

# MySQL : Sécurité



# Introduction

# Pourquoi sécuriser ?

- Imaginez une altération des N° de CB
- Sabotage ? Maladresse ?
- C'est un travail collectif ( pas que le RSSI )  
Responsable de la Sécurité des Système d'information
- L'installation par défaut est permissive
  - Besoin d'agir -> Sécuriser l'installation

# Sécuriser le serveur

# Sécuriser l'installation

- Contrôler les droits ( voir plus loin )
- Mettre un mot de passe à root
- Créer plusieurs utilisateurs

- Modifier le mot de passe -

☐ aucun mot de passe

☒ Mot de passe:  Entrer à nouveau:

Hachage du mot de passe: ☒ MySQL 4.1+  
☐ compatible MySQL 4.0

Générer un mot de passe

# Sécuriser l'installation

	Utilisateur	Serveur	Mot de passe	Privilèges globaux 1	«Grant »	Action
<input type="checkbox"/>	N'importe quel	%	--	USAGE	Non	
<input type="checkbox"/>	N'importe quel	localhost	Non	USAGE	Non	
<input type="checkbox"/>	root	localhost	Oui	ALL PRIVILEGES	Oui	

- Supprimer les comptes anonymes  
-> Pas de mot de passe
- Supprimer le schéma test  
-> Risque saturation de l'espace de stockage

# Mise en pratique

## Modification de mot de passe

```
Query 1
1 • set password for root@localhost=password('Azerty11');
```

Dans la table user de la base mysql le mot de passe est crypté

```
1 • select user,host,password
2   from mysql.user
3  where user='root';
```

100% 26:1

Overview Output Snippets Query 1 Result

Filter:  Fetched 1 records.

user	host	password
root	localhost	*81F5E21E35407D884A6CD4A731AEBFB6AF209E1B

# Mise en pratique



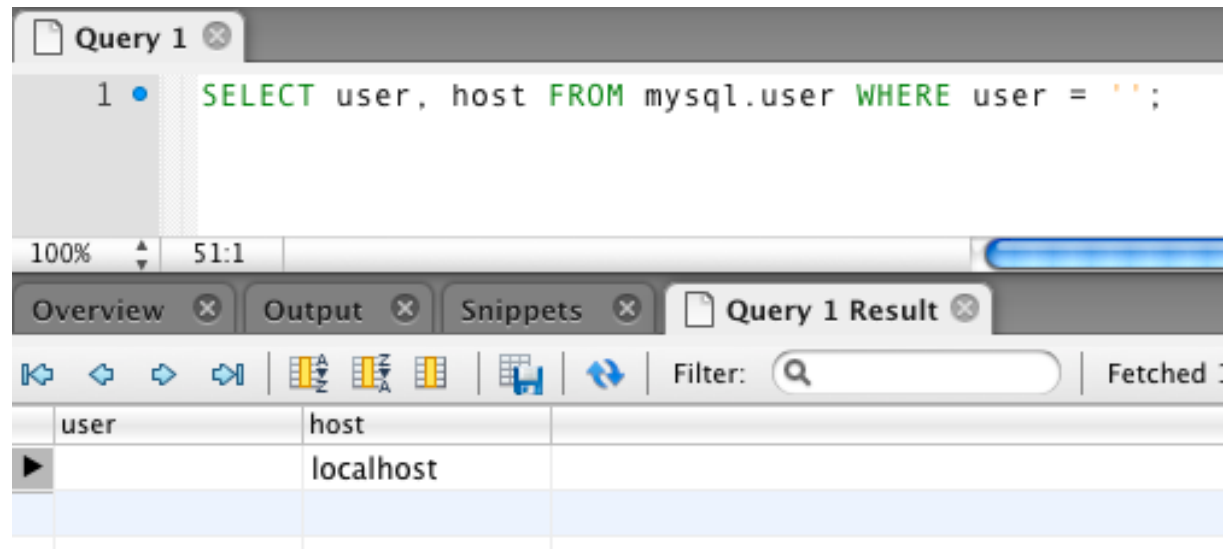
On peut renommer le compte root afin de le rendre plus difficile à trouver

```
Query 1 [X]  
1 • RENAME USER 'root'@'localhost' TO 'super'@'localhost';
```



