

o6– Définition et Manipulation d'une base de données

# SI3 – Exploitation des données

# Introduction

- SQL ne se limite pas à interroger une Base, on peut:
  - Créer des tables
  - Modifier la structure de tables existantes
  - Modifier le contenu des tables
  - ...
- 2 sous langages SQL:
  - LDD : langage de définition de données
  - LMD : langage de manipulation de données

# LMD – Insertion des données

- Dans le LMD, nous avons déjà vu l'instruction SELECT sous toutes ses coutures...
- Avant d'interroger une base, il a déjà fallu la remplir.
- Pour insérer :
  - INSERT INTO dansQuelleTable( listeDesChamps )  
VALUES ( listeDesValeurs );

# Exemple : Insertion

- Insertion de Robert Dupont né le 18 mars 1958 avec l'identifiant 11, dans la table personnel :

```
INSERT INTO personnel(idPersonnel, nom, prenom, ddn)  
VALUES (11, 'Dupont', 'Robert', '1958-03-18');
```

- Etant donné que l'on insère une valeur dans tous les champs :



```
INSERT INTO personnel  
VALUES (11, 'Dupont', 'Robert', '1958-03-18');
```

*Attention : l'ordre des champs est important et pas forcément connu... Donc à éviter...*

# La mise à jour d'une occurrence

- Une occurrence déjà insérée peut être modifiée champ par champ ou bien plusieurs en même temps; grâce à l'instruction :
  - UPDATE nomTable
  - SET nomChamp=nouvelleValeur
  - WHERE conditionDeMàJ

# Exemple Màj

- Mme Myriam MARTIN s'est mariée avec Jean-Pierre Dupont, elle change donc de nom:

```
UPDATE Personnel  
SET nom = "DUPONT"  
WHERE nom = "MARTIN"  
      AND prenom = "Myriam" ;
```

# MàJ : Utilisation ancienne valeur

- On a le droit d'utiliser l'ancienne valeur du champ dans l'affectation de la nouvelle valeur
- Par exemple, pour augmenter de 10% les prix de tous les produits du rayon 2 :

```
UPDATE Produit  
SET prixProd = prixProd * 1.1  
WHERE idRayon = 2 ;
```

# MàJ :Where facultatif

- Que se passe t'il si la clause WHERE est absente du UPDATE :
- -> Pas de condition de filtrage
- -> Toutes les lignes sont affectées par la màj



# MàJ de plrs champs

- Si plusieurs champs doivent être mäj, inutile de créer plusieurs requêtes de mäj
- Exemple : Augmenter de 10% le prix de tous les produits du rayon 2 et les passer dans le rayon 3:

```
UPDATE Produit  
SET prixProd = prixProd * 1.1 , idRayon = 3  
WHERE idRayon = 2 ;
```

# Utiliser requête pour màj ou insert

- On peut utiliser le résultat d'une requête pour une mise à jour ou l'insertion de données.
- A savoir que chaque valeur utilisée dans une insertion ou bien une màj peut être le résultat d'un SELECT.

# Exemple avec plrs colonnes

- Exemple : Affecter le prix moyen de tous les produits du rayon 6 aux produits du rayon 7.

```
UPDATE Produit
SET pxUnit = (Select avg( pxUnit )
              from ( select * from Produit ) as P
              WHERE idRayon = 6 )
WHERE idRayon = 7 ;
```

- Il est interdit sous MYSQL de mettre à jour une table à partir de données issues de la même table. Obligation de créer une table temporaire

# Suppression d'occurrences

- Pour supprimer des occurrences dans une table, on utilise l'ordre SQL DELETE
- Sa syntaxe:
  - DELETE FROM nomTable
  - WHERE conditionSuppression ;
- Une fois de plus, la clause WHERE est optionnelle mais présente le plus souvent.
- Sans WHERE, Vidange de la table spécifiée.

# Exemple suppression

- Supprimer toutes les personnes nées après le 24 juin 1980 et dont le nom commence par MAR.

```
DELETE FROM personnel  
WHERE ddn > '1980-06-24'  
AND nom LIKE "MAR%";
```

- Vider la table Produit

```
DELETE FROM Produit ;
```

- Certains clients SQL n'acceptent pas (par sécurité) l'utilisation de DELETE FROM sans WHERE.

