

SQL ET MYSQL

Michel Blache
Septembre 2019



Sommaire

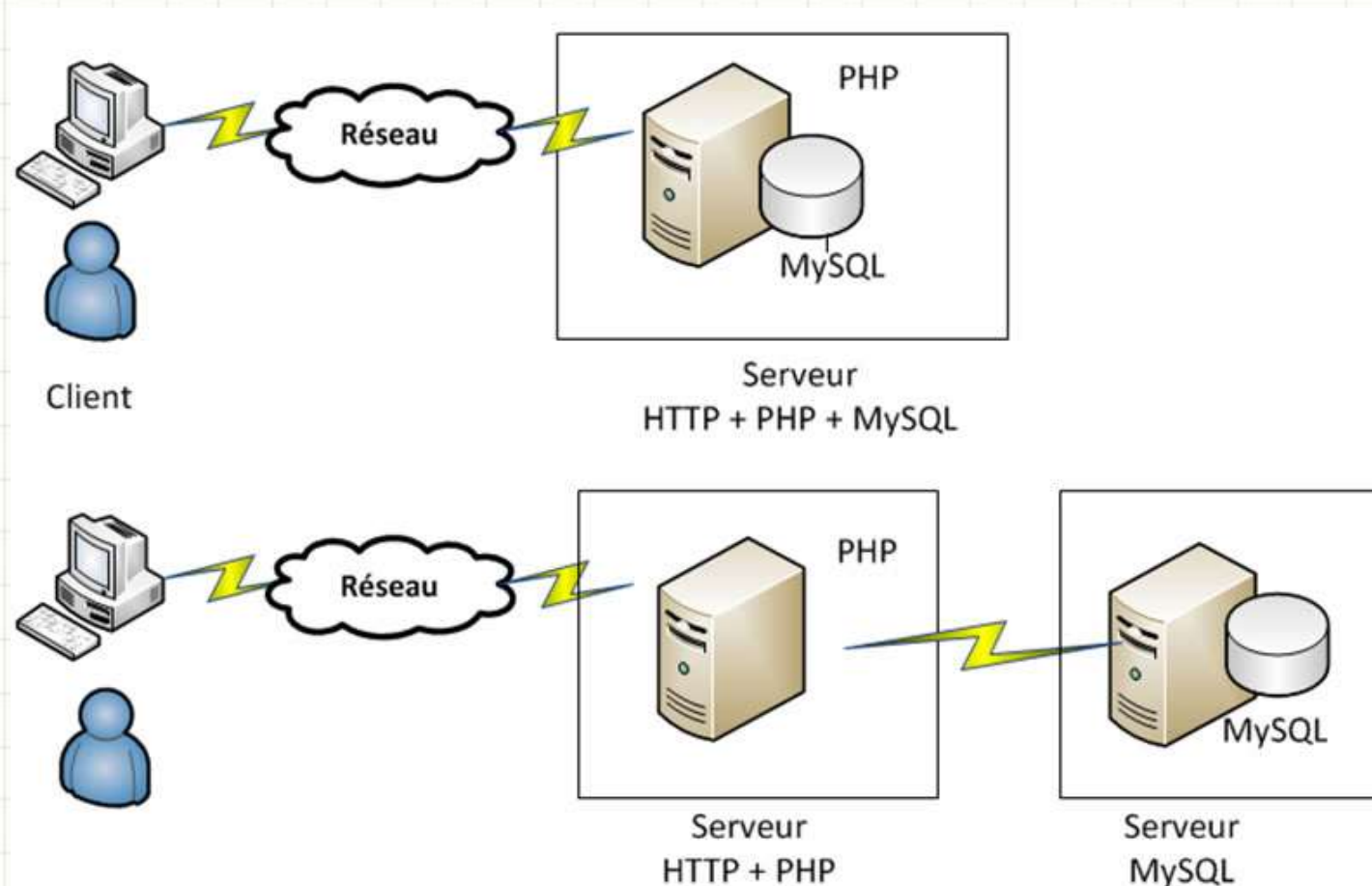
• Sommaire	2
• SQL & MySql	3
• Architecture	4
• Objets	5
• Clé primaire et étrangère	7
• Clé unique	9
• Interface à MySql	10
• PhpMyAdmin	11
• Base de données Pizza	12
• LMD	17
• LDD	30
• Droits d'accès	38
• Fonctions	46
• Index	49
• Colonnes	53

SQL & MySQL

- Le langage SQL (Structured Query Language) est un langage de requête utilisé pour interroger des bases de données exploitant le modèle relationnel. Il fait l'objet d'une norme ANSI.
- Les points forts de MySQL sont :
 - son implémentation libre et populaire ;
 - sa facilité de mise en œuvre ;
 - son appréhension facilité;
 - son support multi-plateforme ;
 - sa fiabilité et sa rapidité.

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.



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).

