Documento a cura di Simone Remoli.

I PERMESSI

I permessi su una base di dati sono fondamentali e gestirli è necessario, perché è importante tenere traccia di chi ha accesso a determinate operazioni e chi no.
Una gestione accurata dei permessi è essenziale per garantire **sicurezza, integrità** e **controllo degli accessi** all'interno di un sistema informativo.
I permessi possono essere assegnati a **utenti** o **ruoli** e riguardano operazioni specifiche sui dati.

I principali tipi di permessi includono:

PERMESSI

- **Permessi di Lettura (SELECT)**
  - Consentono di visualizzare i dati senza modificarli.
    Ad esempio: un utente può eseguire una query **SELECT * FROM utenti**; ma non può modificare i dati.
- **Permessi di Scrittura (INSERT, UPDATE, DELETE)**
  - **INSERT**: Consente di aggiungere nuovi record.
    UPDATE: Consente di modificare i dati esistenti.
    DELETE: Consente di eliminare record.
- **Permessi di Esecuzione (EXECUTE)**
  - Consentono di eseguire funzioni o **stored procedure** senza accedere direttamente ai dati.
- **Permessi di Struttura (ALTER, CREATE, DROP)**
  - **CREATE**: Permette di creare nuove tabelle, viste, indici, ecc.
    ALTER: Consente di modificare la struttura di una tabella (es. aggiungere una colonna).
    DROP: Permette di eliminare tabelle o database.
- **Permessi di Controllo (GRANT, REVOKE, DENY)**
  - **GRANT**: Concede un permesso a un utente o un ruolo.
    REVOKE: Revoca un permesso precedentemente concesso.
    DENY: Impedisce esplicitamente l'accesso, anche se l'utente ha altri permessi.

Innanzitutto, **come si crea un utente su MySQL?**

Per creare un utente in MySQL, usa il comando **CREATE USER:**

```
CREATE USER 'nome_utente'@'host' IDENTIFIED BY 'password';
```

'host' → Indica da quale indirizzo IP o dominio l'utente può connettersi (ad esempio, **'localhost'** se l'utente accede solo dalla stessa macchina).

Assegnare permessi a un utente.

Una volta creato l'utente, devi assegnargli i permessi con il comando ***GRANT***:

```
GRANT permessi ON database.tabella TO 'nome_utente'@'host';
```

permessi → Specifica cosa può fare l'utente (ad esempio, SELECT, INSERT, UPDATE, DELETE, ALL PRIVILEGES, ecc.).

Altri permessi avanzati:

INDEX → Permette di **creare e rimuovere** indici nelle tabelle.
CREATE TRIGGER → Permette di **creare** trigger nel database.
REFERENCES → Permette di **definire** chiavi esterne (FOREIGN KEY).
SHOW VIEW → Permette di **visualizzare** la definizione di una vista.
CREATE VIEW → Permette di **creare** viste.
DROP VIEW → Permette di **eliminare** viste.
EXECUTE → Permette di **eseguire** stored procedures e funzioni.
CREATE ROUTINE → Permette di **creare** nuove stored procedures e funzioni.
ALTER ROUTINE → Permette di **modificare** stored procedures e funzioni.
DROP ROUTINE → Permette di **eliminare** stored procedures e funzioni.
GRANT OPTION → Permette all'utente di **assegnare** permessi ad altri utenti.
SUPER → Permette di **eseguire** operazioni amministrative, come interrompere query (KILL), cambiare configurazioni globali (SET GLOBAL).
CREATE USER → Permette di **creare** nuovi utenti MySQL.
DROP USER → Permette di **eliminare** utenti MySQL.
SHOW DATABASES → Permette di **vedere** tutti i database disponibili nel server.
PROCESS → Permette di **vedere** tutte le query in esecuzione sul server.

database.tabella → Specifica su quale database e tabella applicare i permessi (***.*** significa tutti i database e tutte le tabelle).

