Gestire utenti e permessi con GRANT e REVOKE:

CONCESSIONE PERMESSI CON GRANT e REVOCA DI PRIVILEGI TRAMITE REVOKE

http://www.mrwebmaster.it/mysql/gestire-utenti-permessi-grant-revoke_7041.html
sito ufficiale: http://dev.mysql.com/doc/refman/5.7/en/grant.html

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**: E' una lista di istruzioni di SQL che si vogliono permettere all'utente (CREATE, SELECT, UPDATE, DELETE, ALTER, EXECUTE, ecc..). Se si vuole dare all'utente permessi completi si può utilizzare la parola chiave ALL.
- **database**: E' 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 tabella 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 nuovi record; creare e cancellare tabelle e alterarne 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 alterarne 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 tirarne fuori 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.