non interferiscono nelle operazioni di lettura/scrittura che avvengono sempre in maniera isolata e mai conflittuale. E' comunque sempre possibile ricorrere ai lock espliciti da codice SQL.

## 1.1.3. Transazionale

Una transazione e' un blocco operativo di tipo DML<sup>1</sup> che viene eseguito dal database in maniera atomica e visibile solo alla sessione che l'ha aperta. Una transazione diventa visibile a tutto il database esclusivamente dopo che ne e' avvenuto il commit. PostgreSQL opera in modo tale da garantire l'integrita' dei dati secondo le regole dei livelli di isolamento delle transazioni.

Lo standard SQL definisce quattro livelli di isolamento delle transazioni in funzione di tre fenomeni che si possono verificare o meno tra transazioni concorrenti.

- **dirty read** Una transazione legge i dati modificati e non ancora committati da una transazione concorrente
- **nonrepeatable read** Una transazione ri legge dei dati precedentemente letti e li trova modificati poiche' un'altra transazione concorrente nel frattempo ha committato le modifiche.
- **phantom read** Una transazione riesegue una query che restituisce un recordset che soddisfa la condizione di ricerca scoprendo che il recordset che soddisfa la condizione di ricerca e' cambiato a causa di una transazione committata di recente.

I livelli di isolamento delle transazioni definiti dallo standard SQL sono quattro. Ogni livello permette o meno i fenomeni precedentemente indicati.

**Tabella 1-2. Livelli di isolamento delle transazioni**

Livello di Isolamento	Dirty Read	Nonrepeatable Read	Phantom Read
Read uncommitted	Possibile	Possibile	Possibile
Read committed	Non possibile	Possibile	Possibile
Repeatable read	Non possibile	Non possibile	Possibile
Serializable	Non possibile	Non possibile	Non possibile

PostgreSQL supporta i livelli di isolamento **Read committed** e **Serializable**.

I livelli di isolamento si possono impostare sia a livello di configurazione nel file **postgresql.conf** modificando il parametro **default\_transaction\_isolation** che runtime con il comando **SET default\_transaction\_isolation='serializable'**;

### 1.1.4. Funzionalità di subquery

PostgreSQL permette di definire, come operatore di raffronto nella clausola WHERE di una select, una ulteriore select. In realtà nel secondo operatore della where è possibile avere una qualsiasi funzione e quindi, la funzionalità di subquery è solo un sottoinsieme di tale potente caratteristica.

### 1.1.5. Supporto per più linguaggi procedurali

PostgreSQL permette di costruire delle Stored Procedures interne al database in vari linguaggi. Il capostipite è ovviamente il pl/pgsql le cui funzionalità sono estremamente raffinate ma, previa installazione nel database, è possibile programmare le proprie procedure nei seguenti linguaggi:

- pl/pgsql
- pl/java
- pl/python
- pl/perl
- pl/pgtcl
- pl/php
- pl/ruby
- pl/R

### 1.1.6. Trigger

PostgreSQL permette di automatizzare delle operazioni attraverso delle funzioni attivate da eventi di insert, update e delete sulle tabelle.

### 1.1.7. Viste

Una vista è uno shortcut di query in lettura che si comporta come una tabella. PostgreSQL non supporta le viste updatabili. Comunque è possibile implementarle lavorando con le regole.

### 1.1.8. Funzioni con librerie dinamiche scritte in C

È possibile creare una libreria scritta in C e linkarla dinamicamente nel database come stored procedure

## 1.1.9. Tablespace

Dalla versione 8.x esiste una rudimentale gestione dello spazio che permette di posizionare gli oggetti in aree specifiche del filesystem (tablespace).

Lo gestione dello spazio comunque non e' mai stata una prioritá' del team di sviluppo che ha preferito concentrarsi su aspetti piu' cruciali del DBMS

La motivazione di cio' viene sintetizzata in una frase circolata nelle mailing list di postgresql:

```
PostgreSQL "grew up" when 500 Meg hard drives were door
stops and controlling the use of disk space was often more bother than
just adding on more drives to an array.
Not that there's no need for quotes in postgresql, there is.
It's just not been a priority for anyone to jump into.
```

## 1.2. Numeri limite

- Dimensione massima per un database Illimitata
- Dimensione massima per una tabella 32 TB
- Dimensione massima per una riga 1.6 TB
- Dimensione massima per un campo 1 GB
- Numero massimo di righe per tabella Illimitate
- Numero massimo di colonne per tabella da 250 a 1600
- Numero massimo di tabelle Illimitate

## Note

1. Data Manipulation Language, comandi SQL che modificano i dati come INSERT e UPDATE.

# Capitolo 2. Installazione di PostgreSQL

## 2.1.

### 2.1.1. Installazione su Windows

L'installazione su Windows avviene normalmente in modalità binaria scaricando l'installer da [www.postgresql.org](http://www.postgresql.org). Una volta avviato il programma di installazione verrà richiesto se creare la data directory contestualmente all'installazione. In caso di risposta affermativa la directory verrà creata come sottodirectory della cartella di installazione.

