Support de cours
Introduction à MySQL

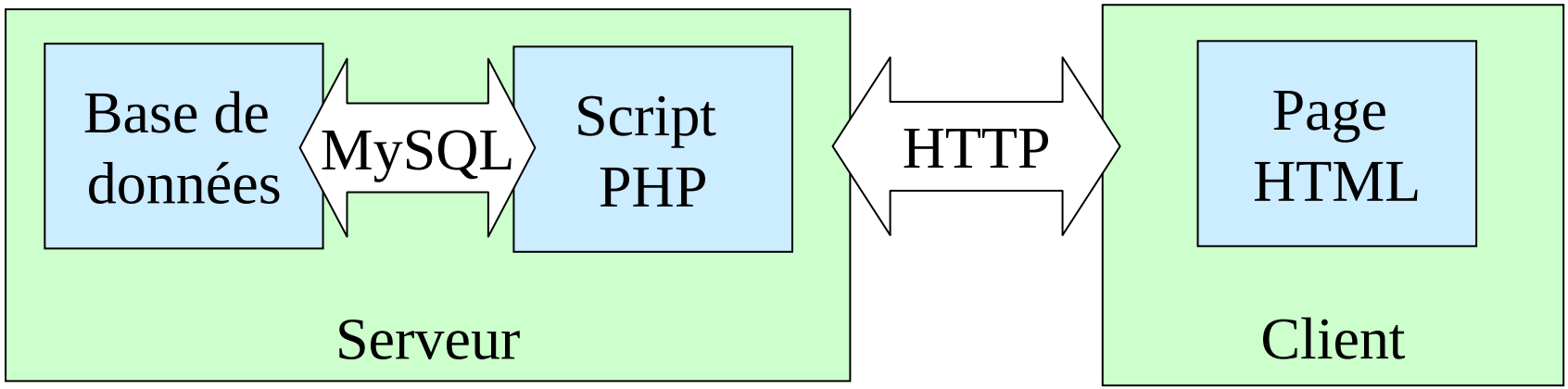
Généralités

Introduction

- MySQL dérive directement de SQL (Structured Query Language) qui est un langage de requête vers les bases de données exploitant le modèle relationnel.
- Il en reprend la syntaxe mais n'en conserve pas toute la puissance puisque de nombreuses fonctionnalités de SQL n'apparaissent pas dans MySQL (sélections imbriquées, clés étrangères…)
- Le serveur de base de données MySQL est très souvent utilisé avec le langage de création de pages web dynamiques : PHP

Architecture

MySQL est basé sur une architecture client/serveur. C'est-à-dire que les clients doivent s'adresser au serveur qui gère, contrôle et arbitre les accès aux données.



Concepts du modèle relationnel

Avant d'attaquer le vif du sujet, un petit glossaire du jargon des bases de données :

- Domaine : ensemble des valeurs d'un attribut.
- Relation : sous ensemble du produit cartésien d'une liste de domaines. C'est en fait un tableau à deux dimensions dont les colonnes correspondent aux domaines et dont les lignes contiennent des tuples. On associe un nom à chaque colonne.
- Attribut : une colonne d'une relation, caractérisé par un nom.
- Tuple : liste des valeurs d'une ligne d'une relation.

- Une relation est un peu une classe (programmation orientée objet) qui ne possèderait que des attributs et donc chaque instance représenterait un tuple.

Objets

Un **serveur** MySQL gère une ou plusieurs **base de données**.

Chaque base de données contient différents types d'objets (tables, index, fonctions).

L'objet le plus représenté d'une base de données est la **table**.

Chaque table (appelées encore « relation ») est caractérisée par une ou plusieurs **colonnes** (ou « attributs »).

Le langage qui permet de gérer ces objets est appelé « *Langage de Description des Données* » (LDD).

Les données sont stockées dans les tables sous forme de **lignes**

(ou « tuples »).

Le langage qui permet de manipuler les données est appelé

« *Langage de Manipulation des Données* » (LMD).

Clés primaires et étrangères

Une table contient généralement une **clé primaire**.

Une clé primaire est constituée d'une ou plusieurs colonnes.

Les valeurs des colonnes qui constituent la clé primaire d'une table sont uniques pour

toutes les lignes de la table. La clé primaire d'une table permet donc de faire référence de manière univoque à chaque ligne de la table.

Par exemple: Commandes (Id, pizza_Id, Date_Commande)

Pizzas (Id, Nom, taille, prix)

les tables pizzas et commandes possèdent toutes deux une clé primaire qui est constituée de leur colonne id respective.

La colonne pizza_id de la table commandes fait référence à la colonne id (donc à la clé primaire) de la table pizzas.

Ce type de référence est appelée « **clé étrangère** ». MySQL n'implémente pas les clés étrangères sous forme de contrainte.

Clés uniques

- En complément à la clé primaire, une ou plusieurs clés uniques (appelées également « clés secondaires » ou « clés alternatives ») peuvent être associées à une table.
- Les clés uniques sont identiques aux clés primaires à la différence prés que les colonnes qui les constituent peuvent contenir une valeur NULL.
- La plupart du temps, toute table devrait posséder au moins une clé primaire.

Interfaces à MySQL

La plupart du temps l'accès aux bases de données se fait via un autre langage de programmation (C, Perl, PHP, etc.).

Cependant, il est régulièrment nécessaire d'exécuter des requêtes sur la base de données, ne serait-ce que pour construire les objets avec le LDD.

L'interface qui est traditionnellement utilisée est l'utilitaire mysql. Il s'agit d'un

programme en ligne de commande qui permet d'exécuter des requêtes SQL et d'en visualiser les éventuels résultats. Un exemple est donné page suivante.

Cependant, grâce à la grande popularité de MySQL, d'autres interfaces plus conviviales ont vu le jour. Les plus utilisées sont :

- phpMyAdmin est, comme son nom l'indique, une interface Web écrite en PHP;

- Webmin, il existe un module d'administration de MySQL pour Webmin ;

- WinMySQLAdmin est une interface pour les systèmes Windows ;

- MySQLCC (« MySQL Control Center »).

L'interface la plus répandue est phpMyAdmin.

Syntaxe de MySQL

Types des attributs (I)

Les propriétés de vos objets peuvent être de types très différents :

- Nombre entier signé ou non (température, quantité commandée, âge)
- Nombre à virgule (prix, taille)
- Chaîne de caractères (nom, adresse, article de presse)
- Date et heure (date de naissance, heure de parution)
- Énumération (une couleur parmi une liste prédéfinie)
- Ensemble (une ou des monnaies parmi une liste prédéfinie)

Il s'agit de choisir le plus adapté à vos besoins.

Ces types requièrent une plus ou moins grande quantité de données à stocker. Par exemple, ne pas choisir un LONGTEXT pour stocker un prénom mais plutôt un VACHAR(40) !

Types des attributs (II) – entiers

nom	borne inférieure	borne supérieure
TINYINT	-128	127
TINYINT UNSIGNED	0	255
SMALLINT	-32768	32767
SMALLINT UNSIGNED	0	65535
MEDIUMINT	-8388608	8388607
MEDIUMINT UNSIGNED	0	16777215
INT*	-2147483648	2147483647
INT* UNSIGNED	0	4294967295
BIGINT	-9223372036854775808	9223372036854775807
BIGINT UNSIGNED	0	18446744073709551615

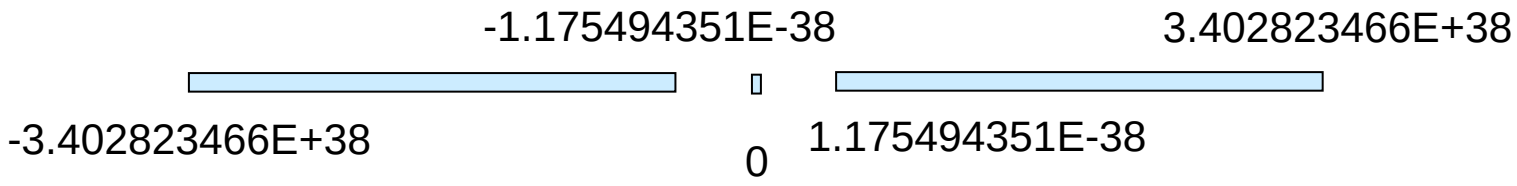
(*) : **INTEGER** est un synonyme de **INT**.

