BASE de DONNEES

SYSTEME de GESTION de BASE de DONNEES

MySQL / SQL

PHP_MYSQL

Sommaire

LES BASES de DONNEES

POURQUOI UTILISER DES BASES de DONNEES

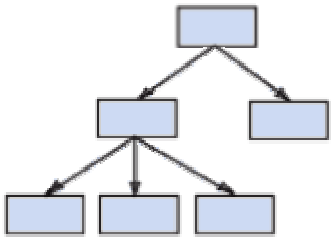
Lorsqu'on a besoin d'organiser les données en ensemble structuré, afin de

- stocker
- consulter
- modifier

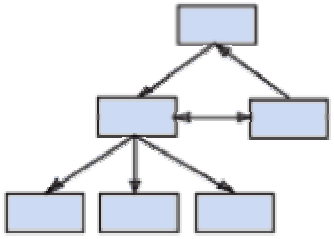
ces informations

PLUSIEURS MODELES de BASES de DONNEES

- **le modèle hiérarchique**: les données sont classées hiérarchiquement, selon une arborescence descendante. Ce modèle utilise des pointeurs entre les différents enregistrements. Il s'agit du premier modèle de SGBD

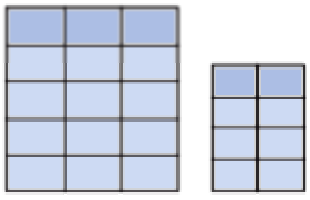


- **le modèle réseau**: Comme le modèle hiérarchique ce modèle utilise des pointeurs vers des enregistrements. Toutefois la structure n'est plus forcément arborescente dans le sens descendant



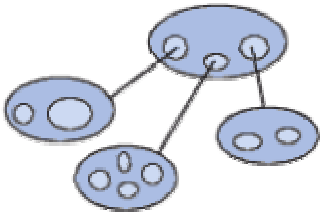
- **le modèle relationnel** (**SGBDR**, *Système de gestion de bases de données relationnelles*): les données sont enregistrées dans des tableaux à deux dimensions (lignes et colonnes). La manipulation de ces données se fait selon la théorie mathématique des relations , théorie ensembliste. (du mathématicien CODD)

(les ensembles = tables /
les colonnes = attributs (nom, prenoms,)
les lignes = les enregistrements = les tuples)



- **le modèle objet** (**SGBDO**, *Système de gestion de bases de données objet*): les données sont stockées sous forme d'objets, c'est-à-dire de structures appelées *classes* présentant des données

membres. Les champs sont des instances de ces classes



- **XML** (bâtie sur un référentiel de contenudécrit et structuré en XML via des DTD ou Schémas. Le langage de requête est du XML : Xquery, XPath) eXist, Apache Xindice

A la fin des années 90 les bases relationnelles sont les bases de données les plus répandues (environ trois quarts des bases de données).

NB 1 : Particularité de bases dites « spécialisées » : documentaires ou géographiques où les schémas traditionnels ne conviennent pas .

LE MODELE RELATIONNEL

Le stockage des données ----------> Tables
Le logiciel ----------> Système de Gestion de BD
(DBMS en anglais, pour Data Base Management System)

CARACTERISTIQUES d'un SGBD

Un logiciel et son fonctionnement

Un SGBD est principalement constitué d'un **moteur** et **d'une interface graphique**. Le moteur est le coeur du logiciel, c'est à dire qu'il assure les fonctions essentielles :

- saisir les données,
- les stocker,
- les manipuler,
- etc.

L'interface graphique permet à l'utilisateur de communiquer commodément avec le logiciel. Pour dialoguer avec les SGBD qui ne sont pas équipés d'une interface graphique, il faut utiliser le langage **SQL** (Structured Query Language), et introduire les instructions à l'aide d'un éditeur de lignes.

Les caractéristiques

L'architecture à trois niveaux définie par le standard ANSI/SPARC permet d'avoir une indépendance entre les données et les traitements. D'une manière générale un SGBD doit avoir les **caractéristiques** suivantes:

- **Indépendance physique**: Le niveau physique peut être modifié indépendamment du niveau conceptuel. Cela signifie que tous les aspects matériels de la base de données n'apparaissent pas pour l'utilisateur, il s'agit simplement d'une structure transparente de représentation des informations
- **Manipulabilité**: des personnes ne connaissant pas la base de données doivent être capables de décrire leur requêtes sans faire référence à des éléments techniques de la base de données
- **Rapidité des accès**: le système doit pouvoir fournir les réponses aux requêtes le plus rapidement possible, cela implique des algorithmes de recherche rapides
- **Administration centralisée**: le SGBD doit permettre à l'administrateur de pouvoir manipuler les données, insérer des éléments, vérifier son intègrité de façon centralisée
- **Limitation de la redondance**: le SGBD doit pouvoir éviter dans la mesure du possible des informations redondantes, afin d'éviter d'une part un gaspillage d'espace mémoire mais aussi des erreurs
- **Vérification de l'intégrité**: les données doivent être **cohérentes entre elles**, de plus lorsque des éléments font références à d'autres, ces derniers doivent être présents
- **Partageabilité des données**: le SGBD doit permettre l'accès simultané à la base de données par plusieurs utilisateurs
- **Sécurité des données**: Le SGBD doit présenter des mécanismes permettant de gérer les droits d'accès aux données selon les utilisateurs

Différencier les SGBDR

Tous les SGBDR présentent à peu près les mêmes fonctionnalités . Ils se distinguent par :

- leur coût,
- le volume de données qu'ils sont capables de gérer,
- le nombre d'utilisateurs qui peuvent interroger la base simultanément,
- la facilité avec laquelle ils s'interfacent avec les autres logiciels d'application

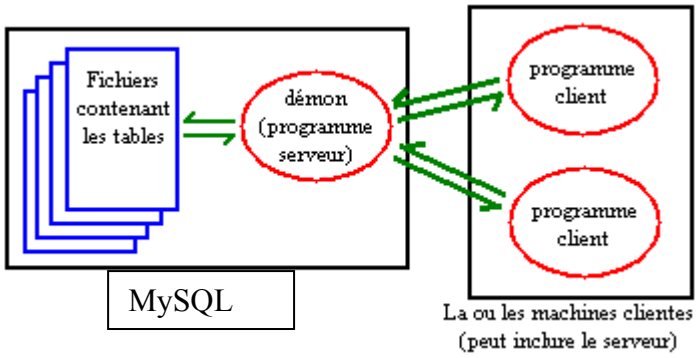
Comment ça marche

Le logiciel de gestion de base de données relationnelles comporte un ensemble de fonctionnalités pour créer des tables, les manipuler, mais aussi y accéder, conformément au modéle relationnel de données.

