**PostgreSQL**

**version 15**

Distribuito tramite il sito www.manualioracle.it

Copyright © 2025 Assi Loris

Qualsiasi abuso sarà perseguito e punito secondo i termini di legge.

Version: 1.2

Alcuni termini usati sono trademarks registrati dei rispettivi proprietari.

**Sommario**

Introduzione al Manuale 4

Cap. 1 - I Database 5

1.1 – I database 5

1.2 – DBMS e RDBMS 6

1.3 – L’SQL e le sue estensioni 7

1.4 – Approccio DTAP nei database 8

Cap. 2 - Prime nozioni relative a Postgres 10

2.1 – Introduzione 10

2.2 – Postgres Versions 11

2.3 – Cluster Database 11

2.4 – Struttura logica di un Cluster Database 12

2.5 – Struttura Fisica di un Cluster Database 13

2.6 – Accesso al database 13

2.7 – Oggetti Locali e Globali 14

2.8 – Case sensitive in Postgres 15

Cap. 3 - Startup e Shutdown del Cluster Database 17

3.1 – Start e Stop del cluster db 17

3.2 – Determinare il servizio Postgresql tramite *systemctl command* 17

3.3 – Operazioni sul cluster db tramite *systemctl* 18

3.4 – Operazioni sul cluster db tramite pg\_ctl 20

3.5 – Operazioni sul cluster db tramite pg\_ctlcluster 21

Cap. 4 - User Interfaces 23

4.1 – Introduzione 23

4.2 – Il psql 23

4.3 – Comandi psql e Comandi SQL 24

4.4 – Casi particolari di Connessione al db 25

4.5 – Connessione ad un DB Dockerizzato 26

4.6 – Password file .pgpass 27

Cap. 5 - Gestire i Database 29

5.1 – Creare un Database 29

5.2 – Rimuovere o Rinominare un Database 30

5.3 – Template Database 30

5.4 – Clonare un Database 31

5.5 – Clonare un Database: impatto sui WAL file 33

Cap. 6 - Parametri del database 35

6.1 – Server configuration file 35

6.2 – PG\_SETTINGS 36

6.3 – Settaggio dei parametri del db 38

6.4 – SET SESSION 38

6.5 – ALTER SYSTEM 39

6.6 – Modifica del file postgresql.conf 42

Cap. 7 - File Postgresql 44

7.1 – Collocazione dei File 44

7.2 – Tablespace e Datafile 45

7.3 – Approfondimenti relativi alle Tablespace 47

7.4 – WAL e Checkpoint 48

7.5 – Logging di Postgres 49

Cap. 8 - Client Authentication 51

8.1 – Introduzione 51

8.2 – File pg\_hba.conf 51

8.3 – Parametro *password\_encryption* e Vista *pg\_hba\_file\_rules* 53

8.4 – User Name Maps 54

8.5 – *pg\_authid, password\_encryption e* *pg\_hba.conf* 55

Cap. 9 - Role, User e Schema 58

9.1 – Database Roles e Users 58

9.2 – Troubleshooting relativi ad utenti e connessioni 59

9.3 – Role Attributes 60

9.4 – Role Membership 61

9.5 – Predefined Roles 62

9.6 – Gli Schemi 63

9.7 – Create, Set, Drop Schema 64

9.8 – Schema Public e Schema Search Path 65

Cap. 10 - Internals 68

10.1 – Processi Postgres 68

10.2 – L’Information Schema 69

10.3 – System Catalogs 70

10.4 – TOAST e Storage Fisico 72

Cap. 11 - Miscellaneous 74

11.1 – Introduzione al Vacuum 74

11.2 – Autocommit 74

11.3 – Overview del Backup & Restore 75

11.4 – Migrazione a Postgres 15 e Nuove Feature riguardanti le Performance 76

11.5 – Altre Nuove Feature in Postgresql 15 77

## Introduzione al Manuale

 **Contenuto**

Argomenti trattati in questo manuale:

 - introduzione ai database relazionali

 - dettagli sul database Postgresql (detto anche Postgres).

Ci riferiamo a Postgresql versione 15 su Linux (soprattutto Debian e Red Hat) ma quasi tutti gli argomenti qui trattati sono validi anche sulle altre piattaforme.

 **Audience**

Il presente manuale è rivolto ai tecnici informatici, agli studenti e a chiunque voglia capire che cosa sia un database e voglia conoscere le principali caratteristiche di Postgres.

 **Particolarità**

Il presente manuale è in italiano ma molti termini tecnici in esso contenuti sono in lingua inglese.