# Exercices - 1

- Ajouter le passage en caisse du 28 juillet 2011 à 17h34 sur la caisse 3
- L'employé Julien MARTIN a fourni sa date de naissance : 17 juillet 1968
- Réduction de 5 Euros sur tous les produits du rayon Boucherie.
- Mme Dupont Ginette s'est mariée avec le manager du rayon Boucherie. Mettre à jour son nom.

## Exercice – 2

- Monsieur Robert Laroche est décédé. Le supprimer de la base. Il était manager du Rayon Habillement. Il est remplacé par Julien Durand né le 25 janvier 1978 qui vient d'être embauché. (3 requêtes)

# LDD – Création de table

- Syntaxe la plus simple sans contrainte:
  - CREATE TABLE nomDeTable(
    - nomColonne1 typeColonne1 ,
    - nomColonne2 typeColonne2,
    - ...
  - )



# LDD – Example CREATE TABLE

```
CREATE TABLE caisse(  
    idCaisse int(10) ,  
    DateProd date  
);
```

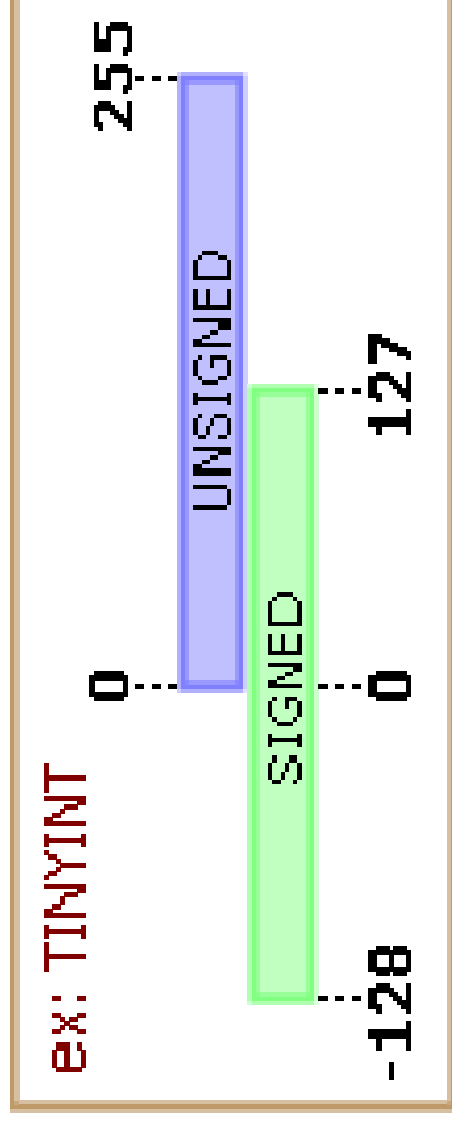
# Les types de colonne

- TINYINT : Entier de 0 à 255 (unsigned)
- SMALLINT : Entier de 0 à 65535 (unsigned)
- MEDIUMINT : Entier de 0 à 16777215 (unsigned)
- INT : Entier de 0 à 4294967295 (unsigned)
- BIGINT : Entier de 0 à 18446744073709551615 (unsigned)
- DECIMAL : Un nombre à virgule flottante
- DATE : Une date, va de '1000-01-01' à '9999-12-31'
- DATETIME : Date et Heure, va de '1000-01-01 00:00:00' à '9999-12-31 23:59:59'

# Les types de colonnes

- **TIMESTAMP** : Date et Heure exprimée en secondes depuis le 1er janvier 1970. Va de '1970-01-01 00:00:00' à quelque part, durant l'année 2037
- **TIME** : Une mesure de l'heure, va de '-838:59:59' à '838:59:59'
- **YEAR** : Une année, va de 1901 à 2155
- **CHAR** : Chaîne de caractère de taille fixe, va de 1 à 255 caractères
- **VARCHAR** : Chaîne de caractère de taille variable, va de 1 à 255 caractères
- **TINYTEXT ou TINYBLOB** : Un objet BLOB ou TEXT, longueur maximale de 255
- **TEXT ou BLOB** : Un objet BLOB ou TEXT, longueur maximale de 65535
- **MEDIUMTEXT ou MEDIUMBLOB** : Un objet BLOB ou TEXT, longueur maximale de 16777215
- **LONGTEXT ou LONGBLOB** : Un objet BLOB ou TEXT, longueur maximale de 4294967295

