**PostgreSQL**

**version 14**

**-**

**SQL, PL/pgSQL,**

**Performance & Tuning**

**del Database**

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 - Introduzione e Interfacce Utenti 5

1.1 – Cluster Database 5

1.2 – Struttura logica di un Cluster Database 5

1.3 – Utenti 6

1.4 – Interfacce Utenti 7

1.5 – Il psql 8

1.6 – Comandi psql e Comandi SQL 8

Cap. 2 - Introduzione al Linguaggio SQL 10

2.1 – L’SQL e le sue estensioni 10

2.2 – SQL Statements in Postgres 11

2.3 – Creare ed Eliminare una Tabella 12

2.4 – Query e Subquery 13

2.5 – Esempi di Query 14

2.6 – DML sulle Tabelle 15

2.7 – Oggetti Locali e Globali 15

Cap. 3 - Esempi di Progetti e Database 17

3.1 – Creare e Rimuovere un Progetto 17

3.2 – Le tabelle EMP e DEPT 18

3.3 – Popolare EMP e DEPT 19

3.4 – Interrogare EMP e DEPT 20

Cap. 4 - Strumenti e Meccanismi per l’Integrità e l’Efficienza 22

4.1 – I constraint 22

4.2 – Le Viste 23

4.3 – Gli Indici 24

4.4 – Index Types 24

4.5 – Le Transazioni 25

4.6 – I Lock 27

4.7 – I Row-level Lock 28

4.8 – I Deadlock 29

Cap. 5 - JSON Type in Postgres 31

5.1 – Introduzione ai JSON type 31

5.2 – Esempi 31

5.3 – Vantaggi e svantaggi nell’uso dei JSON 33

5.4 – JSONB: containment 34

5.5 – JSONB: existence 35

5.6 – JSON Functions e Operators 36

Cap. 6 - Partitioning 38

6.1 – Overview sul Partizionamento 38

6.2 – Esempio di creazione e gestione di una Tabella Partizionata 39

6.3 – Indici e DDL su Tabelle partizionate 41

6.4 – pg\_partman 42

Cap. 7 - Introduzione al PL/pgSQL 44

7.1 – PL/pgSQL e Blocco Anonimo 44

7.2 – I Postgres Type System 45

7.3 – Le Funzioni 46

7.4 – Le Procedure 47

7.5 – Trigger e Trigger Function 48

7.6 – Tipi di Trigger 49

7.7 – Errori e Messaggi 50

Cap. 8 - Performance & Tuning: Introduzione 52

8.1 – Processi Postgres 52

8.2 – Aree di memoria Postgres 53

8.3 – Maintenance Tasks e vacuuming 54

Cap. 9 - Ottimizzazione delle Query 56

9.1 – Flusso di Esecuzione di una Query 56

9.2 – Planner/Optimizer e effective\_cache\_size 56

9.3 – Introduzione all’EXPLAIN 57

9.4 – Esempi di uso dell’EXPLAIN 58

9.5 – Opzioni dell’EXPLAIN 60

9.6 – Partition Pruning ed EXPLAIN di una query di una Tabella Partizionata 61

9.7 – Sort, work\_mem ed EXPLAIN su singola tabella 63

9.8 – Sort, work\_mem ed EXPLAIN su Join 66

Cap. 10 - PG\_STAT\_STATEMENTS 68

10.1 – Introduzione 68

10.2 – Installazione del PG\_STAT\_STATEMENTS 68

10.3 – Interrogare la PG\_STAT\_STATEMENTS 70

10.4 – QUERYID e univocità di un SQL statement 71

10.5 – Configuration Parameters 73

10.6 – Usare al meglio la PG\_STAT\_STATEMENTS 75

Cap. 11 - Performance & Tuning sistemistico 77

11.1 – Parametri da Modificare 77

11.2 – Statistics Collector 79

11.3 – Limitare le connessioni al Cluster Database ed ai singoli database 80

11.4 – Parametro log\_min\_duration\_statement 81

11.5 – Usare al meglio il log\_min\_duration\_statement 83

Cap. 12 - Viste e Metriche 85

12.1 – Database Cache hit ratio 85

12.2 – PG\_STAT\_DATABASE 86

12.3 – Commit ratio 86

12.4 – WAL configuration e Checkpoint 87

12.5 – PG\_STAT\_WAL 88

Cap. 13 - Database Tools ed Utilities 90

13.1 – Introduzione 90

13.2 – Reindexing 90

13.3 – pgbench 91

Cap. 14 - Vacuum e ANALYZE 92

14.1 – Vacuum 92

14.2 – Recuperare lo spazio disco tramite Vacuum 92

14.3 – Esempi di uso del Vacuum 93

14.5 – Vacuum e ANALYZE 95

14.6 – ANALYZE e Statistiche su Tabelle ed Indici 96

14.7 – System Catalog e Parametri associati al Vacuum 97

Cap. 15 - PgStatTuple 100