UNSIGNED permet d'avoir un type non signé.

ZEROFILL : remplissage des zéros non significatifs.

Types des attributs (III) – flottants

- **Les flottants – dits aussi nombres réels – sont des nombres à virgule. Contrairement aux entiers, leur domaine n'est pas continu du fait de l'impossibilité de les représenter avec une précision absolue.**
- **Exemple du type FLOAT :**



nom	domaine négatif : borne inférieure borne supérieure	Domaine positif : borne inférieure borne supérieure
FLOAT	-3.402823466E+38 -1.175494351E-38	1.175494351E-38 3.402823466E+38
DOUBLE*	-1.7976931348623157E+308 -2.2250738585072014E-308	2.2250738585072014E-308 1.7976931348623157E+308

(*) : **REAL** est un synonyme de **DOUBLE**.

Types des attributs (IV) – chaînes

nom	longueur
CHAR(M)	Chaîne de taille fixée à M, où 1<M<255, complétée avec des espaces si nécessaire.
CHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
VARCHAR(M)	Chaîne de taille variable, de taille maximum M, où 1<M<255, complété avec des espaces si nécessaire.
VARCHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
TINYTEXT	Longueur maximale de 255 caractères.
TEXT	Longueur maximale de 65535 caractères.
MEDIUMTEXT	Longueur maximale de 16777215 caractères.
LONGTEXT	Longueur maximale de 4294967295 caractères.
DECIMAL(M,D)*	Simule un nombre flottant de D chiffres après la virgule et de M chiffres au maximum. Chaque chiffre ainsi que la virgule et le signe moins (pas le plus) occupe un caractère.

(*) : **NUMERIC** est un synonyme de **DECIMAL**.

Types des attributs (V) – chaînes

- **Les types TINYTEXT, TEXT, MEDIUMTEXT et LONGTEXT peuvent être judicieusement remplacés respectivement par TINYBLOB, BLOB, MEDIUMBLOB et LONGBLOB.**

- **Ils ne diffèrent que par la sensibilité à la casse qui caractérise la famille des BLOB. Alors que la famille des TEXT sont insensibles à la casse lors des tris et recherches.**

- **Les BLOB peuvent être utilisés pour stocker des données binaires.**

- **Les VARCHAR, TEXT et BLOB sont de taille variable. Alors que les CHAR et DECIMAL sont de taille fixe.**

Types des attributs (VI) – dates et heures

nom	description
DATE	Date au format anglophone AAAA-MM-JJ.
DATETIME	Date et heure au format anglophone AAAA-MM-JJ HH:MM:SS.
TIMESTAMP	Affiche la date et l'heure sans séparateur : AAAAMMJJHHMMSS.
TIMESTAMP(M)	Idem mais M vaut un entier pair entre 2 et 14. Affiche les M premiers caractères de **TIMESTAMP**.
TIME	Heure au format HH:MM:SS.
YEAR	Année au format AAAA.

En cas d'insertion d'un enregistrement en laissant vide un attribut de type TIMESTAMP, celui-ci prendra automatiquement la date et heure de l'insertion. Contrairement à Unix (où le timestamp est le nombre de secondes écoulées depuis le 1er janvier 1970), en MySQL, il est une chaîne de format comme indiqué ci-contre.

nom	description
TIMESTAMP(2)	AA
TIMESTAMP(4)	AAMM
TIMESTAMP(6)	AAMMJJ
TIMESTAMP(8)	AAAAMMJJ
TIMESTAMP(10)	AAMMJJHHMM
TIMESTAMP(12)	AAMMJJHHMMSS
TIMESTAMP(14)	AAAAMMJJHHMMSS

Types des attributs (VII) – dates et heures

nom	description
DATE	Date au format anglophone AAAA-MM-JJ.
DATETIME	Date et heure au format anglophone AAAA-MM-JJ HH:MM:SS.
TIMESTAMP	Affiche la date et l'heure sans séparateur : AAAAMMJJHHMMSS.
TIMESTAMP(M)	Idem mais M vaut un entier pair entre 2 et 14. Affiche les M premiers caractères de **TIMESTAMP**.
TIME	Heure au format HH:MM:SS.
YEAR	Année au format AAAA.

Le format de date AAAA-MM-JJ signifie : année sur 4 chiffre, mois sur 2 chiffres et jour sur 2 chiffres avec pour séparateur le tiret. Le format d'heure HH:MM:SS signifie : heure, minute et seconde chacune sur 2 chiffres, avec pour séparateur les deux points.

Dans le format étendu qui comprend la date et l'heure, des deux dernières sont séparées par un espace. Les formats de date sont assez permissifs car des variantes sont tolérées. Il est possible de mettre n'importe quel caractère qui ne soit pas un chiffre en guise de séparateur, il est aussi possible de na pas en mettre, comme cela est affiché par TIMESTAMP.

Types des attributs (VIII) – énumérations

Un attribut de type **ENUM** peut prendre une valeur parmi celles définies lors de la création de la table plus la chaîne vide ainsi que **NULL** si la définition le permet. Ces valeurs sont exclusivement des chaînes de caractères. Une énumération peut contenir 65535 éléments au maximum.

Définition d'un tel attribut :

nom_attribut ENUM(‘’valeur 1’’,’’valeur 2’’…)

nom_attribut ENUM(‘’valeur 1’’,’’valeur 2’’…) NULL

A chaque valeur est associée un index allant de 0 à N si N valeurs ont été définies. L'index 0 est associé à la chaîne nulle, l'index 1 à la première valeur… L'index **NULL** est associé à la valeur **NULL**.

Si une sélection (**SELECT** ou **WHERE**) est faite dans un contexte numérique, l'index est renvoyé. Sinon, c'est la valeur qui est retournée.

Il peut être défini jusqu'à 65535 valeurs distinctes insensibles à la casse.

Types des attributs (IX) – ensembles

Un attribut de type **SET** peut prendre pour valeur la chaîne vide, **NULL** ou une chaîne contenant une liste de valeurs qui doivent être déclarées lors de la définition de l'attribut lors de la création de la relation.

Par exemple, un attribut déclaré comme ci :

SET(‘‘voiture’’, ‘’moto’’, ’’vélo’’) NOT NULL

peut prendre les valeurs suivantes :

‘’’ (chaîne vide)

‘’voiture, moto’’

‘’vélo, voiture, moto’’

et autres combinaisons de listes des trois valeurs définie plus haut.

Un attribut déclaré comme suit :

SET(‘‘voiture’’, ‘’moto’’, ’’vélo’’) NULL

peut prendre, en plus ce celles précédentes, la valeur **NULL**.

Il ne peut être défini que 64 éléments maximum.

Langage de Description de Données (LDD)

LDD, verbes

Le langage de description de données (LDD) est constitué de trois verbes :

CREATE DROP ALTER

Ce langage permet de décrire les objets qui contiendront les données. Il ne sert pas à

manipuler ces données.

En général il n'est utilisé que lors de la création du schéma de la base de données au moment de l'installation du logiciel ou bien lors de la modification de ce schéma si des mises-à-jour sont effectuées.

LDD, create table

Le verbe create table est utilisé pour créer une table avec l'ensemble de ses colonnes.

Syntaxe :

create table Nom*table*
(
    colonne1 type [not null],
    colonne2 type [not null],
     ...,
    primary key (*colonnes*)
    [, unique (*colonnes*)]
    , ...
);

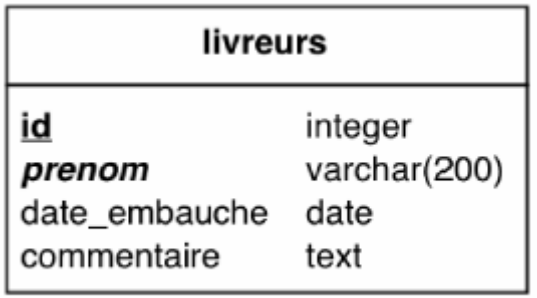
Les colonnes qui constituent la clé primaire doivent toutes être not null.