#### Esempio 2-1.

```
Installazione PostgreSQL in directory c:\programmi\postgresql\8.2\  
  
La directory data viene creata, se non diversamente indicato in  
  
c:\programmi\postgresql\8.2\data
```

**Importante:** Attenzione al tipo di filesystem. L'installer si rifiuterà di creare la directory data contestualmente all'installazione se il filesystem non è NTFS

Successivamente verrà richiesto di indicare un utente di sistema che si occupi di avviare il servizio PostgreSQL.

**Importante:** Per ragioni di sicurezza un utente con privilegi troppo elevati o di dominio non è accettato per la gestione del servizio PostgreSQL. È quindi conveniente, prima di avviare l'installazione, creare un utente locale non locale che possa essere adoperato per l'amministrazione del servizio.

Selezione componenti e linguaggi. In questa fase è possibile selezionare gli eventuali linguaggi procedurali e componenti aggiuntivi, come tsearch2, da inserire di default nel cluster che verrà creato.

Installazione. Al termine dell'installazione il database dovrebbe essere pronto e funzionante.

## 2.1.2. Installazione su GNU/LINUX

### 2.1.2.1. Binaria

L'installazione di PostgreSQL su Linux puo' essere eseguita sia come pacchetto binario attraverso i tool specifici per distribuzione come apt-get yum rpm Sia come sorgente ricompilato. Per l'installazione binaria, nel caso di debian questa si risolve con un banale apt-get install postgresql

### 2.1.2.2. Sorgente

Perche' ricompilare?

Ricompilando si ottiene codice ottimizzato per il processore su cui dovrà girare il software.

Si può adoperare sempre l'ultima release senza essere legati al rilascio dei pacchetti binari

In fase di configurazione e' possibile specificare alcuni parametri che danno al database caratteristiche speciali come il supporto per il Python o la readline

Download dei sorgenti dal sito [www.postgresql.org](http://www.postgresql.org)

Creare un utente postgres e un gruppo postgres

Decomprimere i sorgenti in una directory di appoggio e cambiarne l'ownership a postgres:postgres

Diventare utente postgres e avviare la configurazione del source tree con ./configure

Qualora mancassero librerie necessarie alla compilazione installarle secondo le istruzioni della distribuzione che si sta utilizzando (rpm,dpkg, compilazione).

Compilare il database con make

Al termine della compilazione eseguire il regression test con make check

Installare il database con make install

Creare una directory con ownership postgres.postgres e inicializzarla con initdb -D data\_dir

Impostare le variabili di ambiente PATH e PGDATA per il corretto startup del processo postmaster

Avviare il database system con pg\_ctl start

**Esempio 2-2.**

```
postgres@gilead:~$ LOG: database system was shut down at 2007-04-08 22:22:23 CEST
LOG: checkpoint record is at 3/58B23D8
LOG: redo record is at 3/58B23D8; undo record is at 0/0; shutdown TRUE
LOG: next transaction ID: 25499308; next OID: 176387
LOG: next MultiXactId: 1; next MultiXactOffset: 0
LOG: database system is ready
LOG: transaction ID wrap limit is 1099035127, limited by database "postgres"
```

# Capitolo 3. Amministrazione di PostgreSQL

## 3.1.

### 3.1.1. Windows

L'avvio dell'istanza su Windows si amministra attraverso la scheda servizi

L'installer fornisce pgadmin3 come tool di gestione del cluster database

Pgadmin fornisce un tool rudimentale per editare il file di configurazione postgresql.conf

### 3.1.2. GNU/LINUX

L'istanza viene amministrata sia come processo che come servizio

L'avvio come processo viene avviato invocando direttamente l'eseguibile postmaster passandogli come parametro la posizione della directory data,

L'avvio come servizio viene gestito dal programma pg\_ctl che permette di avviare o di arrestare il database. Se la variabile di ambiente \$PGDATA e' valorizzata richiede, come parametro, solo il comando da eseguire.

pg\_ctl start avvia l'istanza

pg\_ctl stop ferma l'istanza

L'opzione -m indica la modalita' di shutdown che puo' avvenire in 3 modalita'