Abbiamo fatto tale scelta perché spesso a lavoro tali termini vengono detti in inglese e perché riteniamo più utile che i lettori del manuale conoscano tali termini nella lingua usata nella documentazione ufficiale.

Molti argomenti sono schematizzati per facilitarne la comprensione e la memorizzazione.

Alcune abbreviazioni o sinonimi usati:

 db : database

 leader database = master database = primary database

 replica database = slave database = secondary database

 restorare : effettuare un restore

 O.S. : *Operating System* (Sistema Operativo)

 *<data\_directory>* : valore del parametro Postgres chiamato *data\_directory*

 **Principali Versioni**

- 30.04.2025 : version 1.0

- 09.05.2025 : version 1.2 varie modifiche e rilascio al pubblico

**Disclaimer**

Non si fornisce alcuna garanzia relativamente al fatto che il presente documento sia privo di errori. Non si assume nessuna responsabilità sugli eventuali errori o danni derivanti dall’uso delle informazioni qui contenute.

## I Database

### 1.1 – I database

Il database è un archivio di dati usato per contenere e restituire informazioni correlate fra loro.

Tali dati sono gestiti sui server attraverso appositi software detti DBMS e sono organizzati tramite tecniche di “modellazione” che hanno i seguenti obiettivi:

- organizzare efficientemente i dati

- ridurre la ridondanza dei dati

- garantire l’integrità dei dati

Ciò porta a queste conseguenze:

- il database immagazzinerà molti dati pur non avendo a disposizione un grande storage

- il database restituirà facilmente e velocemente le informazioni pur non avendo a disposizione un server molto potente

I database hanno 2 tipi di struttura:

* + - Logica:

E’ rappresentata dai componenti che si possono vedere dentro il database (tabelle, indici, ecc..)

* + - Fisica:

Sono i file fisici che un database usa per immagazzinare i dati

Un aspetto importante relativo ai database è quello della **sicurezza:** i dati contenuti in un database possono essere molto importanti per una azienda per cui è importante proteggerli da:

- perdite di informazioni (accidentali o no). Per questo si usano le tecniche di Backup e Recovery

- lettura o modifica dei dati stessi da parte di persone non autorizzate. Per questo sul database si usano i GRANT ed altre tecniche atte ad evitare il cosiddetto *SQL injection*.

L’architettura più semplice con cui vengono usati i database è composta da 2 parti (chiamate anche *tier*):

- *Database Server:*

E’ la macchina su cui è installato il database.

- *Database Client:*

 E’ la macchina in cui si collega l'utente che vuole interfacciarsi con il database.

Spesso però c’è un “livello” intermedio fra questi per cui si parla di “architettura a 3 livelli” (“*3-tier architecture*”).

In questi casi, i livelli in questione sono:

....................

....................

....................

## Prime nozioni relative a Postgresql

### 2.1 – Introduzione

*PostgreSQL* (detto anche *Postgres*) è un *Object-Relational DBMS open source* nato nel 1996e supportato da diverse aziende*.*

Come detto all'inizio del manuale, esiste anche una versione a pagamento di Postgres chiamata *EDB Postgres Advanced Server (EPAS)*, tale versione è gestita dall'azienda *EnterpriseDB.*

In questo manuale, descriveremo Postgres solo nella sua versione *open source*, in quanto è quella più diffusa.

Qui l’elenco delle aziende che danno supporto su Postgresql in Europa:

<https://www.postgresql.org/support/professional_support/europe/>

Postgres può girare nella maggior parte dei sistemi operativi Linux, Unix, Windows e supporta oggetti come:

* *foreign keys, joins, views, triggers*
* *stored procedures* scritte in una dozzina di linguaggi, incluso *Java, Perl, Python, Ruby, C/C++* e *PL/pgSQL* (quest’ultimo è il linguaggio nativo di *Postgres,* simile al *PL/SQL* di *Oracle*)
* *data types* come INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, TIMESTAMP e *binary large objects* (come pictures, sounds o video).

Un particolare tipo di dato esistente in Postgres è il *pg\_lsn :* è usato per contenere un LSN (*Log Sequence Number*) che è puntatore alla location del WAL.

Inoltre Postgres usa l' *ANSI-SQL standard* e supporta:

* *subquery*
* *auto-increment columns* attraverso le sequence
* LIMIT/OFFSET che permettono di ritornare *result set* parziali

Postgresql supporta anche features avanzate come:

*- Multi-Version Concurrency Control (MVCC)*

*- table partitioning*

*-* ricostruzione “a caldo” degli indici: REINDEX CONCURRENTLY

*- point in time recovery*

*- tablespaces*

*- asynchronous replication*

*- nested transactions (savepoints)*

*- online/hot backups*

*- query planner/optimizer*

....................