# Mise en pratique

## Voir les comptes anonymes



The screenshot shows a MySQL query client interface. The query editor at the top contains the following SQL statement:

```
1 • SELECT user, host FROM mysql.user WHERE user = '';
```

Below the query editor, the results are displayed in a table with two columns: 'user' and 'host'. The first row shows an empty string for 'user' and 'localhost' for 'host', indicating an anonymous user.

user	host
	localhost

La requête nous dit qu'il y a un compte anonyme sur le host local

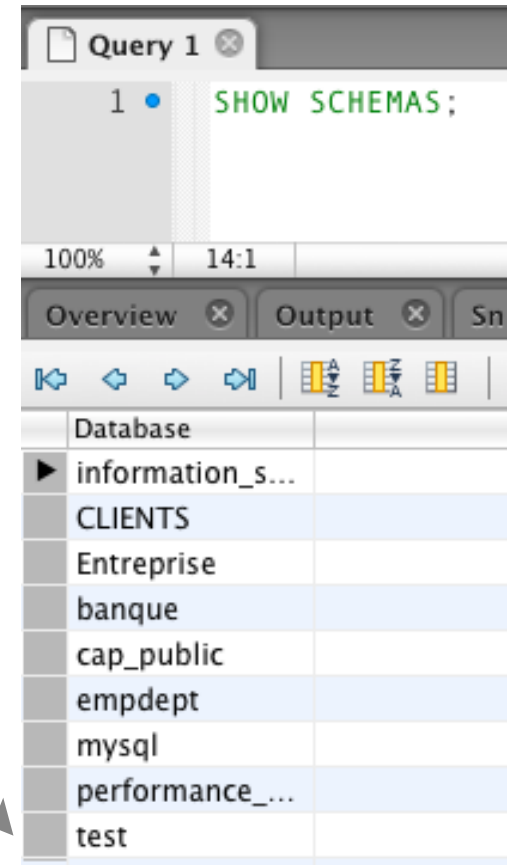
## Supprimer les comptes anonymes

**DROP USER ''@localhost;**

# Mise en pratique

Voir si il y a la base test

**SHOW SCHEMAS;**



Si on veut supprimer une base :

**DROP SCHEMA test;**

# Gestion des utilisateurs

# Gestion des utilisateurs

- Se fait dans la table user de la base mysql
- Un utilisateur est décrit via :
  - user@host
  - on peut donc avoir plusieurs root  
Ex : root@localhost, root@192.178.23.1  
Il s'agit ici de deux utilisateurs différents

# Gestion des utilisateurs

- La création d'un compte se fait avec  
CREATE USER

Exemple :

A screenshot of a SQL query editor window titled 'Query 1'. The editor shows a single line of SQL code: 'CREATE USER 'olivier'@'localhost' IDENTIFIED BY 'azerty';'. The code is color-coded: 'CREATE' is green, 'USER' is green, 'olivier' is orange, '@' is black, 'localhost' is orange, 'IDENTIFIED' is green, 'BY' is green, and 'azerty' is orange. A cursor is positioned at the end of the line.

```
Query 1
1 CREATE USER 'olivier'@'localhost' IDENTIFIED BY 'azerty';
2 |
```

- La suppression d'un utilisateur avec  
DROP USER

Exemple : DROP USER 'olivier'@'localhost';

# Gestion des utilisateurs

The image displays two side-by-side screenshots of a MySQL client interface, likely MySQL Workbench, showing the execution of SQL queries and the resulting output.