LDD, create table : Exemple

```
create table livreurs
(
id integer not null,
prenom varchar(200) not null,
date_embauche date,
commentaire text,
primary key (id),
 unique (prenom)
);
```

Ou bien :

```
create table livreurs
(
 id integer   primary key,
 prenom varchar(200) not null unique,
 date_embauche date,
 commentaire text
 );
```



LDD, alter table

La commande alter table permet de modifier certaines caractéristiques d'une table ou de l'une de ses colonnes.

Exemple :il nous est maintenant nécessaire d'ajouter la colonne Tel à la table livreurs:

```
alter table livreurs add column Tel varchar(10) ;
```

LDD, alter table

La commande alter table permet également :

- d'ajouter une clé unique

  alter table Nom*table* add unique (*colonnes*)

- supprimer une colonne

  alter table Nom*table* drop column *colonne*

- changer le nom et le type d'une colonne

  alter table Nom*table* change column *colonne nouveau_nom nouveau_type*

- renommer une table

  alter table Nom*table* rename as *nouveau_nom*

La commande drop table supprime une table :

drop table Nomtable

Langage de Manipulation de Données (LMD)

LMD, verbes

Le langage de manipulation des données (LMD) est constitué de quatre verbes :

SELECT INSERT UPDATE DELETE

Chacun de ces verbes possède une ou plusieurs clauses.

Lorsqu'elles sont entrées dans l'utilitaire mysql, les commandes SQL doivent se terminer par un point-virgule. Dans d'autres interfaces le point-virgule final est généralement accepté mais pas souvent nécessaire.

Par exemple :

select * from pizzas where nom='Roi';

La clause from permet de spécifier le nom de la table qui est interrogée, la clause where permet de préciser un critère de sélection.

Les chaînes de caractères sont entourées par des apostrophes simples. Si une apostrophe simple est présente dans la chaîne, elle doit être doublée. Par exemple :

'Il faut s''asseoir pour réfléchir !'

LMD, select

Le verbe SELECT permet de faire des requêtes sur une ou plusieurs tables. Il ne modifie jamais les données.

Une forme simple de select est :

select *colonnes* from *tables* where *condition* order by *colonnes*[asc/desc]

Par exemple, la requête suivante affiche le nom et le prix des pizzas pour deux personnes. Le résultat est trié par prix :

select nom, prix from pizzas where taille=2 order by prix;

Pour sélectionner toutes les colonnes d'une table, on utilise le caractère « * ».

Par exemple :

select * from ingredients order by nom desc;

Il est également possible de réaliser des calculs.

Combien coûtent trois pizzas reines ?

select prix*3 from pizzas where nom='Reine';

LMD, select

Les opérateurs de comparaison sont les suivants :

- **arithmétiques : = < > <= >= !=**
- **plage de valeur : between *v1* and *v2***
- **appartenance : in (*v1*, *v2*, etc.)**
- **nullité : is null, is not null**
- **comparaison de chaîne : like '*modèle*' (caractères joker : « % » et « _ »)**

Exemples :

```
select * from commandes where quantite >= 2;
select nom, prix from pizzas where prix between 5 and 10;
select * from ingredients where id in (1, 2, 4);
select nom, note from clients where note is not null;
select telephone, nom from clients where telephone like '01%' order by nom;
```

Opérateur booléens : and et or.

Exemple :

```
select * from pizzas where taille = 4 and prix < 10;
```

LMD, select

Le verbe select dispose d'une clause qui permet de limiter le nombres d'enregistrements retournés : limit.

Elle s'utilise ainsi :

select *colonnes* from *table* where *condition* limit [*a*,] *b*

Où *a* est l'index (à partir de 0) de la première ligne à afficher (0 si non précisé) et *b* est le nombre maximal de lignes. Si *b* est « -1 », toutes les lignes restantes sont retournées.

Par exemple, pour lister les lignes de la table commandes trois par trois, on utiliserait

successivement :

select * from commandes limit 3;

select * from commandes limit 3, 3;

select * from commandes limit 6, 3;

etc...

Cette clause est particulièrement utile lorsque MySQL est utilisé pour afficher le contenu d'une table qui possède beaucoup de lignes (par exemple via une application Web).

LMD, select

Les fonctions de groupes permettent de répondre à des questions du type « quelle est le nombre de clients de la pizzéria ? ». Une telle question s'écrirait ainsi :

select count(*) from clients;

Les fonctions de groupes sont : count sum max min avg.

Elles sont également appelées « fonctions agrégeantes » lorsqu'elles sont utilisées avec la clause group by.

Exemple :

select telephone_client, sum(prix_total)

from commandes group by telephone_client;

Il peut être pratique d'utiliser les alias de colonnes. Par exemple, pour établir le palmarès des trois pizzas les plus vendues :

select pizza_id as pizza_no, sum(quantite) as total

from commandes group by pizza_id order by total desc ;

LMD, jointures

Pour obtenir la liste et la quantité des pizzas commandées, il est nécessaire d'extraire des données provenant des tables commandes (quantités commandées) et pizzas (nom des pizzas). L'identifiant de la pizza (donnée présente dans les deux tables) formera le pivot

de cette requête. Cette technique est appelée jointure.

```
select commandes.id, quantite, nom
from commandes, pizzas
where commandes.pizza_id = pizzas.id
order by 1;
```

LMD, jointures

Il existe deux formes pour écriture les jointures :

1) select commandes.id, commandes.quantite, pizzas.nom

from commandes, pizzas

where commandes.pizza_id = pizzas.id order by 1;

2) select commandes.id, commandes.quantite, pizzas.nom

from commandes inner join pizzas on commandes.pizza_id = pizzas.id order by 1;

Ces deux requêtes fournissent le même résultat.

LMD, jointures

Une jointure peut se faire avec plus de deux tables.

Le nom de chaque colonne doit être préfixé par le nom de la table dont il est issu car il y aurait si non ambiguïté. Il est également possible d'utiliser des alias pour les tables

LMD, insert

Le verbe insert permet d'ajouter des lignes à une table. Il s'utilise de deux manières :

insert into table (*colonnes*) values (*valeurs*)

insert into table (*colonnes*) select *colonnes* from *tables*

La liste des colonnes entre parenthèses est optionnelle. Si elle est omise la liste des valeurs doit fournir une valeur pour toute les colonnes de la table dans l'ordre dans lequel elle ont été spécifiées lors de la création de la table.

La seconde forme permet d'insérer dans la table de destination le résultat d'une requête.

Par exemple :

insert into ingredients (id, nom) values (8, 'Poivron');

est équivalent à :

insert into ingredients values (8, 'Poivron');

si les colonnes id et nom sont ordonnées de cette manière dans la table ingredients.

Pour le vérifier :

desc ingredients;

LMD, insert

Il n'est pas nécessaire de fournir une valeur pour les colonnes dites « nullable » (c'est-àdire créée avec l'option not null).

Il est possible d'effectuer une insertion à partir du résultat d'une requête tout en fournissant certaines valeurs « externes ». Par exemple, pour créer une commande de deux pizzas « Roi » pour le client Ali Brahimi :

```
insert into commandes
(telephone_client, pizza_id, date_commande, quantite, prix_unite, prix_total)
select
clients.telephone, pizzas.id, now(), 2, pizzas.prix, pizzas.prix*2
from pizzas, clients
where pizzas.nom = 'Roi'
and clients.nom = ‘Ali Brahimi ';
```

LMD, update

Le verbe update permet de modifier des lignes déjà présentes dans une table.

Sa syntaxe est la suivante :

update *table* set *colonne*=*valeur*, *colonne*=*valeur* where *condition*

La clause where est optionnelle. Si elle est omise, les modifications sont appliquées à la totalité des lignes de la table.

Par exemple, nous livrons la commande de d' **Ali Brahimi** à 20h28 :

update commandes set date_livraison = '2003-05-14 20:28' where id = 7;

Il est possible d'introduire une formule dans la requête de mis-à-jour.

Par exemple, nous décidons d'augmenter de 10% le prix de toutes nos pizzas pour deux personnes :

update pizzas set prix = prix*1.1 where taille = 2;

LMD, delete

Le verbe delete est utilisé pour supprimer des lignes d'une table.

Sa syntaxe est la suivante :

delete from *table* where *condition*

La clause where est optionelle. Toutes les lignes sont supprimées si elle est omise.