# Unsigned



# Propriétés de colonne

- Une propriété peut s'ajouter pour chaque colonne après le type:
  - AUTO\_INCREMENT : Incrémente de 1 la valeur de la colonne à chaque insertion. Souvent utilisé pour la clef primaire.
  - NOT NULL : Indique l'obligation de mettre une valeur dans le champ.
  - UNIQUE : Les valeurs dans la colonne doivent toutes être différentes les unes des autres.
  - DEFAULT : Spécifie une valeur par défaut à l'insertion d'une occurrence (ligne) pour le champ en question.

# Exemple propriétés

```
CREATE TABLE caisse(  
    idCaisse int(10) unsigned NOT NULL AUTO_INCREMENT,  
    DateProd date NOT NULL DEFAULT '2011-09-01'  
);
```

# Contraintes PRIMARY KEY

- La clef primaire
  - Se définit par la contrainte PRIMARY KEY
  - Obligatoire dans une table
  - On ne peut la trouver qu'une seule fois par table, mais peut être une concaténation de champs
  - Possibilité de l'écrire à la suite de la définition de la colonne si PK sur un seul champ
  - PK = UNIQUE + NOT NULL

# Exemples PRIMARY KEY

- Sur 1 seule colonne :

```
CREATE TABLE `caisse` (  
  `idCaisse` int(10) unsigned NOT NULL AUTO_INCREMENT,  
  `DateProd` date NOT NULL DEFAULT '0000-00-00',  
  PRIMARY KEY (`idCaisse`)  
);
```

- Sur plusieurs colonnes :

```
CREATE TABLE `contenir` (  
  `idPassageCaisse` int(10) unsigned NOT NULL DEFAULT '0',  
  `idProduit` int(10) unsigned NOT NULL DEFAULT '0',  
  `qte` int(10) unsigned DEFAULT NULL,  
  PRIMARY KEY (`idPassageCaisse`, `idProduit`)  
);
```



# Exemple PK après définition champ

```
CREATE TABLE `caisse` (  
  `idCaisse` int(10) unsigned AUTO_INCREMENT PRIMARY KEY,  
  `DateProd` date NOT NULL DEFAULT '0000-00-00'  
);
```

# Contraintes FOREIGN KEY

- Applique la contrainte d'intégrité référentielle sur une clé étrangère:
- Plusieurs FOREIGN KEY autorisées dans la création d'une table
- Syntaxe :

**FOREIGN KEY champLocal REFERENCES AutreTable( champDistant )**

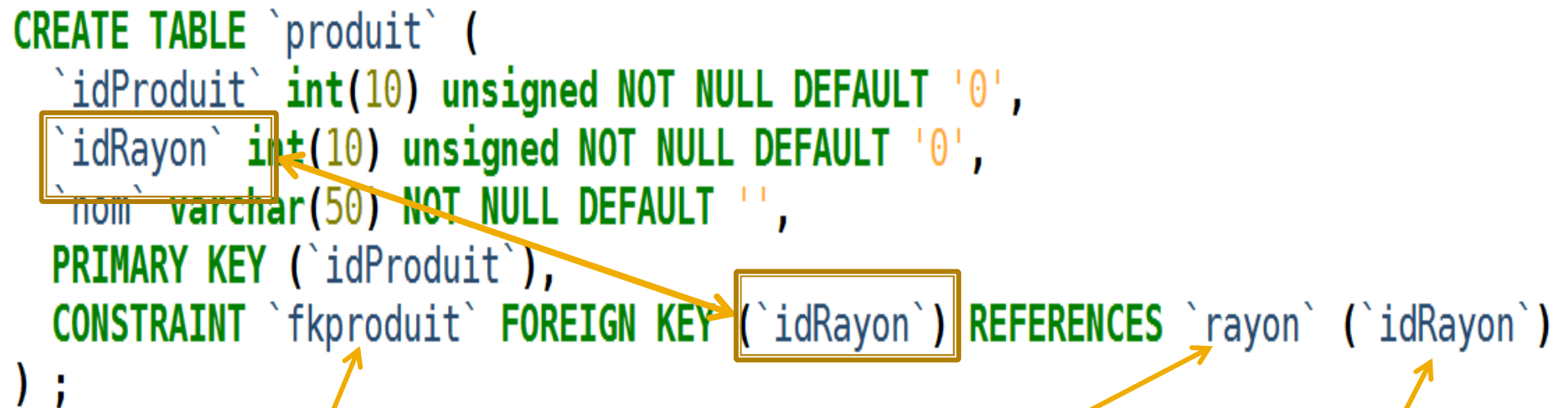
- Possibilité de l'écrire à la suite de la définition de la colonne