Objets

- 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).

serveur MySQL

base de données 1

table 1		
colonne 1	colonne 2	etc...
Ligne 1	donnée	etc...
Ligne 2	donnée	etc...
Ligne 3	donnée	etc...
Ligne 4	donnée	etc...
Ligne 5	donnée	etc...
etc...	etc...	etc...

table 2

colonne 1	colonne 2
Ligne 1	donnée
Ligne 2	donnée
Ligne 3	donnée
Ligne 4	donnée
Ligne 5	donnée
etc...	etc...

base de données 2

table 1

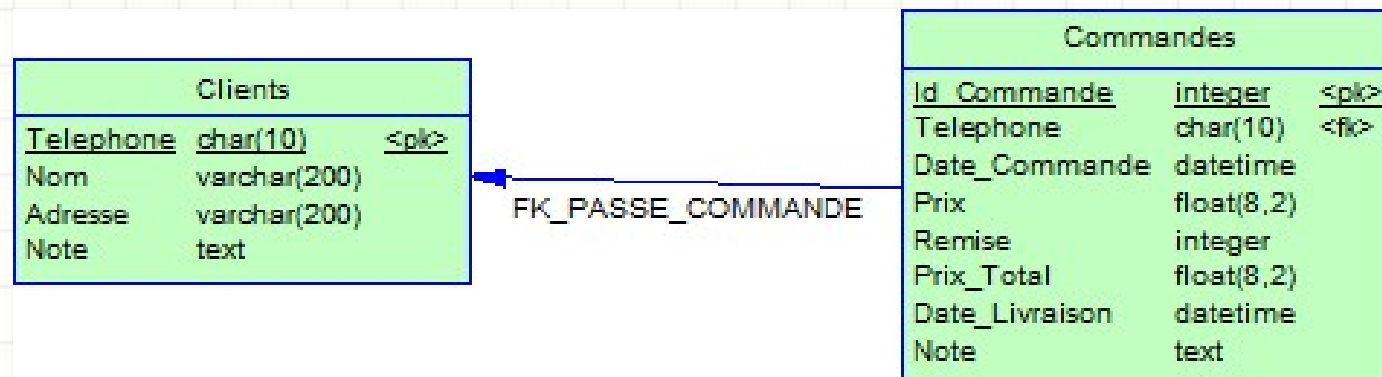
colonne 1	colonne 2
Ligne 1	donnée
Ligne 2	donnée
Ligne 3	donnée
Ligne 4	donnée
Ligne 5	donnée
etc...	etc...

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.

Clés primaires et étrangères

- Prennons le cas où un client commande une pizza, les tables "clients" et "commandes" auront toutes les deux des "**clés primaires**" (Telephone et Id_Commande) qui définiront d'une manière unique un client et une commande.



- La colonne « Telephone" de la table Commandes fait référence à la colonne Telephone de la table client (donc à la clé primaire).
- Ce type de référence est appelée "**Clé étrangère**".

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 près 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 régulièrement 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 », <http://www.mysql.com/products/mysqlcc>).
- L'interface la plus répandue est phpMyAdmin

Interfaces à MySQL

Cet exemple montre comment établir une connexion au serveur MySQL à partir d'un shell Unix.

Cette connexion est ouverte avec le nom d'utilisateur « michel » (-u michel), un mot de passe devra être entré (-p).

Une fois la connexion établie, un prompt (mysql>) attends les requêtes SQL.

Une requête simple de sélection est effectuée sur la table « pizzas ».

Le résultat de cette requête (qui est l'ensemble des lignes de la table) est affiché.

```
michel@michel-Satellite-L670D: $ mysql -u michel -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 100
Server version: 5.7.27-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use pizzeria;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from PIZZAS;
+-----+-----+-----+-----+
| ID_PIZZA | NOM                | TAILLE | PRIX |
+-----+-----+-----+-----+
| 1       | Reine Normale      | 19     | 9.50 |
| 2       | Reine Grande       | 25     | 12.50|
| 3       | Végétarienne petite| 19     | 7.50 |
| 4       | Végétarienne grande| 25     | 12.50|
| 5       | Royale petite      | 19     | 9.50 |
| 6       | Royale grande      | 25     | 12.50|
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> exit
Bye
michel@michel-Satellite-L670D: $
```

phpMyAdmin, installation

Afin de pouvoir utiliser phpMyAdmin il est fortement conseillé d'installer LAMP. Vous disposerez ainsi d'Apache, de MySQL, de Php et de PhpMyAdmin.

Vous pourrez donc accéder à phpMyAdmin à partir de votre poste Windows ou Linux à partir de votre browser en tapant :

<http://localhost/phpmyadmin>

ou


<http://ipadress/phpmyadmin>

Exemple :

<http://192.168.0.17/phpmyadmin>

Important : inclure la ligne suivante dans /etc/apache2/apache2.conf

```
Include /etc/phpmyadmin/apache.conf
```