Exemple : delete from Clients where *Ville* = *'Oujda'*

Bases de données

Les commandes create database et drop database permettent de créer ou de supprimer une base de données.

Leur syntaxe est simple. Attention lors de l'utilisation de drop database !

create database *nombase*

drop database *nombase*

L'utilisation de ces commandes n'est autorisée que si les droits de l'utilisateur connecté le permettent.

Dans l'utilitaire mysql, pour changer de base de données, utiliser la commande use ainsi :

use *nombase*

Par exemple, un script de création d'une base de données commence souvent ainsi :

create database pizzas;

use pizza;

create table...

Langage de Définition des Droits d'Accès (LDDA)

Droits d'accès

Le système de gestion des droits d'accès de MySQL est basé sur cinq tables de la base de données mysql (créée lors de l'installation de MySQL) :

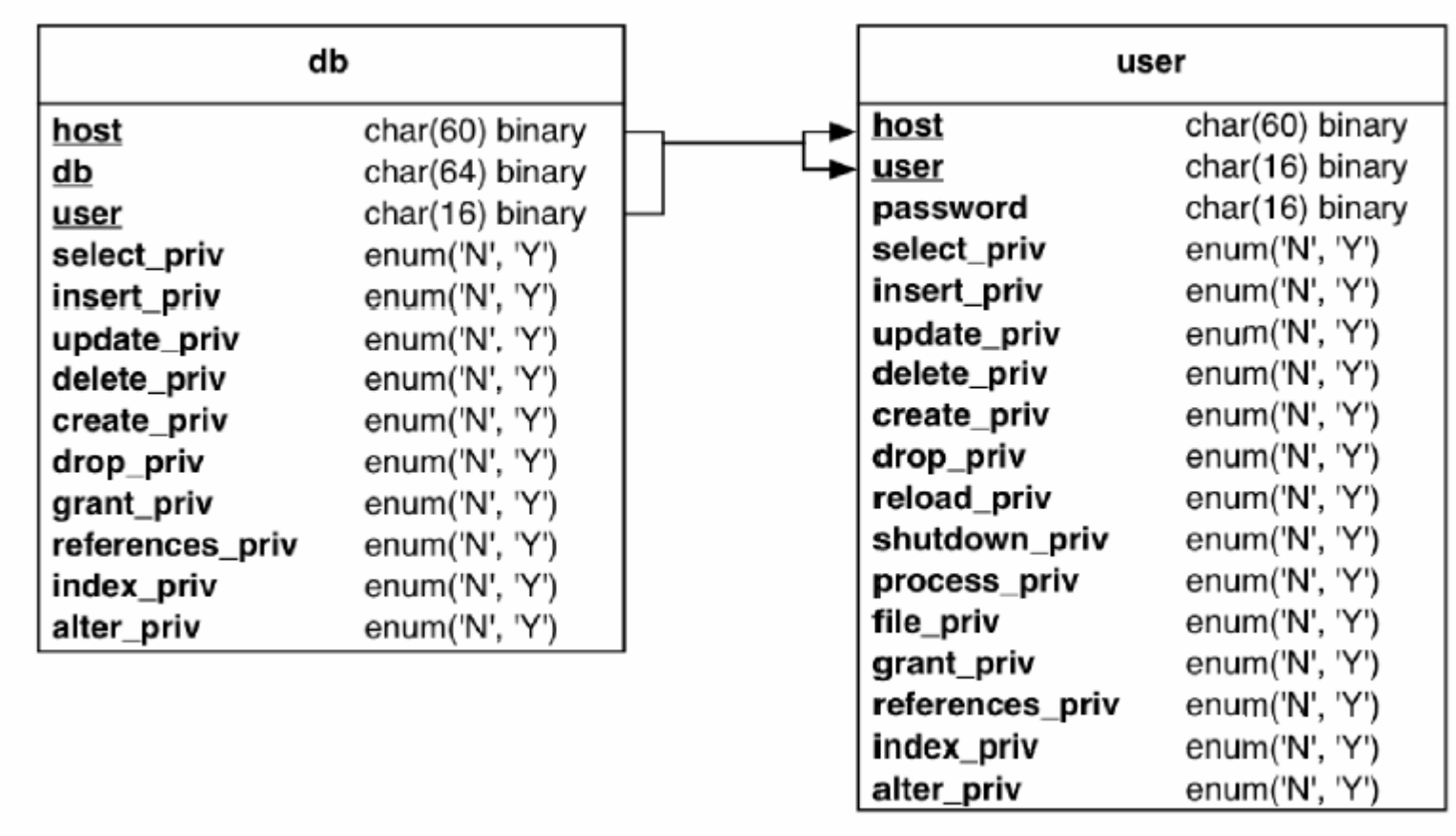
user db host tables_priv column_priv

Dans le cadre de ce support de cours, nous nous limiterons à l'étude des deux premières.

Un utilisateur est identifié par le couple : user, host. La syntaxe utilisée est « user@host ».

Droits d'accès

Des privilèges sont associés à chaque utilisateur soit au niveau global, dans la table user, soit au niveau d'une base de données, dans la table db.



Droits d'accès

La table user contient la liste des utilisateurs qui ont accès au serveur MySQL.

Leur mot de passe est stocké dans la colonne password.

Cette table contient également deux types de privilèges :

- les privilèges associés au serveur MySQL

create(database) drop(database) reload shutdown process file

- les privilèges associés aux tables de toutes les bases de données

select insert update delete create(table,index) drop(table) grant references index alter

Ainsi, un utilisateur qui possède le privilège create dans la table user est autorisé à créer des bases de données et des tables dans toutes les bases de données.

Il lui est également permis de créer des index dans toutes les bases de données mais uniquement lors de la création d'une table avec la commande create table (requis pour la création des clés primaires et uniques).

En général, il existe un seul utilisateur MySQL qui possède tous les droits au niveau global. Il est considéré comme l'administrateur du serveur MySQL. Bien souvent, il s'agit de « root@localhost ».

Droits d'accès, grant

La commande grant permet d'allouer des privilèges à un utilisateur. Celui-ci est créé s'il n'existe pas dans la table user. Sa syntaxe est :

grant *privileges* on *objets* to *user@host* [identified by '*pass*'] [with grant option]

Les clauses identified by et with grant option sont optionnelles. La première permet de préciser un mot de passe pour l'utilisateur, la seconde l'autorise à donner ses privilèges à un autre utilisateur.

La liste *privileges* peut être remplacée par le mot « all » pour signifier tous les

privilèges. Les privilèges sont séparés par des virgules et proviennent de cette liste :

select insert update delete create drop references index alter

à laquelle est ajoutée celle-ci pour les privilèges au niveau global (on *.*) :

reload shutdown process file

La liste *objets* peut prendre l'une de ces valeurs :

- « *.* » donne des privilèges au niveau global (table user) ;

- « *database*.* » affecte toutes les tables d'une base de données (table db) ;

- « *database*.*table* » affecte une table particulière (table tables_priv).

Droits d'accès, grant

Par exemple, un script de création d'une base de données commence souvent ainsi :

create database pizzas;

grant select, insert, update, delete on pizzas.*

to pizzeria@localhost identified by 'italia';

use pizzas;

create table...

Ce script est lancé avec l'utilisateur d'administration du serveur MySQL :

mysql -u root -p mysql < cr_pizzas.sql

Ceci permet de créer un utilisateur spécifique avec des droits qui lui permettent uniquement d'utiliser le LMD afin de manipuler les données dans les tables que l'administrateur aura créées pour lui.

Droits d'accès, revoke, set password

La commande revoke permet de révoquer les droits d'un utilisateur.

Sa syntaxe est :

revoke *privileges* on *objets* from *user@host*

La liste des privilèges et des objets sont utilisées de la même manière qu'avec la commande grant.

La commande set password permet de modifier le mot de passe d'un utilisateur.

Sa syntaxe est :

set password for *user@host* = '*password*'

Elle est souvent utilisée avec la fonction password qui permet de crypter un mot de passe en clair vers le format utilisé par MySQL.