Esempio:

1.Utente con permessi completi (**Amministratore su un database**).

```
GRANT ALL PRIVILEGES ON negozio.* TO ‘admin'@'localhost';
```

2. Utente che può **leggere, scrivere ed eliminare record di una tabella** su un database chiamato **negozio.** Inoltre, vogliamo che possa assegnare questi stessi permessi ad altri utenti.

```
GRANT SELECT, INSERT, UPDATE, DELETE ON negozio.* TO
'mario'@'localhost' WITH GRANT OPTION;
```

Se vuoi dare a "mario" i permessi di lettura e scrittura, ma **NON permettergli di assegnarli ad altri**, allora **NON usare** WITH GRANT OPTION:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON negozio.* TO
'mario'@'localhost';
```

3. l'utente **proc_user** può eseguire solo le e funzioni nel database **negozio.**

```
GRANT EXECUTE ON negozio.* TO ‘proc_user'@'localhost';
```

Può **eseguire** qualsiasi **Stored Procedure** definita nel database **negozio.**
Se nel database **negozio** c'è una stored procedure chiamata **aggiorna_prezzi:**
Ora l'utente **proc_user** può eseguirla con:

```
CALL negozio.aggiorna_prezzi();
```

Se la procedura contiene una select su una tabella X MySQL restituirà un errore se l'utente non ha il permesso **SELECT** sulla tabella **X specificata nella procedura.**
Soluzione: Devi concedere anche il permesso **SELECT.**

```
GRANT SELECT ON negozio.prodotti TO ‘proc_user’@'localhost';
```

Dove prodotti è la tabella del db negozio che è chiamata nella store procedure.

EXECUTE permette di eseguire **Stored Procedures** e **Funzioni**. **Non dà accesso diretto alle tabelle**, quindi se una procedura usa **SELECT**, **INSERT**, **UPDATE**, l'utente avrà bisogno di quei permessi separatamente.

4.L'utente può creare e modificare la struttura del database, ma non può aggiungere, modificare o eliminare dati.

```
GRANT CREATE, ALTER, DROP, INDEX ON negozio.* TO
'designer'@'localhost';
```

5.L'utente può creare ed eliminare utenti e concedere loro permessi, ma non ha accesso ai dati.

```
GRANT CREATE USER, DROP USER ON *.* TO
'user_manager'@'localhost' WITH GRANT OPTION;
```

Non può concedere più permessi di quelli che ha ricevuto.

6.L'utente può leggere e inserire dati solo nella tabella prodotti.

```
GRANT SELECT, INSERT ON negozio.prodotti TO
'limited_user'@'localhost';
```

TRIGGER PERMESSI.

Una volta creato, un trigger è visibile a tutti gli utenti che hanno accesso alla tabella su cui è definito, ma non tutti gli utenti possono modificarlo o eliminarlo.

Cosa fa **SHOW TRIGGERS** in MySQL?

Mostra tutti i trigger definiti nel database attualmente selezionato.

Se vuoi vedere i trigger di un database specifico, devi usare:

```
SHOW TRIGGERS FROM nome_database;
```

Chi può eseguire SHOW TRIGGERS?

- Per eseguire SHOW TRIGGERS, l'utente deve avere **uno dei seguenti permessi** sul database:

- **SUPER** (permesso amministrativo globale).
- **SELECT su INFORMATION_SCHEMA.TRIGGERS**.
- **SHOW VIEW** sul database.
- **Qualsiasi permesso sulle tabelle con trigger associati** (SELECT, INSERT, UPDATE, DELETE).

Per permettere a **dev_user** di creare trigger che eseguono SELECT, devi assegnare anche il permesso SELECT sulla tabella coinvolta:

```
GRANT SELECT ON negozio.magazzino TO ‘dev_user'@'localhost';
```

Se il trigger deve anche **modificare** i dati, devi aggiungere UPDATE, INSERT, DELETE:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON negozio.magazzino TO ‘dev_user'@'localhost';
```

E alla fine:

```
GRANT CREATE TRIGGER ON negozio.* TO ‘dev_user’@‘localhost';
```

Se vuoi che un utente possa vedere i trigger interrogando INFORMATION_SCHEMA.TRIGGERS, concedi:

```
GRANT SELECT ON information_schema.TRIGGERS TO 'user1'@'localhost';
```

Ora, user1 può eseguire:

```
SELECT * FROM information_schema.TRIGGERS WHERE TRIGGER_SCHEMA = ‘negozio';
```

Se voglio vedere tutti i trigger del database **negozio** applicati alla tabella **ordini**, posso filtrare così:

```
SELECT * FROM information_schema.TRIGGERS WHERE TRIGGER_SCHEMA = 'negozio' AND EVENT_OBJECT_TABLE = ‘ordini’;
```

SHOW TABLES.

SHOW TABLES non è un permesso.
In MySQL **non esiste un permesso SHOW TABLES.**
Il comando funziona **automaticamente** in base ai permessi che l'utente ha sulle tabelle di un database.
Se un utente non ha alcun permesso su un database, il comando restituirà una lista vuota!