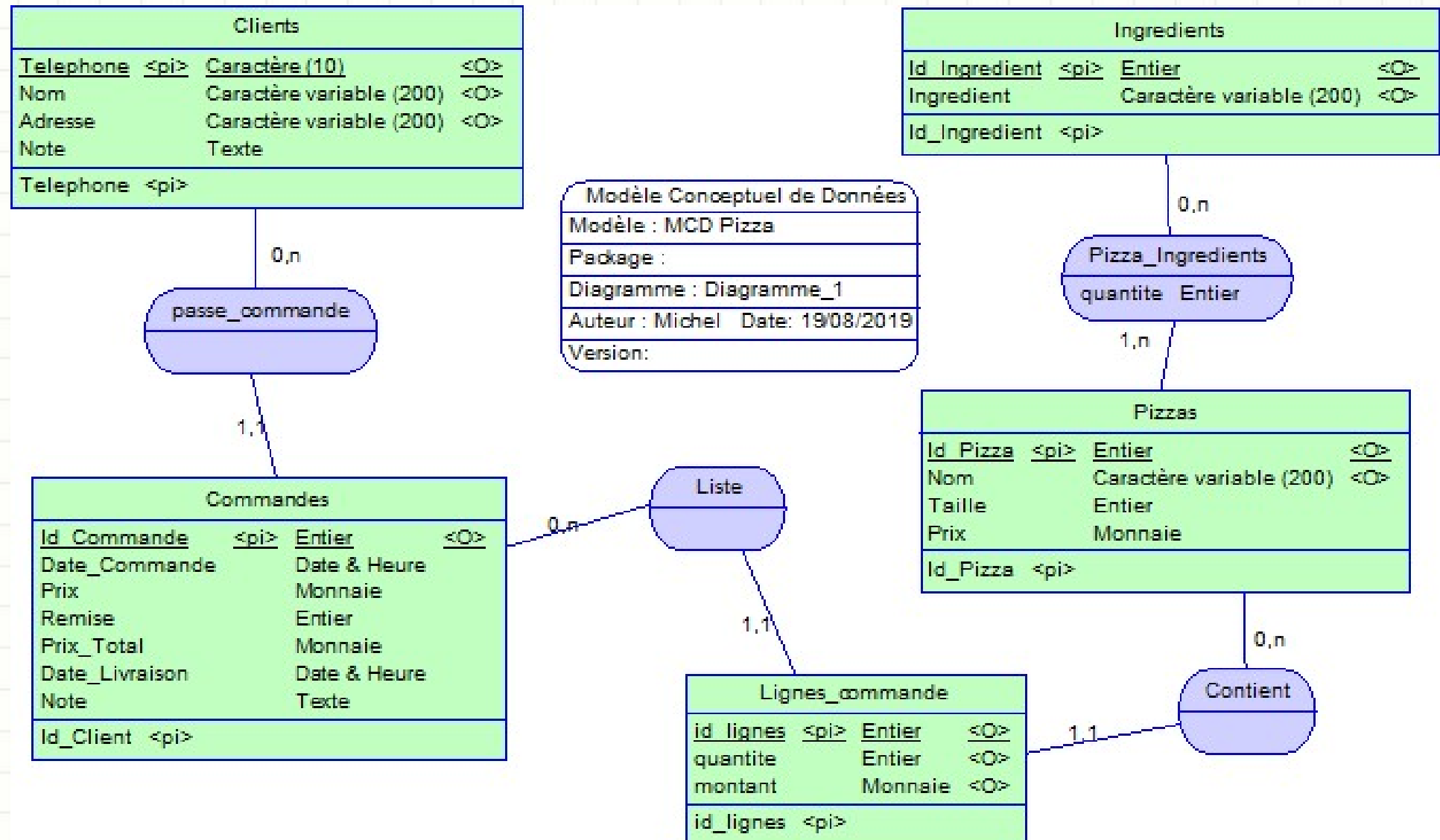


The screenshot shows the phpMyAdmin web interface. At the top is the phpMyAdmin logo, which features a stylized sailboat. Below the logo, the text "Bienvenue dans phpMyAdmin" is displayed. There are two main sections: "Langue - Language" and "Connexion". The "Langue - Language" section contains a dropdown menu currently set to "Français - French". The "Connexion" section contains two input fields: "Utilisateur :" with the text "phpmyadmin" and "Mot de passe :" with a masked password represented by dots. At the bottom right of the "Connexion" section is a button labeled "Exécuter".