Par exemple :

set password for pizzeria@localhost = password('PastaFantastica');

Droits d'accès, show grants

Pour en terminer avec la gestion des droits d'accès, l'une des options de la commande

show permet d'obtenir une liste synthétique, mais exhaustive, des privilèges qui ont été octroyés à un utilisateur.

Sa syntaxe est :

show grants for user@host

Par exemple :

mysql> show grants for pizzeria@localhost;

Administration avec l’outil web phpMyAdmin

Présentation

L'outil phpMyAdmin est développé en PHP et offre une interface intuitive pour l'administration des base de données du serveur.

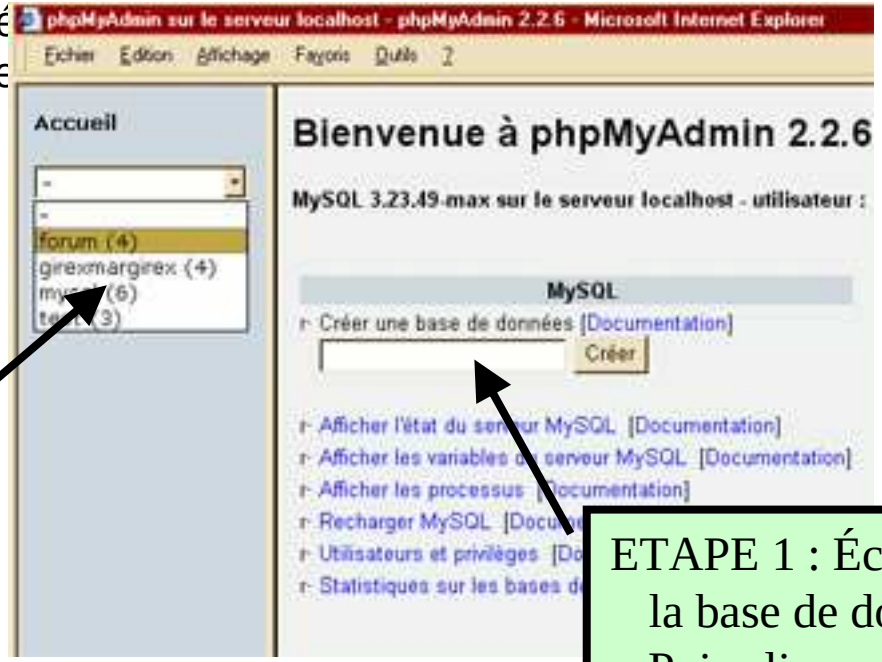
Il est téléchargeable ici : http://phpmyadmin.sourceforge.net

Cet outil permet de :

1. créer de nouvelles bases
2. créer/modifier/supprimer des tables
3. afficher/ajouter/modifier/supprimer des tupes dans des tables
4. effectuer des sauvegarde de la structure et/ou des données
5. effectuer n'importe quelle requête
6. gérer les privilèges des utilisateurs

Création/sélection d'une base de données

Avant de manipuler des donné
créer une ou des bases de

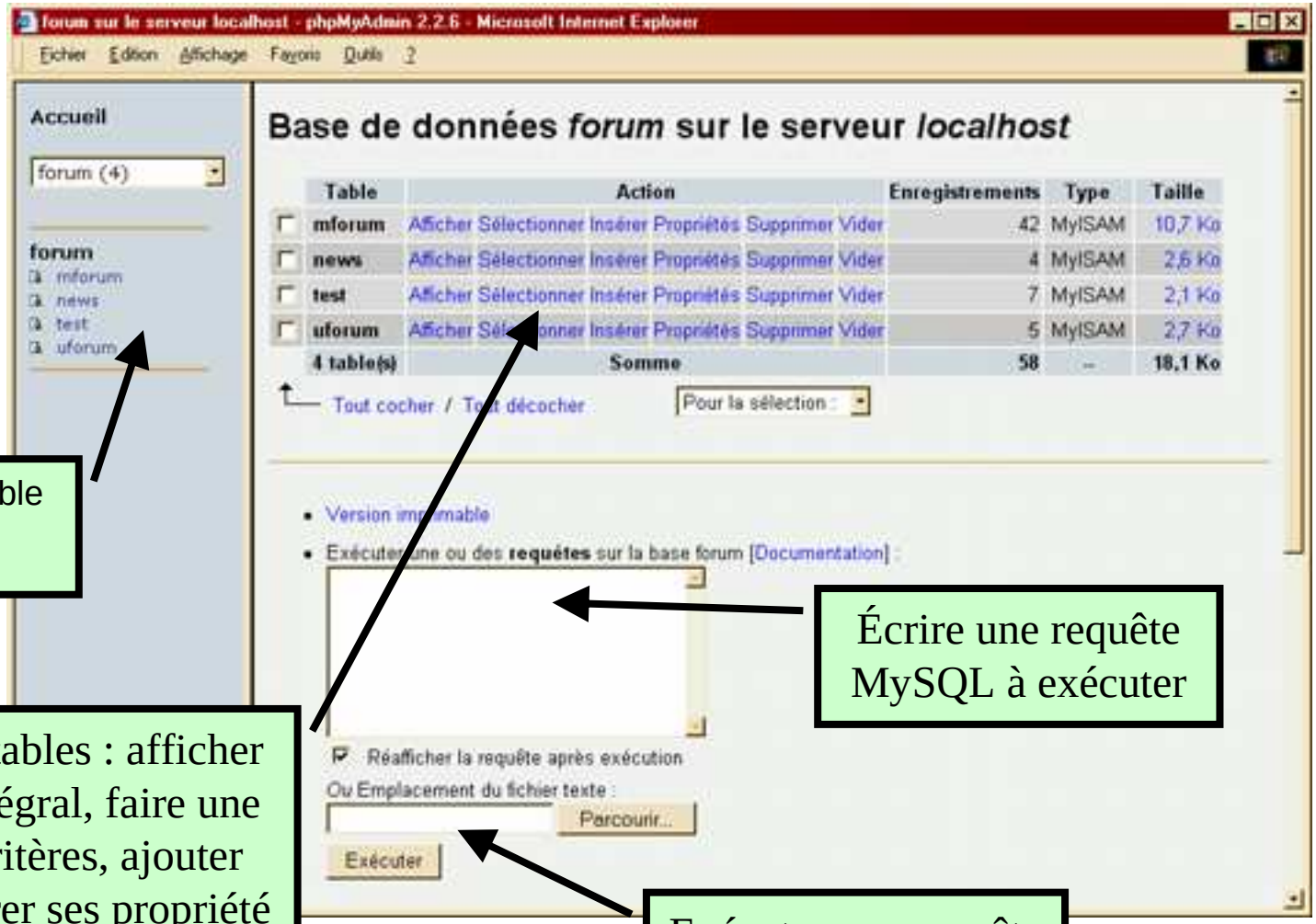


ETAPE 2 : sélectionnez le nom de la base à manipuler (le nombre de tables de la base apparaît entre parenthèses)

ETAPE 1 : Écrivez le nom de la base de donnée à créer. Puis cliquez sur « Créer »

Et aussi.. choix de la langue de l'interface de phpMyAdmin

Gestion de la base de données (I)