- **smart** attende che le connessioni siano terminate per chiudere l'istanza
- **fast** chiude le connessioni esegue il rollback delle transazioni non committate e poi chiude l'istanza
- **immediate** termina tutti i processi server e il postmaster lasciando il cluster dirty. Dopo uno shutdown immediate durante lo

# Capitolo 4. L'accesso al database

PostgreSQL autentica su due livelli Host L'accesso viene filtrato dal file pg\_hba.conf Database L'accesso viene gestito dall'RDBMS attraverso i ruoli utente

## 4.1. Il file pg\_hba.conf

Il file pg\_hba.conf si trova nella directory \$PGDATA creata con initdb E' un file di testo dove ogni riga mappa le regole di accesso al database secondo lo schema

**Tabella 4-1. Struttura file pg\_hba.conf**

TIPO	DATABASE	UTENTE	INDIRIZZO	METODO
local,host	nome_database,all	username,all	indirizzo ip o classe di indirizzi	trust password  crypt  md5

Valori ammessi Tipo (connessione): local,host Database: nome\_database,all User: username,all  
Indirizzo: indirizzo ip o classe di indirizzi (subnet mask opzionale) Metodo: trust,password,crypt,kerberos

## 4.2. Le tipologie di connessione

local

La connessione avviene su socket locale. Ammessa solo per connessioni localhost

host

La connessione avviene via tcp/ip. Nelle versioni precedenti alla 8 richiedeva che il postmaster fosse avviato con l'opzione -i Un record host puo' gestire sia connessioni SSL che non SSL per usare SSL e' necessario che il database sia compilato con SSL abilitato. Per funzionare correttamente richiede che il parametro listen\_addresses del file postgresql.conf sia correttamente valorizzato.

## 4.3. Le modalita' di autenticazione

trust La connessione avviene senza dover fornire password

Nella modalita' password viene trasmessa in chiaro

Nella modalita' crypt viene usato l'algoritmo crypt per criptare le password

Nella modalita' md5 viene fatto un hash md5 delle password

## 4.4. Il client testuale psql

E' il client predefinito di PostgreSQL E' un client a riga di comando con funzionalita' readline integrate  
Invocazione: psql -U [USERNAME] -p [PORT] [DBNAME]

Accetta comandi sql

Puo' leggere file esterni ed eseguire lo spool su disco Permette di analizzare gli oggetti presenti con i comandi \d Lanciato con l'opzione -E visualizza le query generate dai comandi interni

E' lo strumento principale per il restore dei database da dump sql

### 4.4.1. Esempio di restore da dump testuale

#### Esempio 4-1. Esempio modalita' di import

```
Welcome to psql 8.1.6, 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

templatel=# create database test_import;
LOG:  transaction ID wrap limit is 1073742586, limited by database "postgres"
CREATE DATABASE

templatel=# \c test_import
You are now connected to database "test_import".

test_import=# \i drupal.sql
SET
NOTICE:  CREATE TABLE will create implicit sequence "access_aid_seq" for serial column "access.aid"
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "access_pkey" for table "access"
CREATE TABLE
NOTICE:  CREATE TABLE will create implicit sequence "accesslog_aid_seq" for serial column "accesslog.aid"
```

```

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "accesslog_pkey" for table "accesslog"
.
.
.
.
SET
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
test_import=#
test_import=# \d
List of relations
Schema | Name | Type | Owner
-----+-----+-----+-----
public | access | table | fedec
public | access_aid_seq | sequence | fedec
public | accesslog | table | fedec
public | accesslog_aid_seq | sequence | fedec
public | aggregator_category | table | fedec
public | aggregator_category_cid_seq | sequence | fedec
public | aggregator_category_feed | table | fedec
public | aggregator_category_item | table | fedec
public | aggregator_feed | table | fedec
public | aggregator_feed_fid_seq | sequence | fedec
public | aggregator_item | table | fedec
public | aggregator_item_iid_seq | sequence | fedec
public | authmap | table | fedec
public | authmap_aid_seq | sequence | fedec
public | blocks | table | fedec
public | book | table | fedec
public | boxes | table | fedec
public | boxes_bid_seq | sequence | fedec
public | cache | table | fedec
public | client | table | fedec
public | client_cid_seq | sequence | fedec
public | client_system | table | fedec
public | comments | table | fedec

```

Come prima operazione creiamo un database **test\_import** nel cluster. Con il comando `\c test_import` ci connettiamo al database in oggetto. Infine con il comando `\i drupal.sql` viene caricato lo script che genera, in questo caso, gli schemi per il CMS drupal.