Un ensemble de fichiers, propre au logiciel et interne à ce dernier, lui permet d'assurer ses fonctions. Ces fichiers sont gérés par un logiciel serveur aussi appelé démon (qui doit donc tourner pour pouvoir accèder à la base). Chaque logiciel a sa propre organisation physique et gestion interne (d'où certaines différences).

Pour utiliser une table en dehors de ce contexte, il est nécessaire de l'EXPORTER « en dehors » du logiciel. Pareillement, on IMPORTERA des données du monde réel vers/dans la base.

L'utilisateur (client) devra s'adresser au serveur (service) de BD pour pouvoir manipuler les données.



Comment sont rangées les données

Une base de données se compose, donc, d'une à plusieurs tables, dont chacune est identifiée par un nom.
Les tables d'une base de données sont, en principe, toutes reliées entre elles selon un schéma de relation.
Les tables contiennent un à plusieurs enregistrements, c'est-à-dire des lignes de données.

Chacune des tables d'une base de données se décompose en un à plusieurs champs appelés également colonnes.

Table			
Champ_1 (col_1)	**Champ_2 (col_2)**	**Champ_3 (col_3)**	**Champ_4 (col_4)**
Valeur	Valeur	Valeur	Valeur
Valeur	Valeur	Valeur	Valeur

(Ligne =<---------- Enregistrement ---------->)

Ces colonnes sont représentées d'une part par un nom individuel servant à leur identification dans une table et d'autre part par **un type de données** pour le genre d'informations qu'elles comprennent comme du texte, des nombres, des dates et des heures ou encore des valeurs binaires (BLOB : Binary Large OBject) telles que des images.

Fiche_Personne			
id	**Nom**	**Prenom**	**CP**
NUMBER(10)	**VARCHAR(20)**	**VARCHAR(20)**	**NUMBER(5)**

Chaque enregistrement d'une table doit possèder une clé unique, utilisée pour les distinguer individuellement, à l'image d'un numéro de sécurité sociale pour chaque individu. Une colonne spécialement conçue à cet effet doit contenir ce genre d'informations où aucun doublon n'est permis. Cette colonne est appelée la clé primaire d'une table. L'indexation ainsi effectuée permet un accès rapide et sans équivoque à un enregistrement particulier d'une table.

Fiche_Personne			
id	**Nom**	**Prenom**	**CP**
187	JANVIER	Denis	77870
1097	NAPOLI	Victor	75020

Le schéma de relation des tables précité, dépend étroitement des clés primaires affectées à chaque table de la base de données.
Les tables sont reliées les unes aux autres par ces fameuses clés communes.
Par exemple, une table pourrait contenir deux colonnes, une appelée clé primaire et l'autre clé secondaire. Une seconde table contiendrait une clé primaire différente et une clé secondaire correspondant à la clé primaire de la première table. En conséquence, la seconde table pourra être mise en relation avec la première table par le biais de cette dernière clé strictement identique aux deux tables. il sera alors, possible de joindre les enregistrements connexes des deux tables entre eux.

La manipulation des données s'effectue par l'intermédiaire de requêtes regroupant plusieurs instructions SQL. Une requête précise est capable entre autres, d'accomplir des extractions, des ajouts, des mises à jour, des suppressions de données.
L'exécution de certaines requêtes sur des bases de données peuvent retouner des objets spécifiques, comme des vues ou des curseurs.

Une vue est une table virtuelle dont le contenu est déterminé par une requête. Une vue possède donc une structure identique à celle d'une table de base de données hormis que ses lignes et ses colonnes proviennent d'une à plusieurs tables indiquées dans la requête.

Un curseur représente la valeur en cours stockée en mémoire et résultant d'une requête appliquée sur une base de données. Un curseur peut contenir d'un à plusieurs enregistrements qui peuvent être accédés par des commandes SQL spécifiques

---

En résumé

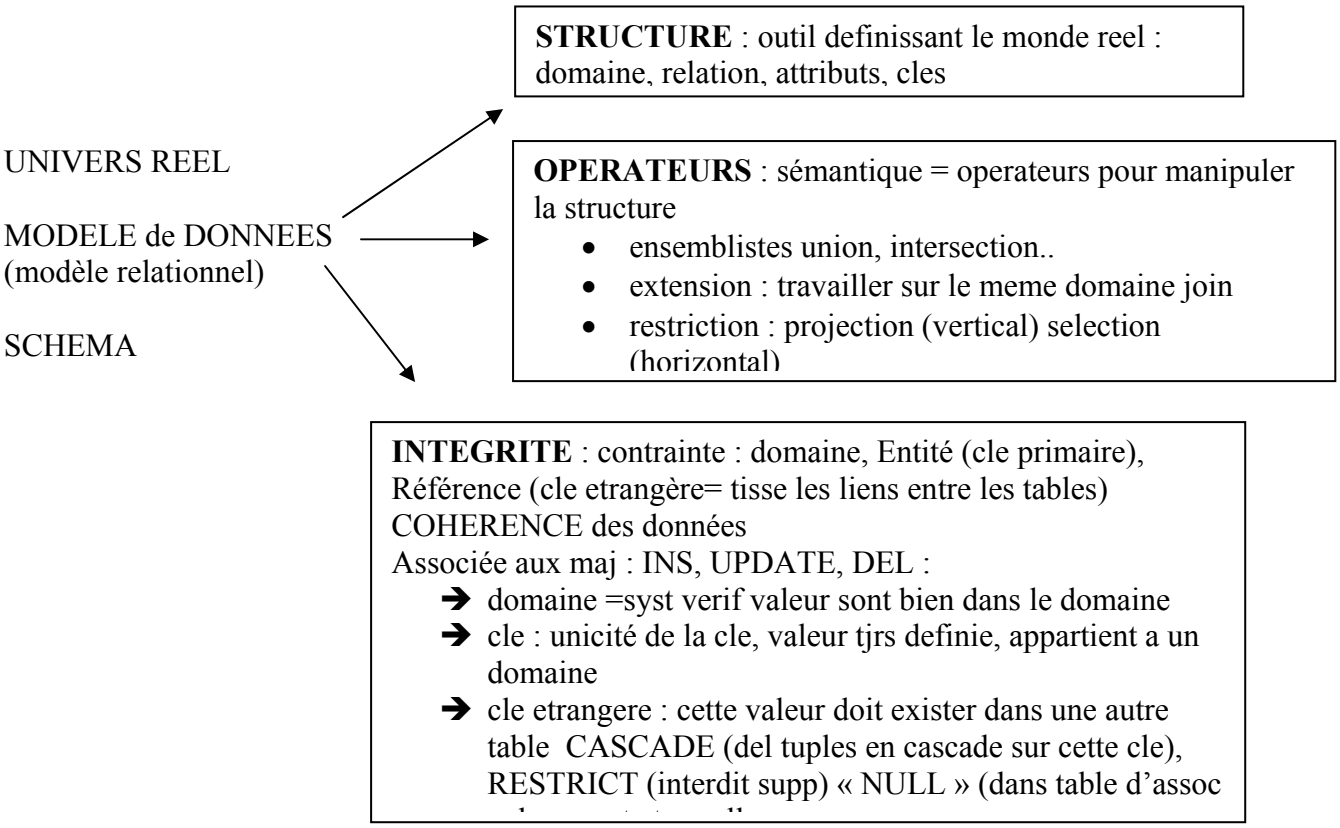
On s'appuie sur un système de gestion de base de données relationnelle (SGBDR) qui est un LOGICIEL pour ORGANISER les éléments d'information (fichiers, textes, images) au sein d'une structure en TABLES liées entre-elles.