# Nommer les contraintes

- Il est conseillé de « nommer une contrainte » afin de pouvoir la retrouver.
- Pour nommer une contrainte :
  - `CONSTRAINT nomContrainte laContrainte`

# Exemple contrainte FK

```
CREATE TABLE `produit` (  
  `idProduit` int(10) unsigned NOT NULL DEFAULT '0',  
  `idRayon` int(10) unsigned NOT NULL DEFAULT '0',  
  `nom` varchar(50) NOT NULL DEFAULT '',  
  PRIMARY KEY (`idProduit`),  
  CONSTRAINT `fkproduit` FOREIGN KEY (`idRayon`) REFERENCES `rayon` (`idRayon`)  
);
```



Nom de la contrainte

Table que l'on veut relier

Colonne que l'on veut relier

# Contrainte CHECK

- Utilisé pour réduire un domaine de définition
- Exemple pour vérifier que la taille d'une personne ne dépasse pas 300 cm.

```
CREATE TABLE PersonneGrande (  
  `idPers` int(10) unsigned AUTO_INCREMENT PRIMARY KEY,  
  `taille` int(3) zerofill unsigned,  
  check (taille < 300 and taille > 0 )  
);
```

Malheureusement, cette contrainte est ignorée par le SGBD MYSQL

# Gestion plus fine de la contrainte d'intégrité référentielle

- Lorsque la valeur d'un champ FK est modifiée le système va vérifier que la nouvelle valeur possède bien une correspondance dans la table liée en tant que PK.
- Lorsque qu'une occurrence est supprimée dans une table et que sa PK est utilisée comme référence FK dans une autre table : le SGBD refuse.

# Options sur FK : SUPPRESSION

- Possibilité d'ajouter des options à la création de la contrainte FK:

- ON DELETE CASCADE:

- Suppression des occurrences liées

```
CONSTRAINT fk1 FOREIGN KEY (idPersonnel) REFERENCES responsable(idPersonnel) ON  
DELETE CASCADE
```

- ON DELETE SET NULL

- Mise à NULL du champ FK des occurrences liées

- ON DELETE RESTRICT : interdit la suppression

- ON DELETE NO ACTION : interdit la suppression (par défaut si ON DELETE pas spécifié)

# Options sur FK : Mise à jour

- ON UPDATE CASCADE :
  - Màj des occurrences liées en cas d'UPDATE
- ON UPDATE RESTRICT :
  - interdit la màj



# Exemple Complet

```
CREATE TABLE contenir2(  
  `idPassageCaisse` int(10) unsigned NOT NULL DEFAULT '0',  
  `idProduit` int(10) unsigned NOT NULL DEFAULT '0',  
  `qte` int(10) unsigned DEFAULT NULL,  
  PRIMARY KEY (`idPassageCaisse`, `idProduit`),  
  CONSTRAINT `fkcontenir-produit` FOREIGN KEY (`idProduit`)  
    REFERENCES `produit` (`idProduit`)  
    ON DELETE NO ACTION ON UPDATE CASCADE,  
  CONSTRAINT `fkcontenir-passage` FOREIGN KEY (`idPassageCaisse`)  
    REFERENCES `passagecaisse` (`idPassageCaisse`)  
    ON DELETE NO ACTION ON UPDATE NO ACTION  
);
```