15.1 – Installazione di PgStatTuple 100

15.2 – Come usare del PgStatTuple 101

15.3 – Query da usare 103

15.4 – Interpretare l’output del PgStatTuple 105

Cap. 16 - Caratteristiche del Database Server 108

16.1 – Macchina Fisica o Virtuale 108

16.2 – Bare Metal server 109

16.3 – I File System 110

16.4 – Macchine Virtuali 110

16.5 – Tipo di Processore 111

16.6 – Standard Unix Tool per monitorare il Database 112

Cap. 17 - Performance & Tuning: Miscellaneous 114

17.1 – Viste relative alle Statistiche 114

17.2 – Temporary files, work\_mem, log\_temp\_files 115

## Introduzione al Manuale

**Contenuto**

Argomenti trattati in questo manuale:

- l’ SQL in Postgres

- introduzione al PL/pgSQL

- Performance & Tuning dell’SQL e del database

Ci riferiamo a Postgres versione 14 su Linux (soprattutto Debian e Red Hat) ma la maggioranza degli argomenti e comandi qui scritti sono validi anche sugli altri sistemi operativi.

**Audience**

Il presente manuale è rivolto ai tecnici informatici, agli studenti e a chiunque voglia conoscere gli argomenti elencati sopra.

**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 al fine di fornire una comprensione ed una memorizzazione superiore.

Alcune abbreviazioni usate:

db : database

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

VM : Virtual Machine (macchina virtuale)

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

**Principali Versioni**

- 16.01.2025 : version 1.1

- 21.01.2025 : version 1.2 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.

## Introduzione e Interfacce Utenti

### 1.1 – Cluster Database

In Postgres, il termine "*cluster database*" si riferisce a un'installazione del database server che gestisce una collezione di database distinti.

Questo è diverso da altri RDBMS (come Oracle), dove spesso si usa il termine "database" per riferirsi a un unico insieme di dati gestito dal server.

Relazione tra *cluster database* e *database*:

* Ogni *cluster database* contiene uno o più database.
* I database in un *cluster database* condividono utenti e configurazioni di base, ma sono completamente isolati tra loro a livello di dati.

Comunque spesso “per semplicità” (o per ignoranza) il *cluster database postgres* viene chiamato semplicemente *database postgres*.

Ecco una sua rappresentazione grafica:

A diagram of a database cluster

Description automatically generated

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

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

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

## Introduzione al Linguaggio SQL

I *SQL statement* (spesso pronunciato *siquel statement*) sono le istruzioni SQL da lanciare sul database per fare delle interrogazioni, creare oggetti o utenti, droppare tabelle o indici, ecc... ecc..

Qui vedremo i *SQL statement* più semplici ed usati.

### 2.1 – L’SQL e le sue estensioni

**SQL (Structured Query Language)**

Ossia *Linguaggio di Interrogazione Strutturato*.

E’ un linguaggio di interrogazione per database relazionali usato per leggere, modificare e gestire dati memorizzati in un RDBMS.

I comandi SQL si possono suddividere nelle seguenti categorie:

* *DQL (Data Query Language) o Query:*

Permettono, tramite SELECT, l’interrogazione dei dati presenti nel database.

L'acronimo DQL è poco usato. Al suo posto si usa di più il semplice termine "query".

* *DML (Data Manipulation Language):*

Permettono di inserire, cancellare, modificare e leggere i dati.

Es.:

INSERT, UPDATE, DELETE di righe dentro una tabella.