**Left Screenshot:** The 'Query 1' window shows two SQL statements:

```
1 CREATE USER 'lion'@'localhost' IDENTIFIED BY 'azerty11';
2 CREATE USER 'lion' IDENTIFIED BY 'Azerty11';
3
```

The 'Output' tab shows the execution results:

	Time	Action	Response
1	15:57:05	CREATE USER 'lion'@'localhost' IDENTIFIED BY 'azerty11'	0 row(s) affected
2	15:57:05	CREATE USER 'lion' IDENTIFIED BY 'Azerty11'	0 row(s) affected

**Right Screenshot:** The 'Query 1' window shows a single SQL statement:

```
1 SELECT user, host FROM mysql.user;
2
```

The 'Query 1 Result' tab shows the output of the query:

user	host
lion	%
lion	localhost
root	localhost

On a deux comptes : un compte si on utilise un client local dans tous les autres cas ça sera l'autre '@'%'

# Gestion des utilisateurs

Notion de joker -> %

On peut utiliser le joker afin d'affiner le filtrage. Par exemple :

```
CREATE USER 'Paul'@'%.bts.sio.net'
```

On peut se connecter en tant que Paul sur les machines dont le nom se termine par .bts.sio.net  
Autre exemple avec l'adresse IP :

```
CREATE USER 'Paul'@'192.168.%'
```

Donner des droits



# Commande GRANT

- Elle permet de donner des droits
- Créer le compte si il n'existe pas
- Syntaxe :

```
GRANT type_de_droits ON portée_des_droits  
TO compte_utilisateur;
```

- Type de droits : SELECT, PROCESS (voir les threads)...
- Portée\_des\_droits : tout le serveur, une base de données, une table ?

# Commande GRANT

- On distingue 4 types de droits :
  - Les droits d'administration
    - ➡ GRANT OPTION : permet de transmettre ces droits
    - ➡ SHUTDOWN : Arrêter le serveur
    - ➡ REPLICATION SLAVE : Récupérer les données du journal maitre
    - ➡ ...
  - Les droits sur les schémas ( BD )
    - ➡ ALTER, CREATE, DELETE, DROP, INSERT, SELECT, UPDATE...
  - Les droits au niveau des tables
    - ➡ ALTER, DROP, TRIGGER, INDEX, ...
  - Les droits au niveau des routines
    - ➡ EXECUTE, CREATE ROUTINE, ALTER ROUTINE

# Petits tests

# GRANT : affecter des privilèges

❑ **GRANT** ALL PRIVILEGES  
**ON** \*.\*  
**TO** admin ;

❑ Le script suivant donne tous les droits sur l'ensemble du serveur :

❑ **GRANT REPLICATION SLAVE**  
**ON** \*.\*  
**TO** replicateur@127.0.0.1 ;

❑ Le script suivant donne les droits nécessaires au compte replicateur (local) pour la réplication

# GRANT : affecter des privilèges

- ❑ `GRANT ALL`  
`ON bddgestion.*`  
`TO admin ;`
- ❑ Le script suivant donne tous les droits (sauf le droit d'accorder des droits) sur toutes les tables de la base *bddgestion* :
- ❑ `GRANT SELECT (nom, prénom),`  
`UPDATE (nom, prénom)`  
`ON bddgestion.user`  
`TO visiteur ;`
- ❑ Le script suivant donne les droits (à visiteur) `SELECT` et `UPDATE` uniquement sur les colonnes *nom* et *prénom* de la table *user*.



# GRANT : affecter des privilèges

- ❑ **GRANT SELECT**  
**ON** bddgestion.produit  
**TO** visiteur ;
- ❑ **GRANT SELECT, UPDATE, DELETE, INSERT**  
**ON** bddgestion.produit,  
bddgestion.gamme  
**TO** admin, visiteur;
- ❑ **GRANT SELECT**  
**ON** bddgestion.\*  
**TO** visiteur ;

Le script suivant donne le droit à **visiteur** d'exécuter la commande **SELECT** sur la table **produit** de la base **bddgestion** :

Le script suivant montre qu'on peut donner en une seule commande GRANT plusieurs privilèges, sur plusieurs tables, à plusieurs utilisateurs :

**nom-base.\*** permet de désigner toutes les tables d'une base de données :

# GRANT : affecter des privilèges

❑ **GRANT CREATE ROUTINE,  
EXECUTE ON** ps.\*  
**TO** 'create\_ps'@'localhost';

Donne le droit de créer et d'exécuter des procédures stockées dans le schéma ps

❑ **GRANT SHUTDOWN  
ON \*.\*  
TO** 'arret\_serveur'@'localhost';

Donne le droit d'arrêter le serveur MySQL à arret\_serveur ( local )