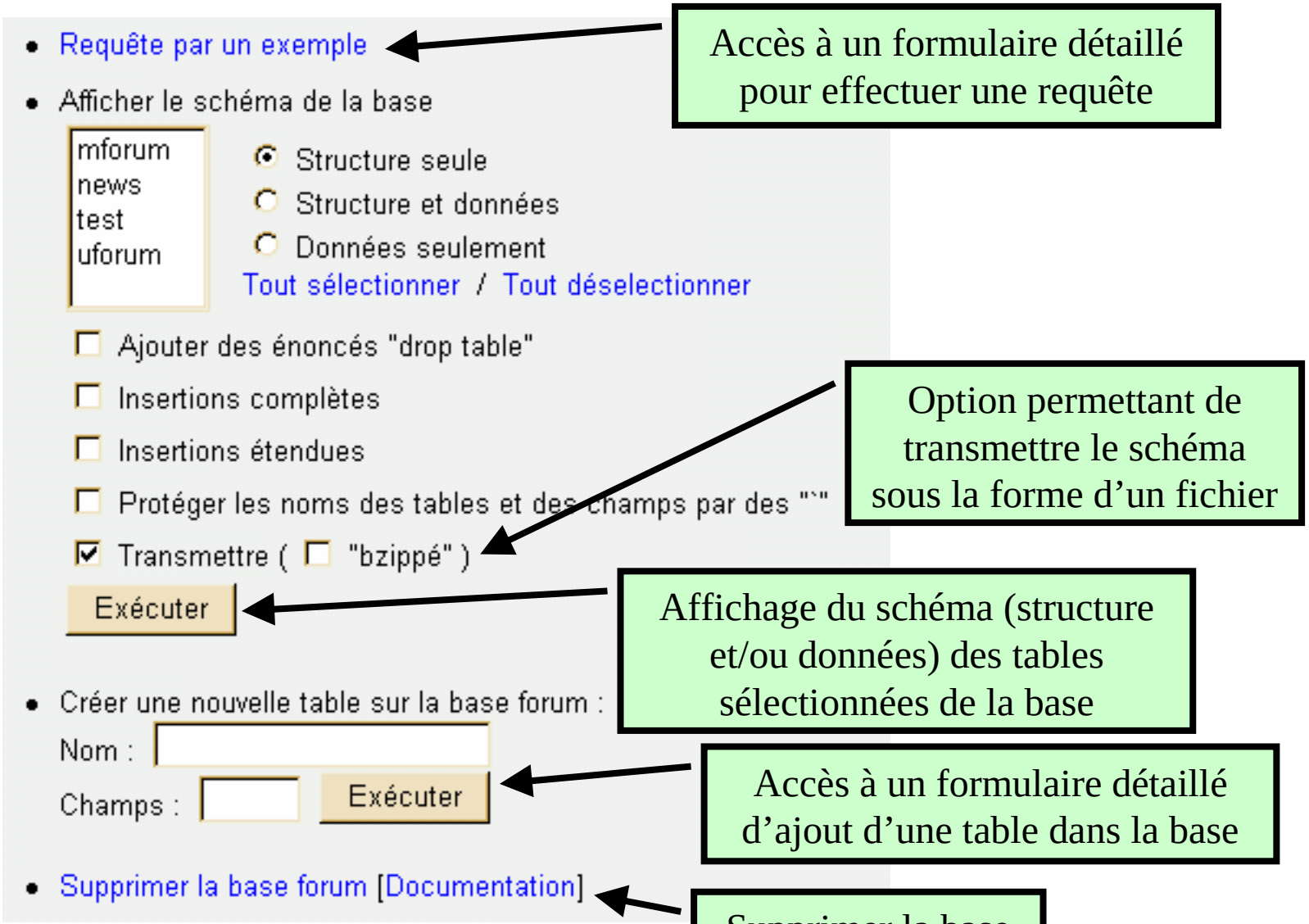
- Choix d'une table à gérer en particulier

Actions sur les tables : afficher leur contenu intégral, faire une sélection sur critères, ajouter des données, gérer ses propriété intrinsèques, supprimer, vider.

Écrire une requête MySQL à exécuter

Exécuter une requête MySQL contenue dans un fichier

Gestion de la base de données (II)



Affichage d'une table

Rappel de la base, de la table et du serveur

Rappel de la requête

Colonnes = noms des attributs de la table

Liste des enregistrements de la table par pages de X lignes

Supprimer un enregistrement

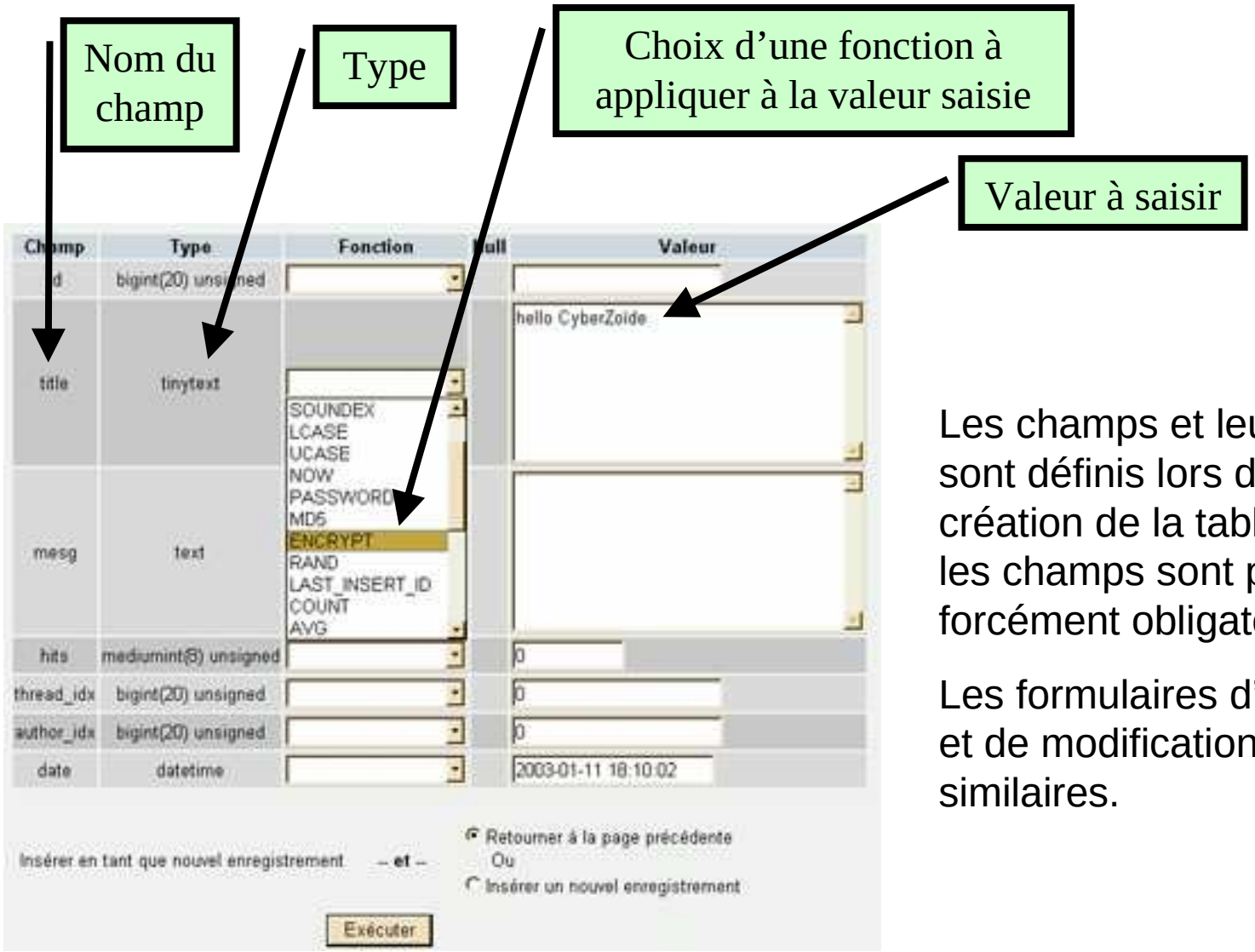
Accès au formulaire de modification d'un enregistrement