Le SELECT possono essere considerate DML solo se utilizzate per inserire in una tabella dati presi da un'altra tabella:

INSERT INTO ...

SELECT ...

FROM ...;

* *DDL (Data Definition Language):*

Permettono di creare, modificare ed eliminare oggetti del db.

Es.:

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

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

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

## Esempi di Progetti e Database

In questo capitolo vedremo i seguenti esempi:

* Gli step che si possono fare quando crei un nuovo progetto in ambito postgresql (vedremo un progetto semplice ed uno più complesso)
* Un piccolo database di esempio in cui creiamo 2 tabelle collegate fra loro tramite una *foreign key*

### 3.1 – Creare e Rimuovere un Progetto

Quando crei un nuovo progetto potresti seguire tanti passaggi, qui vedremo un paio di esempi.

**Progetto semplice**

Ecco un esempio degli step minimi da seguire per la creazione di un nuovo progetto non molto complesso:

* 1. Creare gli utenti sul *database cluster*
  2. Creare il singolo database
  3. Dentro il singolo database, connesso con l’utente corretto, creare tabelle ed eventualmente popolarle
  4. Dentro il singolo database, connesso con l’utente corretto, creare gli indici e constraints
  5. Dentro il singolo database, connesso con l’utente corretto, creare ulteriori oggetti come sequence, viste, funzioni ecc
  6. Dentro il singolo database, connesso con l’utente corretto, dare eventuali grant su singole tabelle o schema

Per rimuovere totalmente un ambiente creato come sopra, puoi seguire i seguenti step:

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

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

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

## Strumenti e Meccanismi per l’Integrità e l’Efficienza

Qui facciamo un introduzione relativa all’importanza di constraint, indici e transazioni per garantire integrità e performance.

### 4.1 – I constraint

Un constraint è un oggetto che crei su una tabella e serve a restringere i valori da poter inserire nella tabella stessa.

Ad es. puoi imporre che nella tabella delle città si debba per forza valorizzare il campo che indica il numero dei suoi abitanti.

In Postgres esistono i seguenti tipi di constraints:

* NOT NULL :

Impedisce l’inserimento di valori NULL all'interno di un campo di una tabella

* UNIQUE :

Impedisce di avere 2 valori uguali nella stessa colonna (o combinazione di colonne) ma non impedisce l’inserimento di più valori NULL

* PRIMARY KEY :

E’ la combinazione dei constraint NOT NULL e UNIQUE. Secondo alcune teorie relative ai database relazionali, ogni tabella dovrebbe avere una *primary key*

* FOREIGN KEY :

Richiede che i valori di uno o più campi di una tabella esistano nelle corrispettive colonne di altre tabelle (chiamate anche “tabelle padri” o “tabelle madri”, in inglese si dice “*parent tables*”).

Ad esempio, supponiamo di avere la tabella delle città e fra le sue colonne c’è quella che indica lo Stato in cui si trovano tali città.

Dunque potresti fare una *Foreign Key* dalla tabella delle città verso l’anagrafica delle Stati.

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

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

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

## JSON Type in Postgres

### 5.1 – Introduzione ai JSON type

I *JSON data type* sono usati per contenere dati di tipo JSON (*JavaScript Object Notation*).

Questi dati possono anche essere conservati come TEXT, ma il *data type JSON* ha il vantaggio di forzare il controllo sulla validità del dato secondo le regole JSON.

In Postgres, ci sono 2 tipi di *JSON data type* e questi tipi accettano quasi lo stesso set di valori come input:

- *Json*:

Contiene una esatta copia del testo in input e dunque è veloce in scrittura.

- *Jsonb*:

E’ conservato con un formato binario decomposto. E’ un po’ più lento in scrittura e più veloce in lettura.

JSONB supporta la creazione di indici, in particolare indici GIN, che migliorano le prestazioni nelle ricerche su chiavi/valori.

JSONB normalizza la struttura dei dati per ottimizzare spazio e interrogazioni.

Normalmente i dati JSON vengono conservati in colonne di tipo JSONB, tranne dove si preferisce il JSON per motivi specifici.

Conversione fra *JSON primitive type* e *Postgres type:*

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

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

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