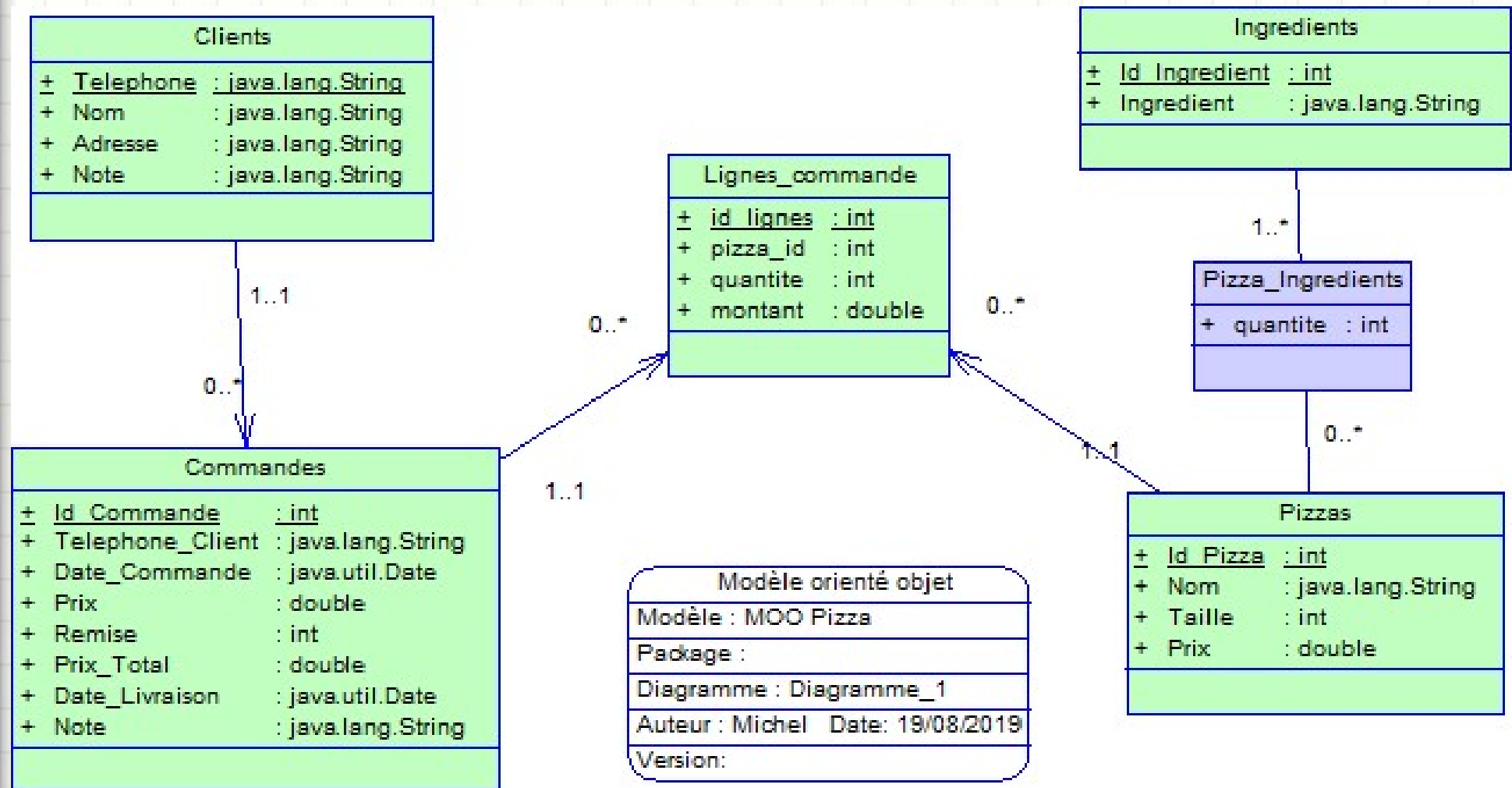
Base de données pizzas

- Tout au long de ce support de cours les exemples se référeront à une base de données fictive. Cette bases modélise (très simplement) une activité de fabrication et de livraison de pizzas.

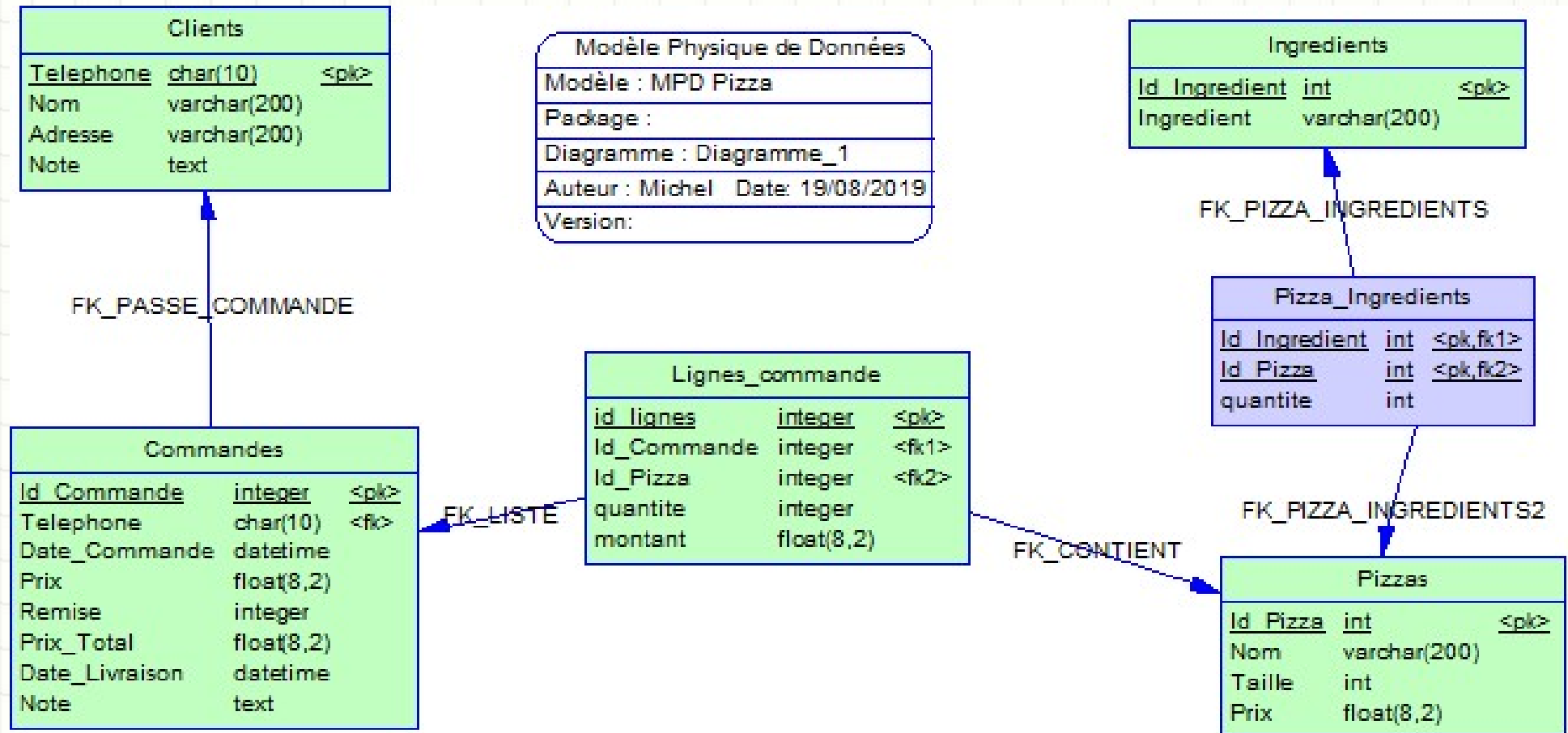
Base de données pizzas MCD Merise



Base de données pizzas MCD UML



Base de données pizzas MPD



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='Reine grande';

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 INGREDIENT desc;

Il est également possible de réaliser des calculs. Combien coûtent trois pizzas reines ?

select PRIX*3 from PIZZAS where nom='Reine petite';

