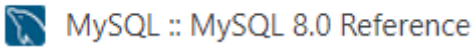
Gestire utenti e permessi con GRANT e REVOKE

CONCESSIONE PERMESSI CON GRANT e REVOCA DI PRIVILEGI TRAMITE REVOKE



Un problema molto frequente quando si lavora con database in ambito industriale è dover assegnare privilegi o permessi limitati a particolari utenti: normalmente è compito del sistemista o DBA (Database Administrator).

MySQL è molto flessibile a riguardo e ci viene incontro tramite gli statement **GRANT** e **REVOKE**.

Il comando **GRANT** permette allo stesso tempo di creare un utente e di assegnargli dei permessi specifici. Vediamone la sintassi:

```
GRANT <istruzioni_consentite>
ON <database>.<tabella>
TO <utente>@<host>
IDENTIFIED BY <password>;
```

La sintassi qui proposta è molto semplice, ma illustriamone comunque i singoli campi:

- **istruzioni_consentite**: è una lista di *statements* SQL che si vogliono permettere all'utente (CREATE, SELECT, UPDATE, DELETE, ALTER, EXECUTE, ecc..). Se si vogliono dare all'utente permessi completi si può utilizzare la parola chiave ALL.
- **database**: è il nome del database che stiamo prendendo in considerazione.
- **tabella**: inserendo il nome di una tabella, si fa riferimento solo ad essa. Per tutte le altre tabelle non varranno le regole che stiamo specificando. Se si vuole fare riferimento a tutte le tabelle si può utilizzare il carattere asterisco (*).
- **utente**: specifica il nome dell'utente che vogliamo creare
- **host**: specifica il/gli host da cui è ammessa la connessione
- **password**: specifica la password associata all'utente che stiamo creando. La password va scritta "in chiaro". Se si desidera inserire la password in forma criptata tramite la **funzione PASSWORD()** di MySQL, si deve far precedere la stringa criptata dalla parola PASSWORD.

Immaginiamo di stare gestendo un database per un Cinema. Immaginiamo che questo database si chiami "Cinema". Immaginiamo infine che questo cinema voglia che gli accessi al database avvengano solo da utenti di questi tre tipi:

- **Root**: questo è l'utente a cui **tutto è permesso**. L'utente root può leggere e modificare i dati delle tabelle; inserire nuove tuple; creare e cancellare tabelle e alterare la struttura di quelle già esistenti. Insomma, può fare tutto.
- **Amministratori**: questi utenti possono inserire, leggere, modificare e cancellare i dati (record) delle tabelle; possono impostare References**. Non possono però creare** nuove tabelle **o alterare la struttura** di tabelle esistenti (Alter, Drop).
- **Statisti**: questo tipo di utenti ha **accesso in lettura** a tutto al database ma non ha in nessun caso i permessi di scrittura. Questi utenti possono solo leggere i record delle varie tabelle, senza alterarli in nessun modo. Possono, insomma, solo utilizzare i dati per produrre delle statistiche.

Vediamo come tutto ciò si traduce in codice SQL. Partiamo dall' utente con più restrizioni (statisti) per arrivare a quello più libero (root):

```
GRANT SELECT
ON Cinema.*
TO 'cinema_stat'
IDENTIFIED BY 'statista_pass';

GRANT SELECT, INSERT, UPDATE, DELETE
ON Cinema.*
TO 'cinema_admin'
IDENTIFIED BY 'admin_pass';

GRANT ALL
ON Cinema.*
TO 'cinema_root@localhost'
IDENTIFIED BY PASSWORD '*6C8989366EA95CEF4';
```

In tutti e tre i casi si è utilizzata la sintassi **Cinema.*** per indicare che i permessi verranno applicati a tutte le tabelle del database.

Notiamo infine due particolarità utilizzate per l'utente root:

- Il suffisso "@localhost" (con la sintassi come da esempio) è un **controllo di sicurezza** che specifica che le connessioni dall'utente root possono essere accettate solo se provenienti da localhost (la macchina locale). E' possibile specificare anche uno specifico IP o un range di IP.
- La password dell'utente root non viene scritta in chiaro ma come **stringa hash** restituita dalla funzione PASSWORD() di MySQL.