Ce SGBDR se chargera d'assurer l'intégrité des données

Leur contenu est accessible par des applications clients via **des langages de requêtes**, exécutés via **des interfaces** : ODBC, JDBC. Le langage le plus utilisé est le SQL (Structured Query Language)

SGBDR publiques	Commerciaux
- Interbase (Borland) - MySql - Postgresql (SGBD OR)	- SAP DB - DB2 (IBM) - **SQL-Sserveur (microsoft)** - **Oracle 9** - **Sybase**

MODELE de DONNEES RELATIONNEL



Terminologie

Domaine = ensemble des valeurs représentant une partie de l univers ; ensemble des valeurs d'un attribut

Exemple : avion{A300,A350} ville{nom ville, code postal} représente une colonne

Relation = relation perçue entre les entités :

Exemple : DUPONT **achete** le produit no21

sous-ensemble du produit cartesien d'une liste de domaines. C'est un tableau à 2 dimensions ; les colonnes correspondent aux domaines et les lignes contiennent des tuples. On associe un nom à chaque colonne.

Entité : objet du monde réel , un nom

Exemple : le client dupont, l'avion Airbus A320, le pilote Torre

Attribut / Propriété = une colonne d'une relation caractérisée par un nom

chaque attribut prend ses valeurs dans son domaine exemple ville de depart ville d'arrivee

Exemple : age, date, vol

Tuple : liste des valeurs d'une ligne d'une relation

Clé primaire = Entité/relation element unique : rôle identifiant une relation

Clé etrangère = référence = attribut principal ailleurs

Règles de gestion

Il existe 2 grandes méthodes d'analyses selon le modèle étudié :
- RELATIONNEL -> méthode MERISE
- OBJET -> « langage » UML

DOCUMENT sur la méthode d'analyse MERISE
ftp://ftp2.developpez.be/developps/sgbd/ConceptionBD.pdf

EXEMPLE / EXERCICE

Entité statique ou INDEPENDANTE / : personne, pilote, avion, vol, departement

Entité dynamique : dépendantes d'autres entités => il existe des liens

1- définir les entités statiques
2- rechercher les liens entre les entités
3- rajout des cles étrangères

Imaginons que l'on veuille stocker dans notre base de données notre carnet d'adresses. On va donc créer la relation ***Personne*** qui aura pour attributs : *nom*, *prénom*, *adresse*, *téléphone*. Autrement dit, c'est une table nommée ***Personne*** possédant les colonnes : *nom*, *prénom*, *adresse*, *téléphone*.

Les *lignes* que contiendra cette table seront appelées *enregistrements ou tuples*.

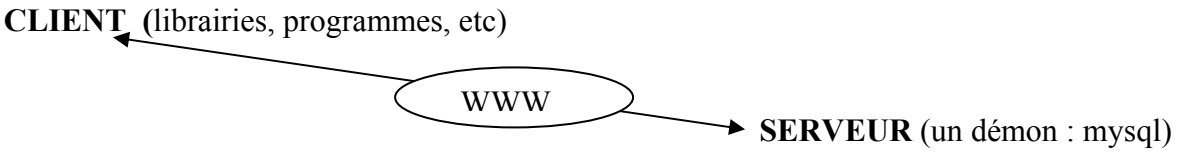
➔ donner la relation PERSONNE

MySQL

Créée en 1996 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.

Base implémentée selon un mode client-serveur : MySQL fonctionne en tant que **service**.



VERSIONS

- dernière version stable : v4.
- version bêta : v5 : compatibilité SQL-2, JDBC type 4, ...