....................

....................

## Startup e Shutdown del Cluster Database

### 3.1 – Start e Stop del cluster db

La corretta gestione del ciclo di vita del *cluster database*, ovvero l'avvio e l'arresto controllato, è un'operazione importante per l'amministrazione di Postgresql.

Un'esecuzione appropriata di queste procedure garantisce la disponibilità del servizio.

La presenza di diversi metodi per gestire lo startup e lo shutdown riflette l'evoluzione dei sistemi operativi Linux e offre flessibilità agli amministratori in base al contesto e alle preferenze.

In Linux esistono diversi metodi per fare lo *start* e lo *stop* di un *cluster database Postgres*:

* Tramite il comando *service*

Metodo storico usato nei sistemi *System V* *init* dunque eviteremo di approfondirlo

* Tramite il comando *systemctl*

E’ il metodo più raccomandato e ampiamente utilizzato sui sistemi Linux moderni che adottano *systemd* (la maggior parte delle distribuzioni attuali, inclusi *Red Hat/CentOS/Fedora, Ubuntu, Debian,* ecc.).

Lo usi tramite l’utente *root*

* Tramite il comando *pg\_ctl*

Lo usi tramite l’O.S. user *postgres*

* Tramite il comando *pg\_ctlcluster*

Funziona solo nei sistemi Debian-like quindi non è molto usato. Lo usi tramite l’utente *root* o *postgres*

Nei prossimi paragrafi approfondiremo dunque 3 di questi metodi.

### 3.2 – Determinare il servizio Postgresql tramite *systemctl command*

Qui vediamo come determinare il nome esatto del Servizio Postgres sulla macchina Linux in cui sei connesso.

I seguenti comandi vanno eseguiti tutti come *root.*

....................

....................

....................

## User Interfaces

### 4.1 – Introduzione

La scelta dell'interfaccia utente più adatta dipende spesso dal ruolo dell'utente (amministratore di database, sviluppatore, analista, ecc), dalle esigenze specifiche e dalla familiarità con lo strumento.

Alcune delle più comuni intefacce utenti usate per lavorare su Postgressono:

* **psql** :

E’ il tool più usato dai DBA. E’ particolarmente adatto se devi lanciare degli script o se sei già connesso al server Postgres(ad esempio tramite PuTTY).

Viene installato assieme all’installazione del *postgres server e* del *postgres client* e non è molto user-friendly.

E’ il tool a cui faremo più riferimento in questo manuale.

* **pgAdmin** :

E’ molto usato da chi sviluppa codice sul database. E’ molto user-friendly e tipicamente è installato sulla macchina (client) in cui stai lavorando.

*pgAdmin* fornisce un ambiente grafico completo per la gestione di database, schemi e oggetti, oltre a un editor SQL con funzionalità di autocompletamento ed evidenziazione della sintassi.

* **DBeaver** :

Anche questo è molto usato da chi sviluppa codice sul database. E’ molto user-friendly e tipicamente è installato sulla macchina (client) in cui stai lavorando.

Può essere usato per accedere a vari RDBMS (Postgres, Oracle e altri).

La sua capacità di connettersi a diverse piattaforme di database lo rende uno strumento versatile per team che lavorano con ambienti eterogenei.

* **IDE (IntelliJ, Eclipse)** :

....................

....................

....................

## Gestire i Database

In questo paragrafo facciamo riferimento ai “singoli database” e non all’intero *cluster database*.

Ricordiamo la differenza fra i singoli database ed il *cluster database* tramite questa figura relativa ad una singola installazione Postgres su una singola macchina:



### 5.1 – Creare un Database

Qui, ripetiamo, parliamo di creazione di un singolo *database* e non di un *database cluster*.

Tutti i comandi di sistemi operativi descritti qui vanno bene in tutte le distribuzioni Linux.

Per vedere velocemente quali database hai nel tuo *cluster database,* dentro *psql* lancia:

 \l

Oppure (da qualsiasi interfaccia, *psql* o altro):

 SELECT \*

 FROM pg\_database;

Per creare un nuovo database devi essere connesso al *cluster database* tramite un utente che abbia il privilegio CREATEDB (ad esempio, tramite il database user *postgres*).

Ecco come creare un ruolo (dopo vedremo la differenza fra ruoli ed utenti) chiamato ADMIN con il privilegio CREATEDB:

 CREATE ROLE admin

 WITH CREATEDB;

Se vuoi creare un database chiamato DB\_LOR assegnandogli come proprietario il *db user* con cui sei connesso, lancia il seguente *SQL command:*

....................

....................

....................