Permet de naviguer dans les pages de résultats

Insertion d'un nouvel enregistrement

Afficher par page de X lignes

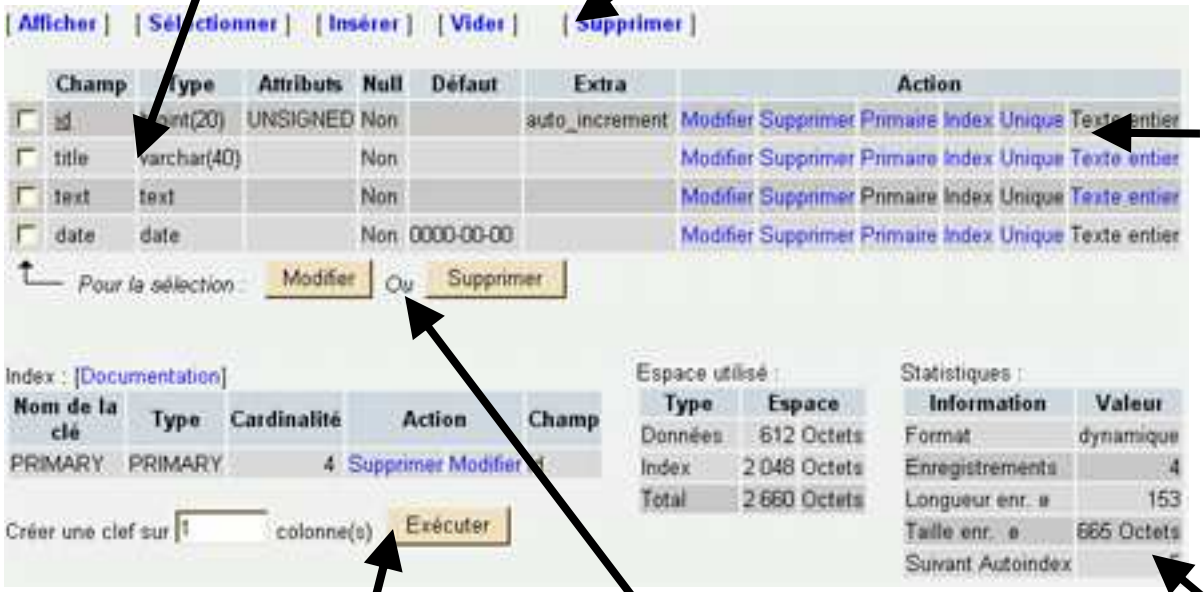
Insertion/modification d'un enregistrement



Les champs et leurs types sont définis lors de la création de la table : tous les champs sont pas forcément obligatoires…

Les formulaires d'insertion et de modification sont similaires.

Gestion d'une table (I)



Propriétés des attributs de la table

Quelques actions rapides : affichage, sélection, insertion d'un enregistrement, vidage, suppression

Quelques actions sur les attributs : modifier, supprimer ; plus les contraintes : clé primaire et unique ; et y mettre un index

Créer une nouvelle clé sur X attributs

Modifier ou supprimer plusieurs attributs en même temps

Quelques statistiques et infos

Gestion d'une table (II)

Affichage de la version imprimable de la page

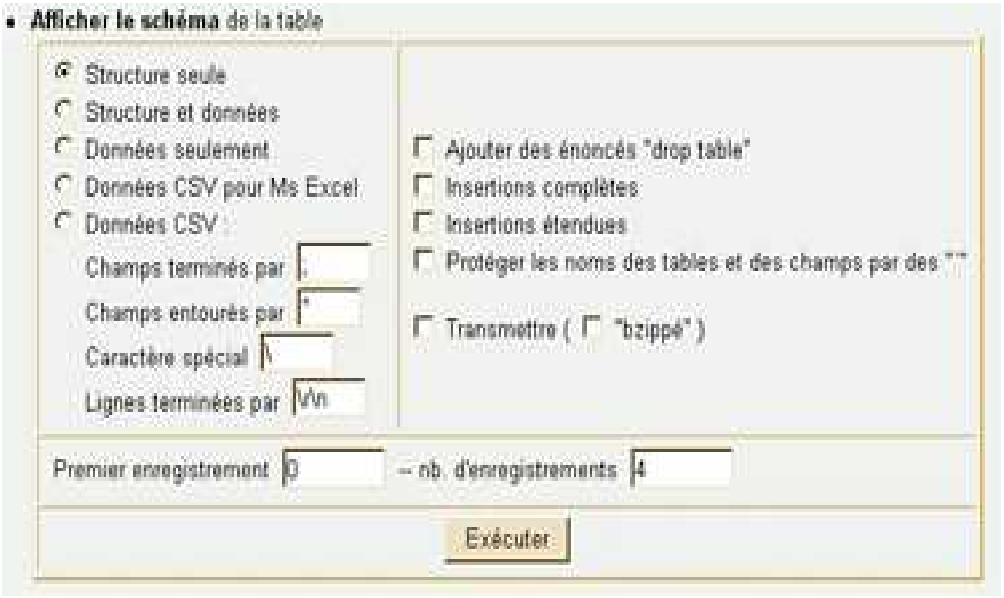
Écrire une requête à exécuter ou spécifier un fichier en contenant une ou plusieurs

Ajouter X champs dans la table à une position particulière

Réordonner les données de table en fonction d'un attribut

Accès au formulaire d'insertion de données dans la table à partir d'un fichier

Gestion d’une table (III)



Permet d’afficher le schéma (structure et/ou données) de la table.

Le schéma d’une table est l’ensemble de la structure et des données d’une table.

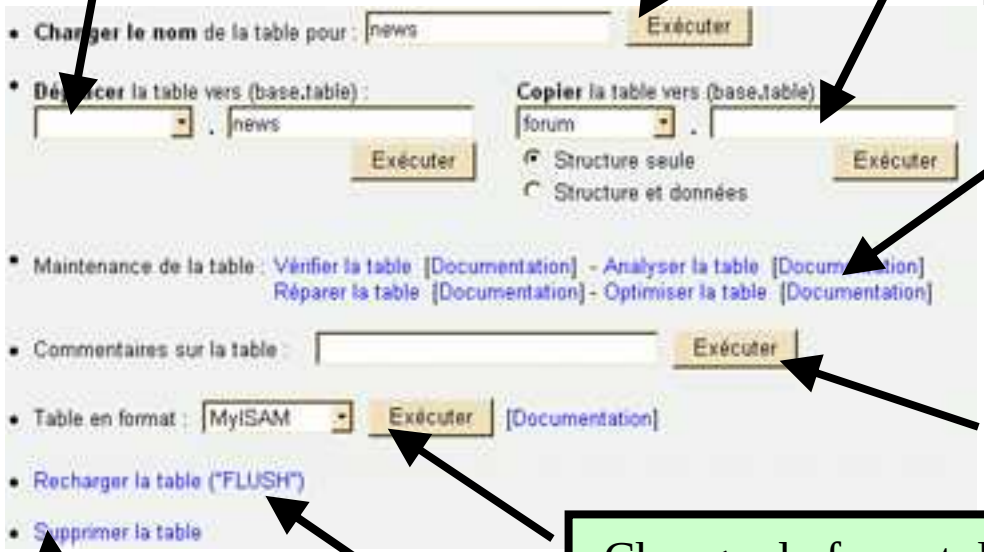
La structure est composée de la définition des propriétés des attributs et des clés.

Le résultat de sortie « structure et/ou données » est constitué des requêtes MySQL de création de la table et d’insertion des enregistrements.

Le format CSV est un fichier texte dont chaque ligne représente un enregistrement.

Le résultat peut être transmis sous la forme d’un fichier (qui lui-même peut être compressé).

Gestion d'une table (IV)



Déplacer la table vers une autre base avec changement de nom possible dans la foulée

Changer le nom de la table

La copier (avec ou sans les données) vers une autre base avec changement de nom possible

Quelques opérations de maintenance : vérification, analyse, réparation, optimisation