NB : Ne retenir que les versions « exe » ou « binaires » directement exécutables

http://www.nexen.net/docs/mysql/annotee/manuel_tocd.php

ATTENTION !
Ne répond pas aux propriétés « ACID » d'un SGBD

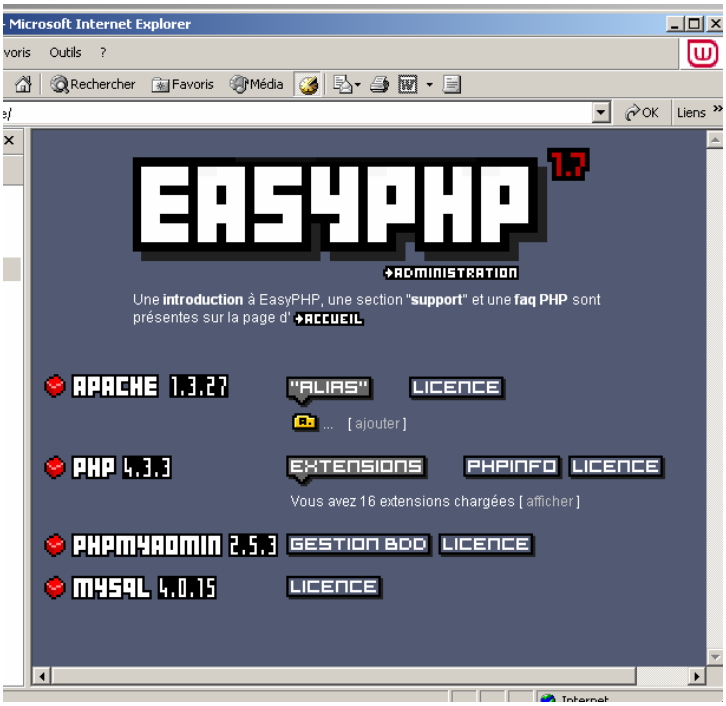
- pas de procédures stockées ni de trigger (fonctions développées liées à des actions)
- pas de langage propre de procédure comme pl/sql ou pgsql
- pas de contrôle d'intégrité
- pas de vues

POINTS FORTS

- communauté d'utilisateurs importante
- développements actifs tendant à assurer les propriétés d'un SGBD (la v5 en principe)

INSTALLATION

- **Système Windows** : Le « package » EasyPHP v 1.7 contient :
  - PHP v4.3.3
  - MySQL v4.0.15 : le SGBDR
  - PhpMyAdmin v2.5.3 : le logiciel graphique admnistratant la base de données
  - Apache v1.3 : le serveur web



L'intérêt majeur de cette distribution : celui d'une installation facile, et opérationnelle .

- **Système Linux :** télécharger chaque module séparément . Cependant certaines marques intègrent ces services (web, SGBDR) dans leur distribution. Il faut alors les sélectionner à l installation

Si vous envisagez de vous connecter à MySQL via ODBC , vous aurez aussi besoin du pilote MyODBC : http://www.nexen.net/docs/mysql/annotee/odbc.php

Quelques liens :

http://www.nexen.net/docs/mysql/annotee/windows-prepare-environment.php
http://www.nexen.net/docs/mysql/annotee/manuel_tocd.php
http://www.toutestfacile.com/phpinit.php?tef_site=sql&chap=priseenmainmysql1
Index des fonctions : http://www.nexen.net/docs/mysql/annotee/function-index.php
Index conceptual : http://www.nexen.net/docs/mysql/annotee/concept-index.php

Une copie du binaire ou de la distribution MySQL pour WIndows, qui peut être téléchargée sur http://www.mysql.com/downloads/ .

NB

mysql_install_db ne va pas écraser d'anciens droits installé, et il peut être utilisé en toutes circonstances. Si vous ne voulez pas de base test , vous pouvez la supprimer avec la commande mysqladmin -u root drop test

Help

mysql --help

CONNEXION / DECONNEXION

1. Linux (en ligne de commande)

   Pour cela, vous devez taper la commande "**safe_mysqld &**" (sous le répertoire *bin* de MySQL si ce dernier n'est pas dans le PATH)

2. Windows (en ligne de commande) cas où vous n utilisez pas easyPHP

   Ouvrez une fenêtre "Commandes MS-DOS" et déplacez vous jusqu'au répertoire *bin* de MySQL Puis tapez la commande "**mysqld**"

>Mysql –u root

```
shell> mysql -h hote -u utilisateur -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 459 to server version: 3.22.20a-
log

Type 'help' for help.

mysql>
```

LES TABLES SYSTEMES

```
shell> mysqlshow
+-----------+
| Databases |
+-----------+
| mysql     |
+-----------+
```

Liste les BD disponibles

```
shell> mysqlshow mysql
Database: mysql
+--------------+
|    Tables    |
+--------------+
| columns_priv |
| db           |
| func         |
| host         |
| tables_priv  |
| user         |
+--------------+
```

Montre les tables associées à la base « mysql »
NB : c'est une base système !

```
shell> mysql -e "SELECT host,db,user FROM db" mysql
+------+--------+------+
| host | db     | user |
+------+--------+------+
| %    | test   |      |
| %    | test_% |      |
+------+--------+------+
```

Donne les bases de données disponibles « mysql »

CREER UNE BASE de DONNEES

1- **connexion au serveur Mysql**
2- **creation ou appel de la base de données**
3- **création ou accès aux tables constituant la table**

```
Shell>Mysql –u nomUtilisateur –p pwd nomBase
Mysql> CREATE DATABASE nombase
       USE nombase
```

OU

mysqladmin create database <nombase>
(Sachant que *mysqladmin* est disponible sous le répertoire *bin* de MySQL. Si vous utilisez Windows ouvrez alors une fenêtre "Commandes MS-DOS").

Quelques synthaxes de requêtes utiles qui sont rappelées plus loin dans le cours ou dans les annexes.

	`SHOW DATABASES ; (requiert des droits , privilèges administrateur)`
	`USE madatabase ; (« ; » non obligatoire)`
	`QUIT madatabase ;`

CreatION DE TABLE	`CREATE TABLE matable (nom varchar(64), age int4);`
Modification de table	ALTER TABLE matable ADD nomCol typeCol CHANGE nomCol NewNomCol RENAME nouveauNomTable DROP nomCol ADD INDEX nomCle *ALTER TABLE aviion RENAME avion ;* *ALTER TABLE avion ADD pl# INT(3) NOT NULL AFTER [FIRST|LAST] av# ;*
	SHOW TABLES
	DESCRIBE TABLE matable
Insertion	`INSERT INTO matable (nom,age) VALUES ('DUPONT',22);` Description: "INSERT INTO matable" indique que nous voulons ajouter un enregistrement à la table "matable"; "(nom,age)" indique les champs que nous voulons spécifier (les autres champs prendrons la valeur par défaut); "VALUES ('DUPONT',22)" permet d'affecter la valeur 'DUPONT' au champ nom et 22 au champ age.
Affichage	`SELECT * FROM matable;` `SELECT age,nom FROM matable;`

Creer un utilisateur	`Use mysql ;`
	`INSERT INTO user(Host,User,Password) VALUES`
	`('localhost','myuser',PASSWORD('mypwd')) ;`
	`INSERT INTO db`
	`(Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv)`
	`VALUES ('localhost','mabase','myuser','Y','Y','Y','Y');`
	`USE mabase;`

Donner des droits	`GRANT ALL ON madatabase.* TO mon_nom_mysql;`
Suppression	`D'une table : DROP matable ;` `D'une base : DROP mabase ;`

SQL

SQL signifie **Structured Query Language** ce qui se traduit
par Langage de requêtes structurées.