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 LIGNES_COMMANDE where ID_COMMANDE >= 2;

select NOM, PRIX from PIZZAS where PRIX between 8 and 10;

select * from INGREDIENTS where ID_INGREDIENT in (1, 2, 4);

select NOM, NOTE from CLIENTS where NOTE is not null;

select * from PIZZAS where NOM like 'Reine%' order by NOM;

Opérateur booléens : **and et or**. Exemple :

select * from PIZZAS where TAILLE = 3 and prix < 10;

LMD, select

Le verbe **select** dispose d'une clause qui permet de limiter le nombre 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 pizzeria ? ». Une telle question s'écrit ainsi :

```
select count(*) from CLIENTS;
```

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

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

Exemple :

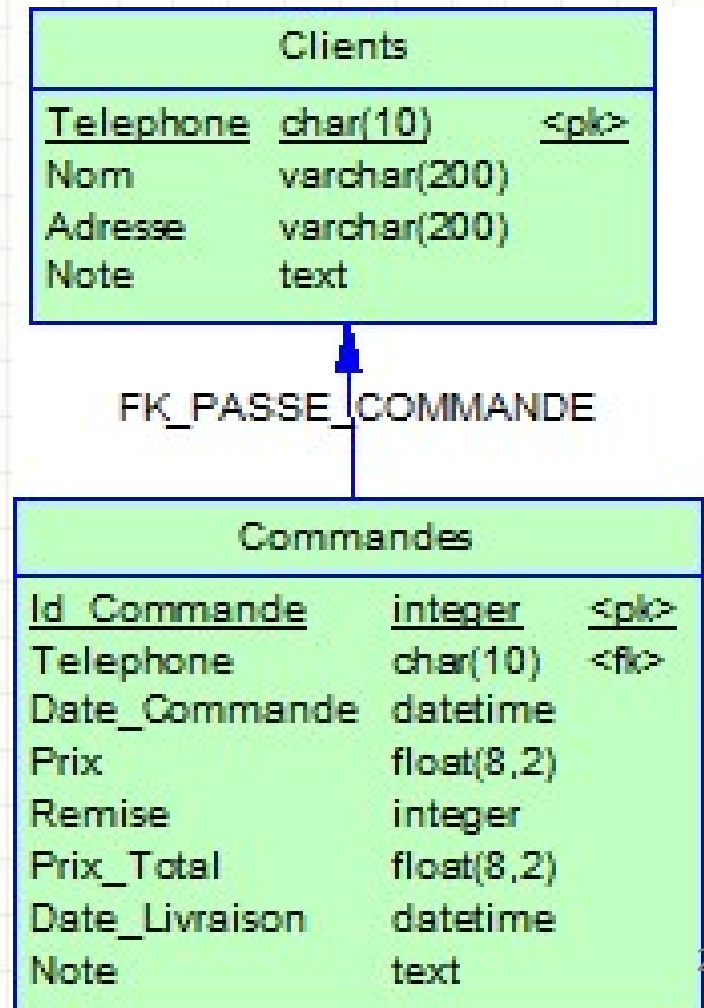
```
select TELEPHONE, sum(PRIX_TOTAL) from COMMANDES group by  
TELEPHONE;
```

LMD, Jointures

Select **COMMANDES.DATE_COMMANDE**, **COMMANDES.PRIX_TOTAL**,
CLIENTS.NOM from **CLIENTS**, **COMMANDES** where
CLIENTS.TELEPHONE = COMMANDES.TELEPHONE order by 1;

Le schéma ci-contre montre côté à côté un morceau du MPD de la base de données, le contenu des tables clients et commandes et le résultat de la requête ci-dessus.

Il met en relief le fait que les jointures ne sont que l'incarnation des liens relationnels qui existent entre les tables de la base de données.



LMD, Jointures

Il existe deux formes pour écriture les jointures :

```
select LIGNES_COMMANDE.ID_COMMANDE, LIGNES_COMMANDE.QUANTITE,  
       PIZZAS.NOM from LIGNES_COMMANDE, PIZZAS  
where LIGNES_COMMANDE.ID_PIZZA = PIZZAS.ID_PIZZA order by 1;
```

```
select LIGNES_COMMANDE.ID_COMMANDE, LIGNES_COMMANDE.QUANTITE,  
       PIZZAS.NOM from LIGNES_COMMANDE inner join PIZZAS on  
       LIGNES_COMMANDE.ID_PIZZA = PIZZAS.ID_PIZZA order by 1;
```

Ces deux requêtes fournissent le même résultat.

LMD, Jointures

Si nous reprenons notre exemple de palmarès des pizzas vendues, il serait bien pratique d'afficher en première colonne le nom de la pizza plutôt que son identifiant.

Ceci est possible en utilisant une jointure.

```
select PIZZAS.NOM, sum(LIGNES_COMMANDE.QUANTITE) as total
  from LIGNES_COMMANDE, PIZZAS
 where LIGNES_COMMANDE.ID_PIZZA = PIZZAS.ID_PIZZA
 group by LIGNES_COMMANDE.ID_PIZZA
 order by total
 desc limit 3;
```

LMD, Jointures

Une jointure peut se faire avec plus de deux tables.

Ainsi, si nous souhaitons obtenir la liste et la quantité des ingrédients pour chaque pizza (afin de l'afficher en cuisine) :

```
select  PIZZAS.NOM as pizza, INGREDIENTS.INGREDIENT as ingredient,  
        PIZZA_INGREDIENTS.QUANTITE  
from    PIZZAS, INGREDIENTS, PIZZA_INGREDIENTS  
where   PIZZA_INGREDIENTS.ID_PIZZA = PIZZAS.ID_PIZZA  
and     PIZZA_INGREDIENTS.ID_INGREDIENT = INGREDIENTS.ID_INGREDIENT  
order  by 1;
```

Le nom de chaque colonne doit être préfixé par le nom de la table dont il est issu car il y aurait sinon 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 toutes 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_ingredient, ingredient) values (5005, 'Poivron');
```

est équivalent à :

```
insert into ingredients values (5005, '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-à-dire 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 ».

