

INSTITUT D'ENSEIGNEMENT TECHNIQUE  
DE MECANIQUE ET D'ELECTRICITE  
MARGUERITE MASSART

**BASES  
DE  
DONNEES**

Professeur : BANZIRA Pascal

1<sup>ère</sup> PARTIE :

**Introduction à  
MySQL / PhPMyAdmin**

# Introduction

- MySQL dérive directement de SQL (Structured Query Language)
- L'outil phpMyAdmin est développé en PHP et offre une interface pour l'administration des base de données
- phpMyAdmin est téléchargeable ici : <http://phpmyadmin.sourceforge.net>
- cet outil permet de :
  - créer de nouvelles bases
  - créer/modifier/supprimer des tables
  - afficher/ajouter/modifier/supprimer des tuples dans des tables
  - effectuer des sauvegardes de la structure et/ou des donnés
  - effectuer des requêtes
  - gérer les privileges des utilisateurs

## Liens intéressants (MySQL)

- La référence MySQL (anglais): <http://www.mysql.org>
- Le manuel MySQL :  
<http://dev.mysql.com/doc/refman/5.7/en>
- Des cours et articles intéressants :
  - <http://www.developpez.com>
  - <http://cyberzoide.developpez.com/php4/mysql/>

# Types des attributs MySQL

- Nombre entier signé ou non
- Nombre à virgule
- Chaîne de caractères
- Date et heure
- Enumeration
- Ensemble

# Entiers

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

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

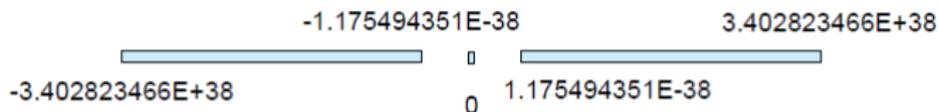
**UNSIGNED** permet d'avoir un type non signé.

**ZEROFILL** : remplissage des zéros non significatifs.

# 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
<b>FLOAT</b>	-3.402823466E+38 -1.175494351E-38	1.175494351E-38 3.402823466E+38
<b>DOUBLE*</b>	-1.7976931348623157E+308 -2.2250738585072014E-308	2.2250738585072014E-308 1.7976931348623157E+308

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

# Chaînes

nom	longueur
<b>CHAR(M)</b>	Chaîne de taille fixée à M, où $1 < M < 255$ , complétée avec des espaces si nécessaire.
<b>CHAR(M) BINARY</b>	Idem, mais insensible à la casse lors des tris et recherches.
<b>VARCHAR(M)</b>	Chaîne de taille variable, de taille maximum M, où $1 < M < 255$ , complétée avec des espaces si nécessaire.
<b>VARCHAR(M) BINARY</b>	Idem, mais insensible à la casse lors des tris et recherches.
<b>TINYTEXT</b>	Longueur maximale de 255 caractères.
<b>TEXT</b>	Longueur maximale de 65535 caractères.
<b>MEDIUMTEXT</b>	Longueur maximale de 16777215 caractères.
<b>LONGTEXT</b>	Longueur maximale de 4294967295 caractères.
<b>DECIMAL(M,D)*</b>	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**.

# 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 <b>TIMESTAMP</b> .
TIME	Heure au format HH:MM:SS.
YEAR	Année au format AAAA.

nom	description
TIMESTAMP(2)	AA
TIMESTAMP(4)	AAMM
TIMESTAMP(6)	AAMMJJ
TIMESTAMP(8)	AAAAMMJJ
...	...

Si attribut de type **TIMESTAMP**

= *NULL* :

- prend la date et heure de l'insertion.
- suit le format indiqué

# Interface graphique (phpMyAdmin)

- développé en PHP
- téléchargeable :  
`http://phpmyadmin.sourceforge.net`
- `http://euterpe.unice.fr/phpmyadmin`
- version installée (5.x)

# Interface graphique (phpMyAdmin) - 1

euterpe.unice.fr / localhost | phpMyAdmin 2.6.2-Debian-3sarge1 - Mozilla Firefox

http://euterpe.unice.fr/phpmyadmin/index.php?lang=fr-utf8server=18collation\_...