Le modèle relationnel a été inventé par E.F. Codd (Directeur de recherche du centre IBM de San José) en 1970

Le langage SQL est composée de plusieurs parties :

Langage de Définition de Données (LDD)	Langage de Manipulation de Données (LMD)
La création d'une table	L' insertion de tuples
La destruction d'une table	La sélection de tuples
La création d'une vue	La suppression de tuples
La destruction d'une vue	La modification de tuples
La définition des privilèges	Les transactions
Performance	
La création d'un index	La suppression d'un index

Algèbre relationnelle

L'algèbre relationnelle regroupe toutes les opérations possibles sur les relations. Voici la liste des opérations possibles :

Projection : on ne sélectionne qu'un ou plusieurs attributs d'une relation (on ignore les autres). Par exemple n'afficher que les colonnes *nom* et *prénom* de la table ***Personnes***.

Jointure : on fabrique une nouvelle relation à partir de 2 ou plusieurs autres en prenant comme pivot 1 ou plusieurs attributs. Par exemple, on concatène la table du carnet d'adresse et celle des inscrits à la bibliothèque en fonction du nom de famille (c'es typiquement du recoupement de fichiers).

Sélection : on sélectionne tous les tuples ou bien seulement une partie en fonction de critères de sélection qui portent sur les valeurs des attributs. Par exemple n'afficher que les lignes de la table ***Personnes*** qui vérifient la condition suivante : le nom ne commence pas par la lettre 'C'.

Cette algèbre est facilement possible avec les commandes de MySQL (SELECT… FROM… WHERE…).

Projection

Personnes

nom	*prénom*	*adresse*	*téléphone*
Martin	Pierre	7 allée des vers	0258941236
Dupond	Jean	32 allé Poivrot	0526389152
Dupond	Marc	8 rue de l'octet	0123456789

SELECT *nom*, *prénom*
FROM ***Personnes***

On projette la table ***Personnes*** sur les colonnes *nom* et *prénom*.

nom	*prénom*
Martin	Pierre
Dupond	Jean
Dupond	Marc

Jointure

Personnes

nom	*prénom*	*adresse*	*téléphone*
Martin	Pierre	7 allée des vers	0258941236
Dupond	Jean	32 allé Poivrot	0526389152

Bibliothèque

nom	*Dernierlivre*
Dupond	Robinson
Jospin	Faust
Martin	Misère

SELECT *Personnes*.*prénom*, *dernierlivre*
FROM *Personnes*, *Bibliothèque*
WHERE *Personnes*.*nom* = *Bibliothèque*.*nom*

prénom	*Dernierlivre*
Jean	Robinson
Pierre	Misère

On joint les deux tables, grâce à la colonne *nom*.

Et on combine cette jointure à une projection sur les attributs *nom* et *dernierlivre*.

Attention à lever toute ambiguïté sur les noms d'attribut dans le cas où deux tables possèdent des colonnes de même nom.

Sélection

Personnes

nom	*prénom*	*adresse*	*téléphone*
Martin	Pierre	7 allée des vers	0258941236
Dupond	Jean	32 allé Poivrot	0526389152
Dupond	Marc	8 rue de l'octet	0123456789

SELECT *
FROM *Personnes*
WHERE *nom* = ''Dupond''

nom	*prénom*	*adresse*	*téléphone*
Dupond	Jean	32 allé Poivrot	0526389152
Dupond	Marc	8 rue de l'octet	0123456789

On ne sélectionne que les tuples dont l'attribut *nom* est égale à 'Dupond'.

EXERCICES

	mysql> **SHOW DATABASES;**	Liste les bases
Droits / privilèges	*mysql>* ***GRANT ALL ON mabase.* TO mon_nom_mysql;***	*Vérifier les droits*
Cration password	*shell>* ***mysql -u root mysql*** *mysql>* ***SET PASSWORD FOR root@localhost=PASSWORD('new_password');*** *Si vous savez ce que vous faites, vous pouvez aussi directement manipuler les tables de droits :*	
Modification pwd	*shell>* ***mysql -u root mysql*** *mysql>* ***UPDATE user SET Password=PASSWORD('new_password')*** ***-> WHERE user='root';*** *mysql>* ***FLUSH PRIVILEGES;*** *Une autre méthode pour modifier de mot de passe est d'utiliser la commande en ligne mysqladmin :*	
OU	*OU*	
.....		
Autre façon de faire	*shell>* ***mysqladmin -u root password new_password***	
Se connecter au serveur	*shell>* ***mysql -h hote -u utilisateur -p*** *Enter password: ******** *shell> mysql [-h nom_d_hote] [-u nom_d_utilisateur] [-p votre_mot_de_passe]* Shell>**mysql –u admin –p admin**	Connexion en tant que « dba » ou utilisateur
Créer base de données	mysql> **CREATE DATABASE mabase;** mysql> **USE mabase;**	
	mysql> ***DROP DATABASE mabase;***	
	mysql> **SHOW TABLES;**	La base de données est vide

<table>
<tr><td>Création de table(s) dans la database</td><td>mysql> CREATE TABLE animal (nom VARCHAR(20), maitre VARCHAR(20),
-> espece VARCHAR(20), sexe CHAR(1), naissance DATE, mort DATE);</td><td></td></tr>
<tr><td></td><td>mysql> DROP TABLE animal;</td><td></td></tr>
<tr><td></td><td>mysql> SHOW TABLES;</td><td></td></tr>
<tr><td></td><td>mysql> DESCRIBE animal;</td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>Charger les données dans la table

- à partir d’un fichier texte</td><td>Nb 1: MySQL attend les dates au format YYYY-MM-DD
Nb 2 : Les valeurs NULL sont représentées dans le fichier texte, par \N