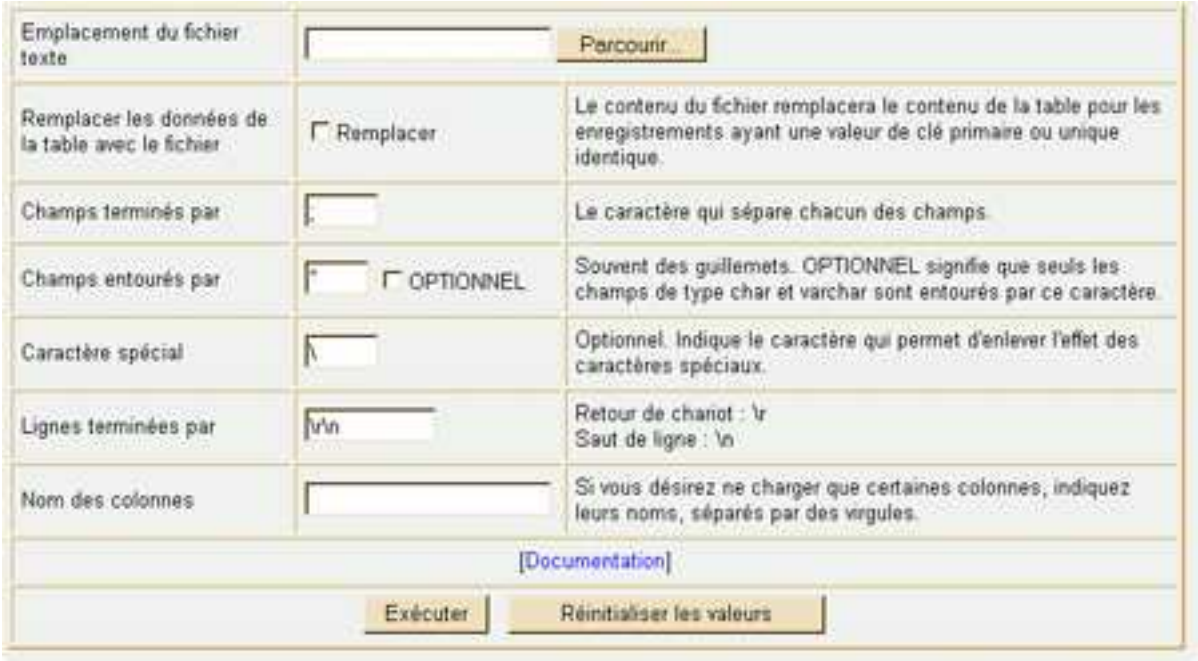
Lui associer un commentaire

Changer le format de la table

Recharger la table en mémoire du serveur

La supprimer

Insertion des données dans une table



On peut changer les valeurs par défaut des séparateurs standards du CSV.

On accède à cet écran var le lien « *Insérer des données provenant d'un fichier texte dans la table* » de la page de gestion de la table.

Permet d'insérer des enregistrements dans une table à partir d'un fichier de données au format CSV.

Création d’une clé

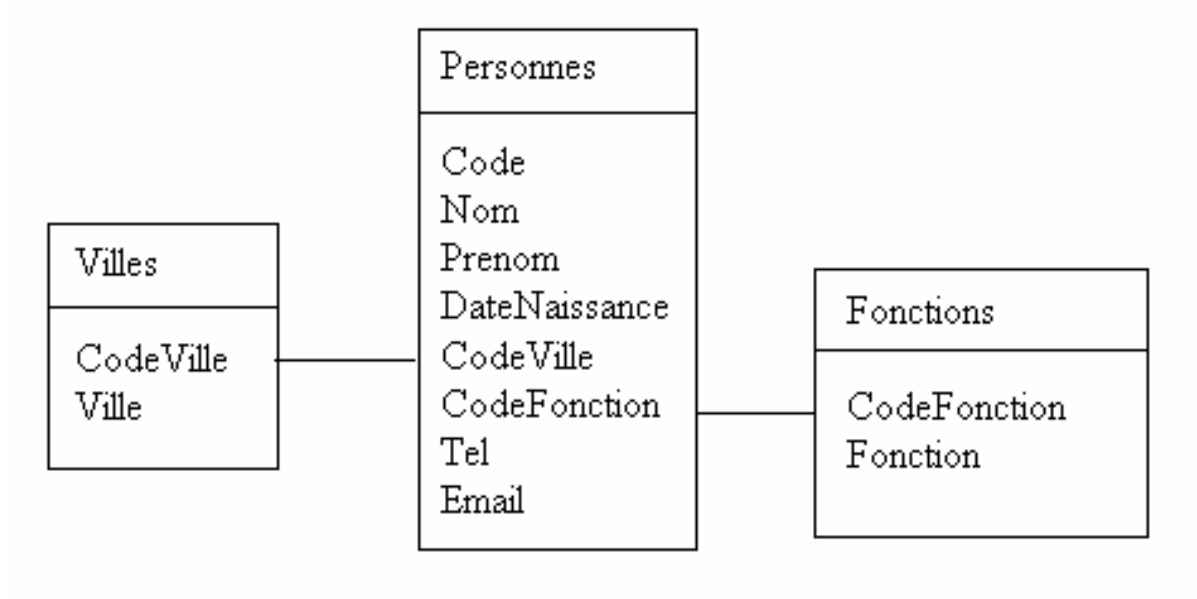
On accède à cet écran var le lien « *Créer une clef sur X colonne(s* » de la page de gestion de la table.

Permet de créer une clé sur une ou plusieurs colonnes.

Il faut nommer la clé, en spécifier le type, et les attributs sur lesquels elle s’applique.

On peut rajouter une autre colonne à cette clé avant de valider l’ajout de la clé.

Base de données Agenda



Création et Sélection d'une BD

- Création de la Base de Données

```
create database Agenda;
```

- Sélection de la Base de Données

```
use Agenda;
```

Création de table

- Création de la table Villes:

```
create table Villes
(
CodeVille integer not null primary key,
Ville varchar(30) NOT NULL
);
```

Création de table

- Création de la table Fonctions:

```
create table Fonctions
(
CodeFonction integer not null primary key,
Fonction varchar(30) NOT NULL
);
```

Création de table

- Création de la table Personnes:

```
create table Personnes
(
Code integer not null primary key,
Nom varchar(30) NOT NULL,
Prenom varchar(30),
DateNaissance date,
CodeVille integer,
CodeFonction integer,
Tel varchar(10),
Email varchar(100)
);
```

Insertion de données

- Insertion des données dans la table Villes

```
INSERT INTO Villes VALUES (1,'Oujda');
INSERT INTO Villes VALUES (2, 'Casa');
INSERT INTO Villes VALUES (3, 'Rabat');
INSERT INTO Villes VALUES (4, 'Tanger');
```

Insertion de données

- Insertion des données dans la table Fonctions

INSERT INTO Fonctions VALUES (1,‘Directeur');
INSERT INTO Fonctions VALUES (2, ‘Ingénieur');
INSERT INTO Fonctions VALUES (3, ‘Secrétaire');
INSERT INTO Fonctions VALUES (4, ‘Médecin');

MAJ des tables

- **Ajout de colonnes**

```
ALTER TABLE Personnes ADD Fax varchar(10);
```

- **Ajout de contraintes**

```
ALTER TABLE Personnes ADD CONSTRAINT fkVille
FOREIGN KEY (CodeVille) REFERENCES Villes (CodeVille);

ALTER TABLE Personnes ADD CONSTRAINT fkFonction
FOREIGN KEY (CodeFonction) REFERENCES Fonctions
    (CodeFonction);
```

Insertion de données

- **Insertion des données dans la table Personnes**

```
INSERT INTO Personnes VALUES (1,'Alami','Farid','1980-03-09',1,3,'0536685645','af@yahoo.fr',NULL);

INSERT INTO Personnes VALUES (2, 'Daoudi','Nadia','1970-08-30',2,4,'0666233454','dn@gmail.com',NULL);

INSERT INTO Personnes VALUES (3, 'Imame','Mohamed','1969-09-12',4,2,'0661232112','im@hotmail.com',NULL);

INSERT INTO Personnes VALUES (4, 'Malki','Ahlam','1989-06-15',1,1,'0536705060','ma@menara.ma',NULL);
```

MAJ des données

- **Modification des données**

```
UPDATE Personnes SET Fax='0536705645'  WHERE Code = 1;
UPDATE Personnes SET Fax='0536705646'  WHERE Code = 2;
UPDATE Personnes SET Fax='0536705647'  WHERE Code = 3;
UPDATE Personnes SET Fax='0536705648'  WHERE Code = 4;
```

Requêtes monotables

```
SELECT Nom, Prenom FROM Personnes WHERE Code = 3;

SELECT Code, Nom, Prenom FROM Personnes
WHERE Prenom='Mohamed'AND CodeVille= 4;

Select * from Personnes where DateNaissance > '1980-01-01';
```

Fonctions et groupements

Select count(*) from Personnes;

Select Min(DateNaissance) from Personnes;

Select CodeVille, count(*) from Personnes group by CodeVille;

Requêtes multitables

Select nom, prenom, ville from personnes, villes
Where personnes.codeville = villes.codeville;

Select nom, prenom, fonction from personnes, fonctions Where personnes.codefonction = fonctions.codefonction;

Suppression de tables

Drop table Villes;

Drop table Fonctions;

Drop table Personnes;

Rem: il faut respecter la priorité de suppression.

Suppression de la BD

Drop database Agenda;

Autres…

- Help
- Exit
- …….