- Interdit la Suppression d'un produit lié dans contenir
- Si màj de la PK d'un produit lié -> màj dans la table contenir

# 1 FK sur plusieurs colonnes

- Il arrive qu'une clef primaire soit composée de plusieurs colonnes.
- Il peut donc arriver qu'une colonne FK fasse référence à une clef primaire composée de plusieurs champs. (cas identifiant relatif)
- Exemple :

```
CREATE TABLE RESERVATION(  
    id_resa INT PRIMARY KEY,  
    id_hotel INT,  
    num_ch INT,  
    date DATE NOT NULL,  
    CONSTRAINT fkRes FOREIGN KEY (id_hotel, num_ch)  
        REFERENCES CHAMBRE (id_hotel, num_ch)  
);
```

# Modification de la structure d'une table

- Même déjà créée, il est possible de modifier la structure d'une table grâce à la commande `ALTER TABLE`:
  - Ajout, suppression ou modification d'un champ
  - Ajout, suppression ou modification d'une contrainte.

# Ajouter 1 ou plrs champ

- Pour ajouter la colonne pxCaisse de type float à la table caisse :

```
ALTER TABLE caisse  
ADD COLUMN pxCaisse FLOAT;
```

- Possibilité d'ajouter plusieurs colonne dans le même ALTER TABLE :

```
ALTER TABLE caisse  
ADD COLUMN `ajout1` VARCHAR(45),  
ADD COLUMN `ajout2` VARCHAR(45);
```

# Modifier 1 ou plrs champs

- Pour redéfinir le champ pxCaisse en entier au lieu de float:

```
ALTER TABLE caisse MODIFY pxCaisse INT ;
```

- Pour modifier plusieurs champs :

```
ALTER TABLE caisse  
MODIFY Column DateProd DATETIME DEFAULT '2011-09-01' ,  
MODIFY Column pxCaisse INT ;
```

# Supprimer un champ

- Pour supprimer la colonne pxCaisse de la table caisse :

```
ALTER TABLE caisse DROP COLUMN pxCaisse ;
```

# Supprimer une table

- On peut supprimer une table grâce à la commande DROP TABLE :

**DROP TABLE** contenir2 ;

# Renommer une table

- Pour renommer une table sans avoir à la supprimer et la recréer, utiliser RENAME TABLE.
- Par exemple, renommer la table Rayon en Rayonnage :

```
RENAME TABLE Rayon TO Rayonnage;
```



# Ajout, suppression contraintes

- Ajout de la contrainte FK à la table contenir qui fait référence à la table produit:

```
ALTER TABLE contenir  
    ADD CONSTRAINT fk1 FOREIGN KEY (idProduit)  
        REFERENCES Produit( idProduit ) ;
```

- Pour supprimer cette clef étrangère

```
ALTER TABLE contenir  
    DROP FOREIGN KEY fk1 ;
```



D'où l'utilité de donner un nom à la contrainte dès la création...

# Les vues

- Tables virtuelles issues d'un SELECT
- Peut être utilisée dans un autre SELECT
- Souvent en lecture seule
- Suppression avec DROP VIEW
- Exemple:

```
CREATE VIEW v_ticket_caisse ( numeroT , totalT )  
  AS  SELECT idPassageCaisse , SUM(qte*pxUnitV)  
        FROM Contenir  
        INNER JOIN Produit ON Produit.idProduit = Contenir.idProduit  
        GROUP BY idPassageCaisse ;
```

```
DROP VIEW v_ticket_caisse ;
```

	numeroT	totalT
▶	1	969
	2	1105
	3	1586
	4	1051
	5	710
	6	931

# Utilisateurs

- Utilisateurs et droits stockés dans la bd « mysql », dans la table « user »
- Ajout d'utilisateurs:
  - Avec une requête « insert »
  - Ou avec « create user »
  - Ou avec assistant graphique.
- Exemple:

```
CREATE USER 'userTest'@'localhost' IDENTIFIED BY 'toto';
```