Le fichier texte :
Fluffy Harold chat f 1993-02-04 \N
Claws Gwen chat m 1994-03-17 \N
Buffy Harold chien f 1989-05-13 \N
Fang Benny chien m 1990-08-27 \N
Bowser Diane chien m 1998-08-31 1995-07-29
Chirpy Gwen oiseau f 1998-09-11 \N
Whistler Gwen oiseau \N 1997-12-09 \N
Slim Benny serpent m 1996-04-29 \N

mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE animal;</td><td>LOAD DATA / INSERT
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name.txt'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[FIELDS
[TERMINATED BY '\t']
[[OPTIONALLY] ENCLOSED BY '']
[ESCAPED BY '\\']
]
[LINES
[STARTING BY '']
[TERMINATED BY '\n']
]
[IGNORE number LINES]
[(col_name,...)]</td></tr>
<tr><td>- à partir de la ligne de commande</td><td>mysql> INSERT INTO animal
-> VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);</td><td></td></tr>
<tr><td>AUTO_INCREMENT</td><td>L'attribut AUTO_INCREMENT peut être utilisé pour générer un identifiant unique pour les nouvelles lignes :

mysql> CREATE TABLE pets (
id MEDIUMINT NOT NULL AUTO_INCREMENT,
name CHAR(30) NOT NULL,
PRIMARY KEY (id)
);
mysql> INSERT INTO pets (name) VALUES ("dog"),("cat"),("penguin"),
("lax"),("whale");
mysql> SELECT * FROM pets;</td><td></td></tr>
</table>

Sélectionner les données	mysql> **SELECT * FROM animal; (visualise toute la table)**	SELECT quoi_selectionner FROM quel_table WHERE conditions_a_satisfaire
Modification d'un enregistrement	Cas du fichier texte à regénérer mysql> **SET AUTOCOMMIT=1;** # Utilisé pour une recréation rapide de la table mysql> **DELETE FROM animal where nom="Claws";** Cas d'une mise à jour en mode commande mysql> **UPDATE animal SET naissance = "1989-08-31" WHERE nom = "Bowser";**	DELETE : efface les « lignes » d'une table ; la table animal est alors vide On ne peut pas revenir en arriere par un « rollback » car le « commit =OK» est actif
Sélection de lignes	mysql> **SELECT * FROM animal WHERE nom = "Bowser";** mysql> SELECT * FROM animal WHERE UPPER(nom) = UPPER("Bowser"); mysql> SELECT * FROM animal WHERE nom = "Bowser";	Comparaison de chaine : casse sensitive !
AND **OR**	mysql> **SELECT * FROM animal WHERE espece = "chien" AND sexe = "f";** mysql> **SELECT * FROM animal WHERE espece = "serpent" OR espece = "oiseau";** mysql> **SELECT * FROM animal WHERE (espece = "chat" AND sexe = "m")** **-> OR (espece = "chien" AND sexe = "f");**	Plusieurs conditions "**AND**" / « OR »
Sélectionner des colonnes **DISTINCT**	mysql> **SELECT nom, naissance FROM animal;** mysql> **SELECT DISTINCT maitre FROM animal;** mysql> **SELECT nom, espece, naissance FROM animal** **-> WHERE espece = "chien" OR espece = "chat";**	
ORDER BY	mysql> **SELECT nom, naissance FROM animal ORDER BY naissance;** mysql> **SELECT nom, naissance FROM animal ORDER BY naissance DESC;** mysql> **SELECT nom, espece, naissance FROM animal ORDER BY type, naissance DESC;**	
LIKE	Pour trouver les noms commençant par la lettre 'b' : mysql> **SELECT * FROM animal WHERE nom LIKE "b%";** Pour trouver les noms finissant par 'fy' : mysql> **SELECT * FROM animal WHERE nom LIKE "%fy";** Pour trouver les noms contenant le caractères 'w' :	Lorsque vous testez une recherche avec ce type de modèle, utilisez les opérateurs REGEXP et NOT REGEXP (ou RLIKE et NOT RLIKE qui sont des synonymes).

<table>
<tr>
<td></td>
<td>
mysql> SELECT * FROM animal WHERE nom LIKE "%w%";
Pour trouver les noms contenant exactement 5 caractères, utilisez le caractère de recherche '_' :
mysql> SELECT * FROM animal WHERE nom LIKE "_____";
Pour trouver les noms qui commencent par la lettre 'b' , utilisez '^' pour trouver le début du nom :
mysql> SELECT * FROM animal WHERE nom REGEXP "^b";
Pour trouver les noms finissant par 'fy' , utilisez '$' pour trouver la fin du nom :
mysql> SELECT * FROM animal WHERE nom REGEXP "fy$";
Pour trouver les noms contenant la lettre 'w' minuscule ou majuscule, utilisez la requête suivante :
mysql> SELECT * FROM animal WHERE nom REGEXP "w";

Le caractère '.' trouve n'importe quel caractère.
Une classe de caractères '[...]' trouve n'importe quel caractère contenu entre les crochets. Par exemple, la classe de caractères '[abc]' trouve le caractère 'a' , 'b' , ou 'c' . Pour définir un intervalle de caractères, utilisez un trait d'union. La classe de caractères '[a-z]' trouvera n'importe quel caractère minuscule, tout comme la classe '[0-9]' trouvera n'importe quel nombre.
Le caractère '*' trouvera aucune ou plus d'instances du caractère qui le précède. Par exemple, 'x*' trouvera n'importe quel nombre de fois le caractère 'x' , '[0-9]*' trouvera n'importe quel nombre, et '.*' trouvera n'importe quel nombre de fois n'importe quel caractère.
Le modèle est trouvé s'il se produit n'importe où dans la valeur testée. (Les modèles SQL ne sont trouvés que s'ils sont présents en valeur entière.)