Se un utente esegue SHOW TABLES, vedrà solo le tabelle e le VIEW su cui ha uno di questi permessi:

- SELECT
- INSERT
- UPDATE
- DELETE
- CREATE
- DROP
- ALTER
- SHOW VIEW
- ALL PRIVILEGES

Quando user1 esegue: SHOW TABLES FROM negozio;

Risultato:

```
+---------------+
| Tables_in_negozio |
+----------------+
| clienti         |
+----------------+
```

Ovviamente se concedi a user1 solo il permesso SELECT sulla tabella clienti.

```
GRANT SELECT ON negozio.clienti TO 'user1'@'localhost';
```

SHOW DATABASES.

Se vuoi che un utente veda **tutti i database esistenti anche quelli su cui non ha accesso**, devi concedergli il permesso globale:

```
GRANT SHOW DATABASES ON *.* TO ‘user1’@'localhost';
```

Se un utente ha SHOW DATABASES ma non ha accesso ai database, può usare USE nome_database? No, se non ha permessi su quel database. L'utente può vedere il nome del DB nella lista dei database, ma non può accedervi.
Se vuoi permettere all’utente di usare il database e vedere le tabelle, devi concedere almeno il permesso SELECT.

Se un utente non ha nessun permesso su un database, non lo vedrà con SHOW DATABASES!

Supponiamo che nel server ci siano questi database:

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| negozio            |
| magazzino          |
+--------------------+
```

Se concediamo solo SELECT su negozio a user1:

```
GRANT SELECT ON negozio.* TO ‘user1'@'localhost';
```

Quando user1 esegue: **SHOW** DATABASES;

Risultato: user1 vede solo negozio, perché è l'unico database su cui ha un permesso.

```
+------------+
| Database   |
+------------+
| negozio    |
+------------+
```

Come concedere a un utente il permesso di assegnare qualsiasi permesso ad altri utenti in MySQL?

```
GRANT ALL PRIVILEGES ON *.* TO 'admin_user'@'localhost' WITH GRANT OPTION;
```

L'utente può:

Assegnare qualunque permesso ad altri utenti.
Creare nuovi utenti.
Modificare i permessi di utenti esistenti.

View.

Una **VIEW** è una tabella virtuale che mostra il risultato di una **query SQL** salvata.
Non contiene dati fisicamente, ma mostra i dati delle tabelle reali in un formato specifico.
Per lavorare con le View servono permessi specifici:

Permesso	Cosa permette di fare
`CREATE VIEW`	Creare nuove VIEW.
`SHOW VIEW`	Vedere le definizioni delle VIEW.
`ALTER VIEW`	Modificare una VIEW esistente.
`DROP VIEW`	Eliminare una VIEW.
`SELECT`	Permette di leggere i dati da una VIEW.

Esempio: dare a un utente il permesso di creare e vedere le VIEW in un database:

```
GRANT CREATE VIEW, SHOW VIEW ON negozio.* TO 'utente'@'localhost';
```

L’utente ora può creare nuove View nel database negozio.

Immagina di avere una tabella **prodotti**:

```
CREATE TABLE prodotti
( id INT PRIMARY KEY,
nome VARCHAR(100),
prezzo DECIMAL(10,2),
sconto DECIMAL(5,2) );
```

Se vuoi vedere solo i prodotti scontati, puoi **creare una VIEW**:

```
CREATE VIEW prodotti_scontati AS
SELECT nome, prezzo, sconto FROM prodotti WHERE sconto > 0;
```

Ora l’utente può vedere la definizione di questa View:

```
SHOW CREATE VIEW prodotti_scontati;
```

Ma cosa non può fare l’utente?

Non può usare SELECT sulle VIEW (serve il permesso SELECT).

- **Non può modificare (ALTER VIEW) le VIEW esistenti.**
- **Non può eliminare (DROP VIEW) le VIEW.**

Se vuoi che possa leggere i dati delle VIEW, devi concedere SELECT:

```
GRANT SELECT ON negozio.* TO ‘utente'@'localhost';
```

A questo punto l’utente può leggere i dati dalle View:

```
SELECT * FROM prodotti_scontati;
```

Se vuoi che l'utente possa anche **eliminare** le VIEW:

```
GRANT DROP VIEW ON negozio.* TO ‘utente'@'localhost';
```

A questo punto l’utente può fare: `DROP VIEW prodotti_scontati;`

Se vuoi che possa **modificare** le VIEW esistenti:

```
GRANT ALTER VIEW ON negozio.* TO ‘utente'@'localhost';
```

A questo punto l’utente può modificarla:

```
ALTER VIEW prodotti_scontati AS
SELECT nome, prezzo, sconto FROM prodotti WHERE sconto >= 10;
```

A cosa servono le VIEW?

Semplificare le query complesse.

Se hai una query complicata che usi spesso, puoi salvarla in una VIEW e richiamarla facilmente.

Revocare permessi a un utente.

Il comando **REVOKE** in MySQL viene utilizzato per **rimuovere** permessi che sono stati concessi a un utente con il comando GRANT.

Serve per revocare l'accesso a tabelle, database, funzioni, VIEW e altri oggetti.

SINTASSI:

```
REVOKE permesso ON nome_database.* FROM ‘utente'@'host';
```

Rimuovere più permessi nello stesso comando:

```
REVOKE INSERT, UPDATE, DELETE ON negozio.* FROM
‘user1'@'localhost';
```

Ora **user1** non può più modificare o eliminare dati nel database **negozio**.

Se un utente ha il permesso di assegnare permessi (**WITH GRANT OPTION**), puoi **revocare solo questa possibilità** senza rimuovere i suoi permessi.

```
REVOKE GRANT OPTION ON negozio.* FROM ‘user1'@'localhost';
```

Revocare **SELECT** solo su una tabella specifica:

```
REVOKE SELECT ON negozio.prodotti FROM ‘user1'@'localhost';
```

Ora **user1** può ancora accedere a **negozio**, ma non può più leggere i dati dalla tabella **prodotti**.

Revocare **EXECUTE** su una stored procedure:

```
REVOKE EXECUTE ON PROCEDURE negozio.aggiorna_prezzi FROM
'user1'@'localhost';
```

Come visualizzare tutte le Stored Procedure in un database MySQL?

Innanzitutto assegno i permessi che permettono all'utente di creare, modificare ed eseguire stored procedure.

```
GRANT CREATE ROUTINE, ALTER ROUTINE, EXECUTE ON
negozio.* TO 'user1'@'localhost';
```

Per vedere l'elenco di tutte le **Stored Procedure** presenti in un database MySQL usi il seguente comando:

```
SHOW PROCEDURE STATUS WHERE Db = 'nome_database';
```

Mostra tutte le Stored Procedure nel database **negozio.**

Se un utente non riesce a eseguire il comando sopra, puoi concedergli il permesso:

```
GRANT SELECT ON information_schema.ROUTINES TO
'user1'@'localhost';
```

Principio del Minimo Privilegio (PoLP).

Il principio stabilisce che gli utenti, i processi e le applicazioni dovrebbero avere solo i permessi strettamente necessari per svolgere il loro lavoro. Questo limita il rischio di errori, malfunzionamenti o attacchi informatici.

In un database MySQL un utente che deve solo leggere dati riceve **solo il permesso SELECT**, evitando accessi non necessari come **DELETE** o **DROP TABLE**.

Limitare i privilegi riduce la probabilità di **effetti collaterali imprevisti** dovuti alle azioni di utenti o applicazioni.
È più facile testare e controllare il comportamento del sistema.

Principio del Need-to-Know.

Il principio stabilisce che un utente deve poter accedere solo alle informazioni strettamente necessarie per svolgere il proprio compito, indipendentemente dal suo livello di sicurezza o altri privilegi.

Ad esempio, un impiegato in un'azienda potrebbe avere accesso ai dati dei clienti, **ma non ai dettagli finanziari**, a meno che non siano necessari per il suo ruolo.

Principio del Minimo Privilegio (Least Privilege) → Limita **i permessi di azione** di un utente (es. può solo leggere, non modificare).

Principio del Need-to-Know → Limita **l'accesso ai dati**, anche se l'utente ha il permesso di interagire con il sistema.

Esempio: Se un utente può vedere solo alcuni campi di una tabella, si può usare una **VIEW**:

CREATE VIEW clienti_pubblici AS
SELECT nome, città FROM clienti;

Ora, chi usa questa vista non vedrà dati sensibili come l'email o il numero di telefono.

Se qualcuno modifica i dati, la VIEW mostra ancora i dati vecchi?

No!

Le VIEW **non memorizzano i dati**, sono solo "finestre" sui dati reali.
Se qualcuno modifica la tabella di origine, la VIEW mostrerà **i dati aggiornati in tempo reale.**

La VIEW riflette immediatamente i dati aggiornati nella tabella originale.