# Suppression utilisateurs

- Avec DELETE
- Ou DROP USER :

```
DROP USER userTest ;
```

# Droits

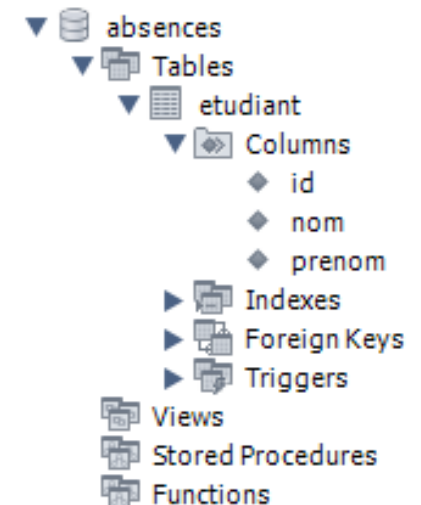
- Par défaut aucun droit
- Ajout de privilèges avec GRANT
- Suppression de privilèges avec REVOKE
- Droits spécifiques à un emplacement (ip)

# Exemple GRANT

- Pour autoriser l'utilisateur « userTest » :
  - à effectuer un « select » sur les colonnes
    - id
    - nom
    - prenom
  - De la table « etudiant » de la bd « absences »
  - Depuis n'importe quel emplacement :

GRANT

```
SELECT (id,nom,prenom )  
ON table absences.etudiant  
TO userTest@'%' ;
```



# Exemple REVOKE

- Pour interdire l'utilisateur userTest d'effectuer:
  - Un select
  - Sur la colonne id
  - De la table etudiant
  - De la bd absences

```
REVOKE
  SELECT (id)
  ON absences.etudiant
  FROM userTest@'%' ;
```

# Spécificités

- Pour tous les privilèges:
  - All Privileges dans le GRANT
- Pour toutes les tables d'une base « bdToto »
  - bdToto.\* dans le ON
- Pour toutes les bases du serveur
  - \*.\* dans le ON
- Depuis n'importe où:
  - % dans le TO après le @
- Exemple :

```
GRANT
All Privileges
ON *.*
TO userTest@'%' ;
```



# Flush

- Afin de forcer le SGBD à relire les droits:
  - Flush privileges ;

# Listes des droits Mysql

Privilege	Column	Context
<u>CREATE</u>	Create_priv	databases, tables, or indexes
<u>DROP</u>	Drop_priv	databases, tables, or views
<u>GRANT OPTION</u>	Grant_priv	databases, tables, or stored routines
<u>LOCK TABLES</u>	Lock_tables_priv	databases
<u>REFERENCES</u>	References_priv	databases or tables
<u>EVENT</u>	Event_priv	databases
<u>ALTER</u>	Alter_priv	tables
<u>DELETE</u>	Delete_priv	tables
<u>INDEX</u>	Index_priv	tables
<u>INSERT</u>	Insert_priv	tables or columns
<u>SELECT</u>	Select_priv	tables or columns
<u>UPDATE</u>	Update_priv	tables or columns
<u>CREATE TEMPORARY TABLES</u>	Create_tmp_table_priv	tables
<u>TRIGGER</u>	Trigger_priv	tables
<u>CREATE VIEW</u>	Create_view_priv	views
<u>SHOW VIEW</u>	Show_view_priv	views
<u>ALTER ROUTINE</u>	Alter_routine_priv	stored routines
<u>CREATE ROUTINE</u>	Create_routine_priv	stored routines
<u>EXECUTE</u>	Execute_priv	stored routines
<u>FILE</u>	File_priv	file access on server host
<u>CREATE TABLESPACE</u>	Create_tablespace_priv	server administration
<u>CREATE USER</u>	Create_user_priv	server administration
<u>PROCESS</u>	Process_priv	server administration
<u>PROXY</u>	see proxies_priv	server administration
<u>RELOAD</u>	Reload_priv	server administration
<u>REPLICATION CLIENT</u>	Repl_client_priv	server administration
<u>REPLICATION SLAVE</u>	Repl_slave_priv	server administration
<u>SHOW DATABASES</u>	Show_db_priv	server administration
<u>SHUTDOWN</u>	Shutdown_priv	server administration
<u>SUPER</u>	Super_priv	server administration
<u>ALL PRIVILEGES</u>		server administration
<u>USAGE</u>		server administration

# Fin

- Questions