Pour ancrer un modèle de sorte qu'il soit trouvé au début ou à la fin de valeur testée, utilisez '^' au début ou bien '$' à la fin du modèle
</td>
<td>Quelques caractéristiques des expressions régulières étendues sont :</td>
</tr>
<tr>
<td>COUNT()</td>
<td>
mysql> SELECT COUNT(*) FROM animal; (compte les lignes)
trouver combien d'animal possède chaque propriétaire :
mysql> SELECT maitre, COUNT(*) FROM animal GROUP BY maitre;
mysql> SELECT espece, COUNT(*) FROM animal GROUP BY espece;

mysql> SELECT espece, sexe, COUNT(*) FROM animal
-> WHERE espece = "chien" OR espece = "chat"
-> GROUP BY espece, sexe;
</td>
<td></td>
</tr>
</table>

Date et calculs **YEAR()** **RIGHT()** **CURRENT_DATE()**	mysql> **SELECT nom, naissance, CURRENT_DATE,** **-> (YEAR(CURRENT_DATE)-YEAR(naissance))** **-> - (RIGHT(CURRENT_DATE,5)<RIGHT(naissance,5))** **-> AS age** **-> FROM animal;** mysql> **SELECT nom, naissance, CURRENT_DATE,** **-> (YEAR(CURRENT_DATE)-YEAR(naissance))** **-> - (RIGHT(CURRENT_DATE,5)<RIGHT(naissance,5))** **-> AS age** **-> FROM animal ORDER BY age;**	**YEAR()** extrait l'année de la date et **RIGHT()** extrait les 5 caractères les plus à droite de la date qui représentent MM-DD (année civile). La partie de l'expression qui compare les valeurs de MM-DD évalue à 1 ou à 0, qui ajustent la différence d'année à la baisse, si CURRENT_DATE se produit plus au début de l'année que la naissance
AUTO JOINTURE	mysql> **SELECT p1.nom, p1.sexe, p2.nom, p2.sexe, p1.espece** **-> FROM animal AS p1, animal AS p2** **-> WHERE p1.espece = p2.espece AND p1.sexe = "f" AND p2.sexe = "m";**	si vous voulez comparer des enregistrements dans une table avec d'autres enregistrements de la même table. Par exemple, pour trouver des paires multiples parmi vos animal, vous pouvez joindre la table animal sur elle-même pour trouver les paires mâles / femelles par rapport à l'espèce

ATTENTION !

Une autre base avec d'autres tables .....-> créer l'environnement

BASE AERIENNE

Rappeler la structure des tables, leurs attributs, les cles primaires voire les index.
Nb : une adresse se résume à la ville.
Nb : prevoir un nom de pilote « concorde », certains pilotes n'ont pas de telephone
Villes : Nice, Paris, Lille, Bordeaux
Capacites des avions de 50 à 700
Type d'avion : airbus, concorde, boeing, un_coucou
Avion : A300, B727, A320 etc

Ecrire les requêtes :

Q1 : liste de tous les pilotes . le listing comportera les champs : num pilote, nom, adresse, salaire
Q2 : réécrire la requête pour obtenir des entetes de colonne : No pilote, Nom pilote, Salaire, Adresse
Q3 : réécrire la requete avec un alias de la table pilote : p
Q4 : calculer le salaire annuel des pilotes, le lister pour chaque pilote
Q5 : calculer la somme des salaires des pilotes
Q6 : donner tous les types d'avion de la compagnie
Q7 : donner les numeros d'avions et type d'avion de capacité sup à 300
Q8 : donner les noms de pilotes habitants Paris ou Nice
Q9 : quels pilotes ont un 't' dans leur nom en 3eme position?
Q10 : Quels sont les vols au depart de Nice, Paris ou Bordeaux ?
Q11 : donner les noms des pilotes ayant un a et un e dans leur nom ?
Q12 : donner les noms et no tel des pilotes qui ont un No de telephone
Q13 : donner le salaire le + eleve et l'afficher sous la forme : <valeur du salaire max> « MAX SALAIRE »
Q14 : quels sont les noms, adresse et salaire des pilotes triés par ordre croissant sur l adresse et pour une meme adresse ordre decroissant de salaire
Q15 : donner les paires de pilotes habitant la meme ville (auto-jointure ! + alias) (ne pas avoir le meme nom dans le couple)
Q16 : donner tous les noms de pilote qui ont un nom d'avion
Q17 : Donner les noms de pilotes qui conduisent un A300 ou un B727
Q18 : donner les noms des pilotes , et pour ceux qui assurent un service, afficher les no de vols (jointure)(+)
Q19 : quels sont les pilotes (avec le nombre de vols) parmie les pilotes 1 à 3 qui assument au moins 2 vols
Q20 : mettre à jour le salaire du pilote no 1 à 3200ff : valider
Q21 : supprimer le pilote no 3
Q22 : ajouter le champs « age » à la table des pilotes

ALLER PLUS LOIN

Quelques remarques

CLE ETRANGERE

Depuis la version 3.23.44 de MySQL, les tables `InnoDB` supportent les contraintes des clefs étrangères. Tables `InnoDB`. Consultez aussi Clés étrangères .

Actuellement, vous n'avez pas besoin de clefs étrangères pour réaliser des jointures entre les tables. La seule chose que MySQL ne fait pas encore (avec les types autres que `InnoDB`), est `CHECK` pour s'assurer que que la clef que vous utilisez existe bien dans la ou les tables que vous référencez et il n'efface pas automatiquement les lignes d'une table avec une définition de clef étrangère. Si vous utilisez vos clefs comme une clef normale, tout marchera parfaitement :

```
CREATE TABLE person (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name CHAR(60) NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE shirt (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    style ENUM('t-shirt', 'polo', 'dress') NOT
NULL,
    color ENUM('red', 'blue', 'orange', 'white',
'black') NOT NULL,
    owner SMALLINT UNSIGNED NOT NULL REFERENCES
person(id),
    PRIMARY KEY (id)
);
 INSERT INTO person VALUES (NULL, 'Antonio Paz');

INSERT INTO shirt VALUES
(NULL, 'polo', 'blue', LAST_INSERT_ID()),
(NULL, 'dress', 'white', LAST_INSERT_ID()),
(NULL, 't-shirt', 'blue', LAST_INSERT_ID());


<P>
INSERT INTO person VALUES (NULL, 'Lilliana
Angelovska');

INSERT INTO shirt VALUES
(NULL, 'dress', 'orange', LAST_INSERT_ID()),
(NULL, 'polo', 'red', LAST_INSERT_ID()),
(NULL, 'dress', 'blue', LAST_INSERT_ID()),
(NULL, 't-shirt', 'white', LAST_INSERT_ID());

</P>

SELECT * FROM person;
+----+--------------------+
| id | name               |
+----+--------------------+
|  1 | Antonio Paz        |
|  2 | Lilliana Angelovska |
+----+--------------------+

SELECT * FROM shirt;
+----+---------+--------+-------+
| id | style   | color  | owner |
+----+---------+--------+-------+
|  1 | polo    | blue   |     1 |
|  2 | dress   | white  |     1 |
```

```
|  3 | t-shirt | blue   |     1 |
|  4 | dress   | orange |     2 |
|  5 | polo    | red    |     2 |
|  6 | dress   | blue   |     2 |
|  7 | t-shirt | white  |     2 |
+----+---------+--------+-------+
 SELECT s.* FROM person p, shirt s
 WHERE p.name LIKE 'Lilliana%'
   AND s.owner = p.id
   AND s.color <> 'white';

+----+-------+--------+-------+
| id | style | color  | owner |
+----+-------+--------+-------+
|  4 | dress | orange |     2 |
|  5 | polo  | red    |     2 |
|  6 | dress | blue   |     2 |
+----+-------+--------+-------+
```