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 n°7 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 optionnelle. Toutes les lignes sont supprimées si elle est omise.

Dans une base de données relationnelle, il n'est pas possible de supprimer des lignes d'une table si des lignes d'une autre table font référence à l'une des valeurs de ces lignes (clé étrangère).

Cependant MySQL n'implémente pas ce genre de contrainte d'intégrité. Il conviendra donc de vérifier au préalable que de telles références n'existe pas avant d'effectuer la suppression.

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. Sa syntaxe est :

```
create table 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

Les types de données les plus utilisés sont :

<code>varchar(n)</code>	chaîne de caractères de taille variable
<code>char(n)</code>	chaîne de caractères de taille fixe
<code>integer</code>	entier
<code>float</code>	nombre à virgule flottante
<code>date</code>	date
<code>time</code>	heure
<code>timestamp</code>	date et heure (positionné à la date et l'heure actuelle si mis à NULL)
<code>datetime</code>	date et heure
<code>text</code>	textes longs
<code>blob</code>	valeur binaire longue
<code>enum(liste)</code>	énumération

LDD, create table

Pour gérer les livraisons, nous avons plusieurs livreurs. Nous allons donc ajouter une table nous permettant d'indiquer qui livre chaque commandes. Un livreur est caractérisé par son identifiant unique, son prénom et sa date d'embauche. De plus une colonne permet d'entrer un commentaire.

```
create table livreurs
```

```
( idinteger          not null , prenom          varchar(50) not null  
, date_embauche date      , commentaire      text  
, primary key (id)          , unique (prenom) );
```

Ou bien :

```
create table livreurs
```

```
( idinteger          not null primary key  
, prenom            varchar(200) not null unique  
, date_embauche date      , commentaire      text );
```


LDD, create table

Pour gérer les livraisons, nous avons plusieurs livreurs. Nous allons donc ajouter une table nous permettant d'indiquer qui livre chaque commandes. Un livreur est caractérisé par son identifiant unique, son prénom et sa date d'embauche. De plus une colonne permet d'entrer un commentaire.

create table livreurs

```
( idinteger          not null , prenom          varchar(50) not null
, date_embauche date      , commentaire text
, primary key (id)          , unique (prenom) );
```

Ou bien :

create table livreurs

```
( idinteger          not null primary key
, prenom          varchar(200) not null unique
, date_embauche date      , commentaire text );
```

livreurs	
<u>id</u>	integer
prenom	varchar(200)
date_embauche	date
commentaire	text

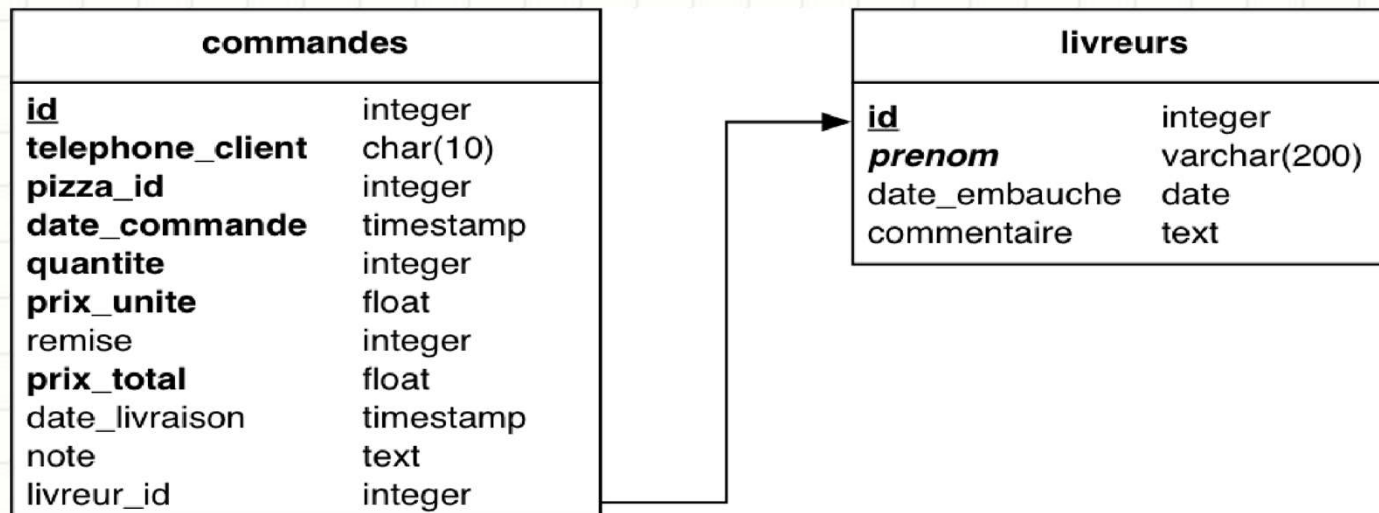
LDD, alter table

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

Elle est fréquemment utilisée pour ajouter une colonne à un table.

Par exemple, il nous est maintenant nécessaire d'ajouter la colonne `livreur_id` à la table `commandes`. Cette colonne contiendra l'identifiant du livreur ou la valeur `NULL` si le client est venu chercher la pizza sur place.