Se si desidera assegnare ad un utente certi permessi da parte di **tutti gli host** possibili bisogna utilizzare il **carattere jolly: %**

Vediamo infine l'istruzione **REVOKE** che svolge la funzione opposta a GRANT, e cioè rimuovere permessi. Ne vediamo solo la sintassi in quanto molto simile all'istruzione GRANT:

```
REVOKE <istruzioni_revocate>
ON <database>.<tabella>
FROM <utente>;
```

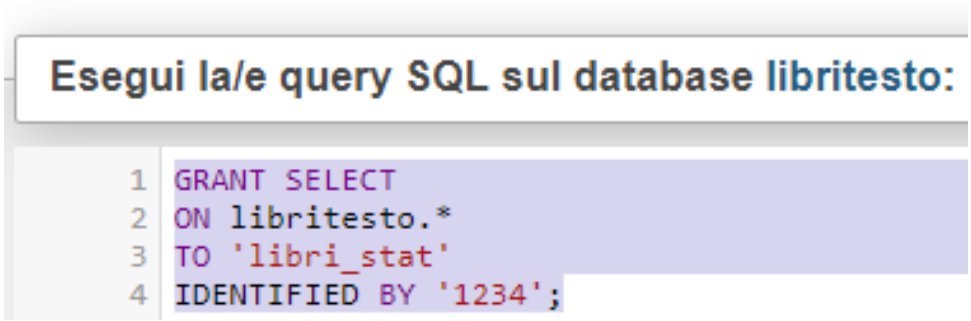
per la quale valgono le stesse regole sopra viste per GRANT.

Tratto da articolo online o tutorial

Altri link interessanti in slides
(scaricabili in formato pdf)