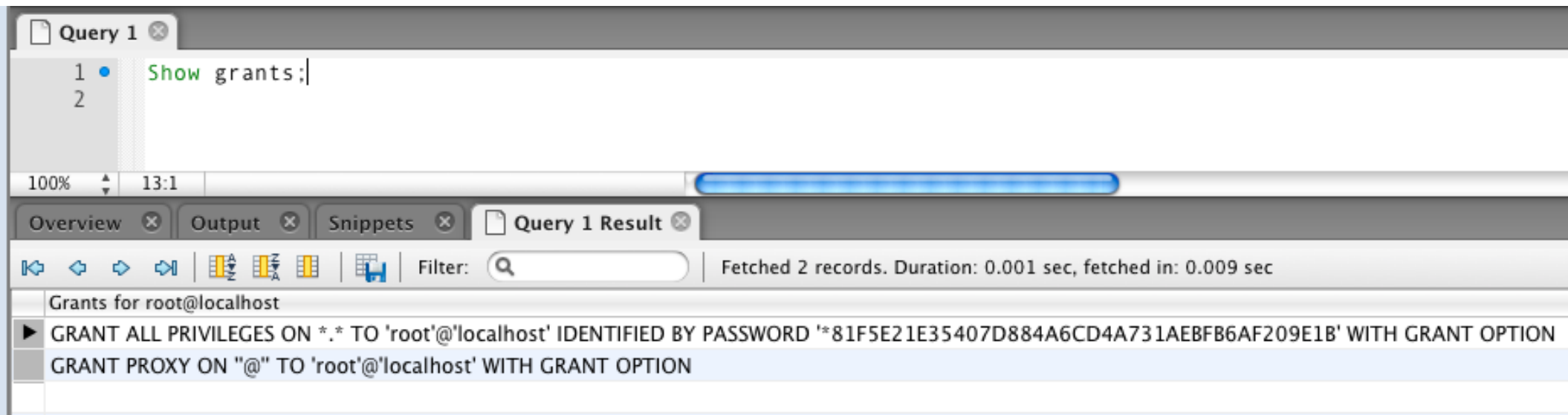
**Visualiser ses droits**



# Visualiser ses droits

## SHOW GRANTS;

- Permet de visualiser ses propres droits
- Ceux des autres si on a les droits



The screenshot shows a database client interface with a query editor and a results pane. The query editor contains the command `SHOW GRANTS;`. The results pane displays the output of the command, showing the grants for the `root@localhost` user. The results are as follows:

Grants for root@localhost
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED BY PASSWORD '*81F5E21E35407D884A6CD4A731AEBFB6AF209E1B' WITH GRANT OPTION
GRANT PROXY ON "" TO 'root'@'localhost' WITH GRANT OPTION

**Supprimer des droits**

# REVOKE:Retirer des droits

## **Syntaxe:**

- REVOKE type\_de\_droits ON portée\_des\_droits FROM compte;

*Remarque :*

Revoke retire les droits mais ne supprime pas l'utilisateur (-> DROP USER)

# REVOKE : retirer des privilèges

❑ **REVOKE UPDATE**  
**ON** bddgestion.**user**  
**FROM** visiteur ;

Le script suivant retire le droit de mise à jour sur la table *user* :

❑ **REVOKE ALL PRIVILEGES**  
**ON** bddgestion.**user**  
**FROM** visiteur ;

Le script suivant retire tous les privilèges de la table *user* à visiteur.

❑ **REVOKE GRANT OPTION**  
**FROM** secretaire ;

Le script suivant conserve les privilèges, mais retire le droit de les accorder à d'autres utilisateurs :

# TP

- Création d'un compte Admin sur photoForYou : **adminPhoto** nécessaire à la gestion de la base ( il a tous les droits nécessaires mais que sur la base PhotoForYou ). Ce compte se connecte en local.
- Création d'un compte pour les pages Web pour le code PHP ( Quels droits sont nécessaires sur la base PhotoForYou ? ). **webPhotoForYou**
- Remarques : une bonne pratique est de retirer tous les droits et d'accorder après que ceux nécessaires.