```
alter table commandes add column livreur_id integer;
```



LDD, alter table

La commande **alter table** permet également :

- d'ajouter une clé unique

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

- supprimer une colonne

alter table table **drop column** colonne

- changer le nom et le type d'une colonne

alter table table **change column** colonne nouveau_nom
nouveau_typecolumn

- renommer une table

alter table table **rename** as nouveau_nom

La commande **drop table** supprime une table :

drop table table

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...
```

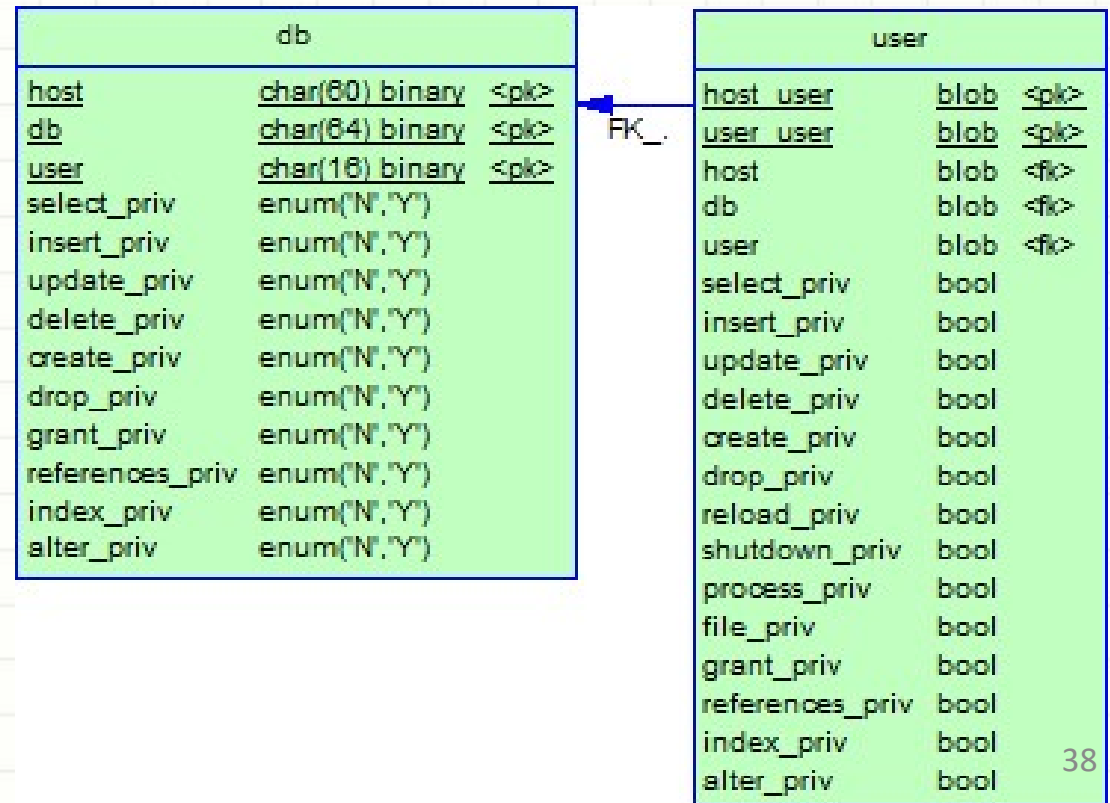
Droit d'accès

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 ».se

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.



Droit 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` ».

Droit d'accès

La table db précise les privilèges des utilisateurs pour chaque base de données.

Les privilèges contenus dans cette table sont pris en compte uniquement si le même privilège est « N » dans la table user pour le même couple user@host.

L'exemple ci-contre montre une configuration classique où l'utilisateur root@localhost possède tous les privilèges (il est considéré comme administrateur) et l'utilisateur pizzeria@localhost ne possède aucun privilège au niveau global et presque tous les privilèges sur une seule base de données.

table user

host	localhost	localhost
user	root	pizzeria
password	jupiter	charly
select_priv	Y	N
insert_priv	Y	N
update_priv	Y	N
delete_priv	Y	N
create_priv	Y	N
drop_priv	Y	N
reload_priv	Y	N
shutdown_priv	Y	N
process_priv	Y	N
file_priv	Y	N
grant_priv	Y	N
references_priv	Y	N
index_priv	Y	N
alter_priv	Y	N

table db

host	localhost
db	pizzas
user	pizzeria
select_priv	Y
insert_priv	Y
update_priv	Y
delete_priv	Y
create_priv	Y
drop_priv	Y
grant_priv	N
references_priv	Y
index_priv	Y
alter_priv	Y

Droit d'accès

Il existe deux méthodes pour gérer les utilisateurs et les privilèges qui y sont associés :

- modifier les tables user, db, etc. en utilisant le LMD ;
- utiliser les commandes SQL grant, revoke et set password.

La seconde méthode est désormais préférée à la première.

Si l'une des tables user, db, host, tables_priv ou column_priv est modifiée, il est nécessaire d'utiliser l'instruction flush privileges afin que MySQL prenne en compte les changements.

Par exemple :

```
update user set password=password('Z') where user='root' and  
host='localhost';
```

```
flush privileges;
```

Ceci n'est pas nécessaire lorsque les commandes grant, revoke et set password sont utilisées. Cette commande est équivalente aux deux commandes de l'exemple précédent :

```
set password for root@localhost = password('Z');
```

Droit d'accès

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 optionelles. 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).

Droit d'accès

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.

Droit d'accès

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');
```

Droit d'accès

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;
```

```
+-----+
| Grants for pizzeria@localhost |
+-----+
| GRANT USAGE ON *.* TO 'pizzeria'@'localhost' IDENTIFIED BY PASSWORD '72de524a554dc726'
| GRANT SELECT, INSERT, UPDATE, DELETE ON pizzas.* TO 'pizzeria'@'localhost'
+-----+
```

2 rows in set (0.00 sec)

Fonctions

MySQL met à disposition un ensemble de fonctions.