Esempio con uso phpMyAdmin in ambiente XAMPP

Selezionato il database *libritesto*

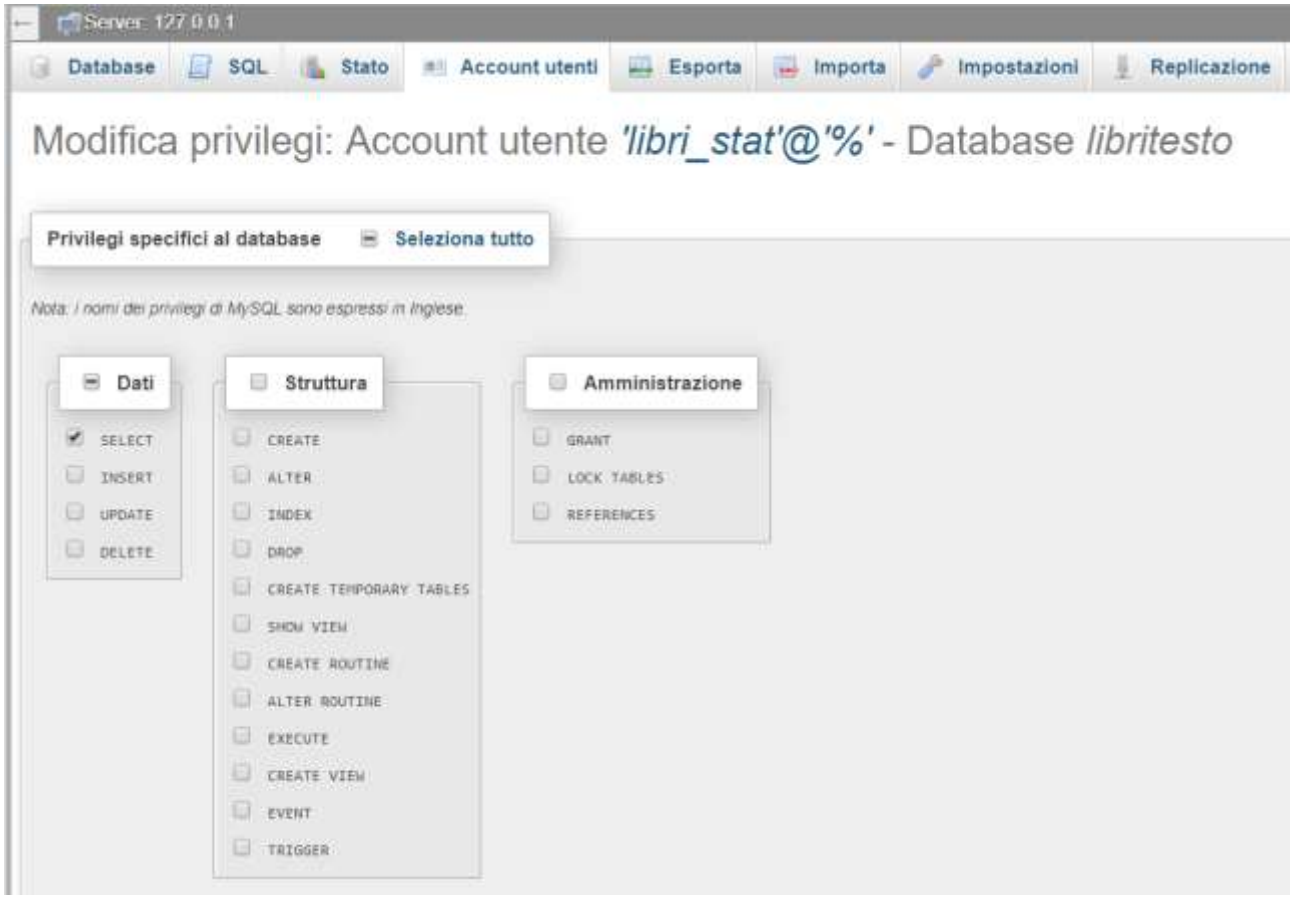


Si visualizza, tra i **Privilegi**:

Con possibilità di modifica semplificata

Rispetto alla sola possibilità

Basta un click per marcare/ smarcare:



Nb: con account free su altervista.org è negato l'accesso per creare utenti con privilegi modificabili

```
GRANT SELECT
ON my_new345.*
TO 'statista'
IDENTIFIED BY '1234'
```

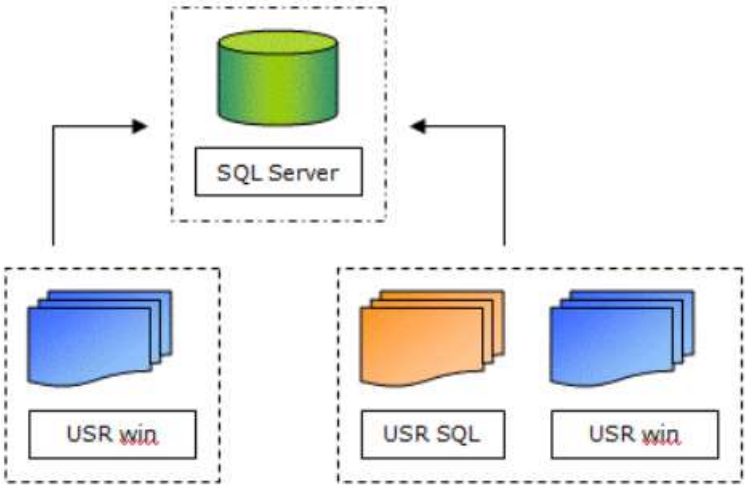
Messaggio di MySQL:

```
#1044 - Access denied for user 'new345'@'localhost' to database 'my_new345'
```

Due modi per l'autenticazione in MS SQL Server

Con **SQL Server** siamo in grado di gestire l'autenticazione in due modi:

- tramite gli utenti e i ruoli presenti in **windows**;



- tramite gli utenti di **SQL Server**.

Con questa doppia scelta possiamo applicare i ***permessi offline*** o per ***progetti online***.

Per gestione utenti, ruoli e account di accesso in MS SQL SERVER

in dettaglio per autorizzazioni: principio dei privilegi minimi
autorizzazioni baste sui ruoli
o tramite codice procedurale
catene di priorità

Istruzione di autorizzazione	Descrizione
GRANT	Consente di concedere un'autorizzazione.
REVOKE	Consente di revocare un'autorizzazione. Corrisponde allo stato predefinito di un nuovo oggetto. Un'autorizzazione revocata a un utente o a un ruolo può tuttavia ancora essere ereditata da altri gruppi o ruoli a cui è assegnata l'entità di sicurezza.
DENY	Consente di revocare un'autorizzazione in modo che non possa essere ereditata. Ha la precedenza su tutte le autorizzazioni, ma non si applica a proprietari di oggetti o a membri di `sysadmin`. Se si usa DENY per negare le autorizzazioni su un oggetto al ruolo `public`, le autorizzazioni verranno negate a tutti gli utenti e ruoli, ad eccezione dei parte proprietari dell'oggetto e dei membri di `sysadmin`.

Server Management Object