Ajouter de nouveaux utilisateurs à MySQL

Vous pouvez ajouter des utilisateurs de deux façons différentes : en utilisant la commande GRANT ou en manipulant la table des droits de MySQL directement. La méthode préférée consiste à utiliser la commande GRANT , car elle est plus concise et qu'il y a moins de risques d'erreur. Syntaxe de GRANT et REVOKE .

Il y a aussi beaucoup de programmes contribués comme phpmyadmin qui peuvent être utilisés pour créer et administrer les utilisateurs.

Les exemples suivants montrent comment utiliser le client mysql pour créer de nouveaux utilisateurs. Ces exemples supposent que les privilèges sont attribués en accord avec les valeurs par défaut discutées dans la section précédente. Cela signifie que pour effectuer des changements, vous devez être sur la même machine où mysqld tourne, vous devez vous connecter en tant qu'utilisateur MySQL root , et l'utilisateur root doit avoir le droit INSERT sur la base mysql et le droit d'administration RELOAD . Si vous avez changé le mot de passe de l'utilisateur root , vous devez le spécifier dans les commandes mysql ci-dessous.

Vous pouvez ajouter de nouveaux utilisateurs en utilisant des commandes GRANT :

```
shell> mysql --user=root mysql
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@localhost
    ->     IDENTIFIED BY 'un_mot_de_passe' WITH GRANT OPTION;
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@"%"
    ->     IDENTIFIED BY 'un_mot_de_passe' WITH GRANT OPTION;
mysql> GRANT RELOAD,PROCESS ON *.* TO admin@localhost;
mysql> GRANT USAGE ON *.* TO dummy@localhost;
```

Ces commandes GRANT ajoutent trois nouveaux utilisateurs :

monty
Un super-utilisateur qui peut se connecter au serveur d'où il veut, mais qui doit utiliser le mot de passe 'un_mot_de_passe' pour le faire. Notez que nous devons exécuter une commande GRANT pour monty@localhost et monty@"%" . Si nous n'ajoutons pas l'entrée avec localhost , l'entrée concernant l'utilisateur anonyme pour localhost qui est créée par mysql_install_db prendra précédence lors de la connexion à partir de l'hôte local, car elle a une entrée plus spécifique pour la valeur du champ Host et de plus, elle vient en premier dans l'ordre de tri de la table user .

admin
Un utilisateur qui peut se connecter depuis localhost sans mot de passe et qui a les droits administratifs RELOAD et PROCESS . Cela permet à cet utilisateur d'exécuter les commandes mysqladmin reload , mysqladmin refresh , et mysqladmin flush-* , ainsi que mysqladmin processlist . Aucun droit lié aux bases de données n'est donné. (Ils peuvent l'êetre plus tard en utilisant d'autres instructions GRANT .)

dummy
Un utilisateur qui peut se connecter sans mot de passe, mais seulement à partir de l'hôte local. Les droits globaux sont tous à 'N' -le type de droit USAGE vous permet de créer un utilisateur démuni de privilèges. Il est supposé que vous lui assignerez les droits spécifiques aux bases de données plus tard.

Vous pouvez ajouter les mêmes droits d'accès aux utilisateurs en utilisant directement des requêtes INSERT puis en demandant au serveur de recharger les tables de droits :

```
shell> mysql --user=root mysql
mysql> INSERT INTO user
VALUES('localhost','monty',PASSWORD('un_mot_de_passe'),
    -
>          'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y
');
mysql> INSERT INTO user
VALUES('%','monty',PASSWORD('un_mot_de_passe'),
    -
>          'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y
');
mysql> INSERT INTO user SET Host='localhost',User='admin',
    ->           Reload_priv='Y', Process_priv='Y';
mysql> INSERT INTO user (Host,User,Password)
    ->                  VALUES('localhost','dummy','');
mysql> FLUSH PRIVILEGES;
```

Selon votre version de MySQL, vous pouvez avoir un nombre différent de valeurs 'Y'

	GRANT ... WITH MAX_QUERIES_PER_HOUR N1 MAX_UPDATES_PER_HOUR N2 MAX_CONNECTIONS_PER_HOUR N3;	

SELECT imbriqués : Utiliser tables temporaires

```
SELECT article, dealer, price
FROM   shop s1
WHERE  price=(SELECT MAX(s2.price)
              FROM shop s2
              WHERE s1.article = s2.article);
```

En MySQL il vaut mieux le faire en plusieurs étapes :

- Récupérer la liste de (article, plusgrandprix).
- Pour chaque article, récupérer la ligne qui a le plus grand prix stocké.

Cela se fait facilement avec une table temporaire :

```
CREATE TEMPORARY TABLE tmp (
        article INT(4) UNSIGNED ZEROFILL DEFAULT
'0000' NOT NULL,
        price   DOUBLE(16,2)             DEFAULT
'0.00' NOT NULL);

LOCK TABLES shop read;

INSERT INTO tmp SELECT article, MAX(price) FROM shop
GROUP BY article;

SELECT shop.article, dealer, shop.price FROM shop, tmp
WHERE shop.article=tmp.article AND
shop.price=tmp.price;

UNLOCK TABLES;

DROP TABLE tmp;
```