Ces fonctions peuvent être utilisées dans les clauses select, where, update du LMD.

Elle sont traditionnellement rangées selon ces catégories : numériques, chaînes, dates, générales et agrégeantes.

Consulter la référence de MySQL pour obtenir la description précise de ces fonctions

(par exemple ici : <https://fr.wikibooks.org/wiki/MySQL/Fonctions>).

Seules quelques fonctions sont présentées sous forme d'exemples dans les pages qui suivent.

Fonctions

```
select user();
```

```
-> pizzeria@localhost
```

```
select substring_index( user(), '@', 1 );
```

```
-> pizzeria
```

```
select version();
```

```
-> 3.23.49-log
```

```
select sum(prix_total), round( sum(prix_total), 2 ) from commandes;
```

```
-> 167.59999847412 167.60
```

```
select concat( 'Pizza ', nom ) from pizzas;
```

```
-> Pizza Reine
```

```
    Pizza Roi
```

```
    Pizza Végétarienne
```

Fonctions

```
update commandes set date_livraison = now() where id = 11;  
select dayofweek(date_commande) as jour, count(*) as commandes  
from commandes group by dayofweek(date_commande) order by 2 desc;
```

-> jour	commandes
7	4
6	2
1	2
2	1
3	1

(1=dimanche, 2=lundi, ..., 7=samedi)

```
select nom, ifnull(note, '-pas de commentaire-') as note from clients;
```

-> nom	note
David Solomon	-pas de commentaire-
Linda Tchaïkowsky	-pas de commentaire-
Arthur Bab's	Trés bon client
Alex Feinberg	Mauvais payeur

Index

Lorsque un serveur MySQL évalue une clause where, il doit parcourir l'ensemble des valeurs des colonnes concernées. Cette recherche est coûteuse en temps.

Afin d'améliorer la vitesse d'exécution des requêtes sur des tables importantes, on utilise des objets nommés index. Un index est une relation simple entre l'ensemble des valeurs définies sur une colonne et les lignes de la table qui contiennent ces valeurs.

L'index est mis-à-jour en temps réel lorsque les lignes de la table sont modifiées par des requêtes insert, update ou delete. Par conséquent, plus il y a d'index sur une table, moins l'enregistrement des modifications est rapide. Mais le gain en performance sur la clause where est tel que les index restent très efficaces dans presque toutes les situations.

Index

Les index sont spécifiés lors de la création d'une table ou en utilisant le verbe alter table. Par ailleurs, des index implicites sont créés lorsque des clés primaires ou uniques sont utilisées, par exemple :

```
create table livres
( id integer unsigned not null primary key
, code_isbn varchar(20) not null unique
, titre varchar(200) not null
, auteur_id integer unsigned not null
, date_parution date not null
, resume text
, index (titre)
, index (auteur_id)
, index (date_parution) );
```

Cette requête de création de table va générer cinq index en tout.

Utiliser beaucoup d'index est cohérent dans ce cas car le nombre de modification qui auront lieu sur cette table sera probablement bien inférieur au nombre de requêtes.

Index

La commandes show index permet d'obtenir la liste de tous les index d'une table. Par exemple :

```
mysql> show index from livres;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Comment
livres	0	PRIMARY	1	id	A	0	NULL	NULL	
livres	0	code_isbn	1	code_isbn	A	0	NULL	NULL	
livres	1	titre	1	titre	A	NULL	NULL	NULL	
livres	1	auteur_id	1	auteur_id	A	NULL	NULL	NULL	
livres	1	date_parution	1	date_parution	A	NULL	NULL	NULL	

5 rows in set (0.00 sec)

Il est possible de nommer les index et les clés uniques lors de leur création :

```
create table livres
```

```
.../...
```

```
, unique code_isbn_uk (code_isbn)
```

```
, index titre_idx (titre)
```

```
.../...
```

Index

Le verbe alter table permet de supprimer ou de créer des index sur une table a posteriori.

```
alter table table add index [nom_index] (colonnes)
alter table table add unique [nom_index] (colonnes)
alter table table drop index nom_index
alter table table drop primary key
```

Il existe également ces formes équivalentes :

```
create [unique] index [nom_index] on table (colonnes)
drop index nom_index on table
```

Colonnes auto-increment

Très souvent les clés primaires des tables sont constituées d'une seule colonne dont le type est un entier. Il est alors très intéressant d'utiliser l'option `auto_increment` pour cette colonne lors de la création de la table.

Lors de l'insertion, si une colonne possède cette option et si la valeur qui y est insérée est `NULL`, MySQL affecte automatiquement la prochaine valeur à cette colonne. Par exemple :

```
create table messages
( id integer unsigned not null auto_increment primary key
, texte text not null );
insert into messages (texte) values ('Hello World!');
insert into messages values (NULL, 'Salut et merci pour le
    poisson');
select * from messages;
-> id  texte
    1  Hello World!
    2  Salut et merci pour le poisson
```


Colonnes auto-increment

Il ne peut y avoir qu'une seule colonne auto_increment dans une table et cette colonne doit faire partie de la clé primaire.

La fonction last_insert_id est utile pour connaître la valeur qui a été donnée à cette colonne lors de l'insertion.

Par exemple :

```
insert into messages (texte) values ('Et pourtant elle  
tourne');
```

```
select last_insert_id();
```

```
+-----+
```

```
| last_insert_id() |
```

```
+-----+
```

```
|          3      |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```



**MERCI DE VOTRE
ATTENTION**