Bienvenue à phpMyAdmin 2.6.2-Debian-3sarge1

MySQL 4.1.11-Debian\_4sarge4-log sur le serveur localhost - utilisateur : lahire@localhost

phpMyAdmin

Base de données:  
(Bases de données) ...

Choisissez une base de données

Accès

MySQL

Créer une base de données

Interassement

Créer

Afficher l'état du serveur

Afficher les variables du serveur

Afficher les processus

Jeu de caractères et interassement

Moteurs de stockage

Recharger MySQL

Privilèges

Log binaire

Basas de données

Exporter

Modifier le mot de passe

Quitter

Création

phpMyAdmin

Language: French (fr-utf8)

Jeu de caractères pour MySQL: UTF-8 Unicode (utf8)

Interassement pour la connection MySQL: utf8\_general\_ci

Thème / Style: Original

Documentation de phpMyAdmin

Afficher les informations relatives à PHP

Site officiel de phpMyAdmin

[ChangeLog](#) [CVS](#) [Lists](#)

http://euterpe.unice.fr/phpmyadmin/changelog.php

# Interface graphique (phpMyAdmin) - 2

The screenshot shows the phpMyAdmin web interface in a Mozilla Firefox browser window. The address bar shows the URL: `http://euterpe.unice.fr/phpmyadmin/index.php?lang=fr-utf-8&server=18&collation=co`. The interface includes a top navigation bar with options like 'Structure', 'SQL', 'Exporter', 'Rechercher', 'Requête', 'Opérations', and 'Supprimer'. The main content area displays the 'Base de données lahire06-07' and a 'requête SQL' section with the text 'CREATE DATABASE `lahire06-07` ;'. Below this, a message states 'Aucune table n'a été trouvée dans cette base.' and a section for 'Créer une nouvelle table sur la base lahire06-07:' with a form for 'Nom:' (containing 'client') and 'Champs:' (containing '4'). A sidebar on the left shows 'Base de données:' and 'lahire06-07'. Three green callout boxes with arrows point to specific elements: 'Autres bases' points to the database selection dropdown, 'Requête SQL' points to the SQL query text, and 'Création de table' points to the table creation form.

Autres bases

Requête SQL

Création de table

# Interface graphique (phpMyAdmin) - 3

The screenshot shows the phpMyAdmin interface for a database named 'lahire06\_07' and a table named 'Client'. The table structure is displayed with the following columns:

Champ	Type	Taille/Valeurs*	Interclassement	Attributs	Null	Défaut**	Extra
	VARCHAR				not null		
	VARCHAR				not null		
	VARCHAR				not null		
	VARCHAR				not null		

Annotations in green boxes point to specific features:

- 4 champs**: Points to the four columns in the table structure.
- Type possible (pas de domaine)**: Points to the 'Type' column, which shows 'VARCHAR' for all fields.
- Signé?**: Points to the 'Interclassement' column, which is currently empty.
- Auto?**: Points to the 'Extra' column, which is currently empty.
- Clé primaire? Index? Unique?**: Points to the 'Extra' column, which is currently empty.

Other visible elements include the 'InnoDB' storage engine, the 'Saisie obligatoire?' checkbox, and the 'Auto?' checkbox. The interface also shows a 'Sauvegarder' button and a 'Commentaires sur la table' section.

## Quelques mots sur InnoDB

- Moteur de tables (licence GNU GPL)
- gestionnaire de tables transactionnelles (verrouillage de lignes)
- maximisation de performances (grands volumes de données)
- support clé étrangère
- stockage des tables et index : espace de tables (un ou plusieurs fichiers). MyISAM : un fichier par table

# Interface graphique (phpMyAdmin) - 4

Terminé

Champ	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
<input type="checkbox"/> numero	smallint(5)		UNSIGNED	Non		auto_increment	[Edit] [Delete] [Refresh] [Add] [Drop]
<input type="checkbox"/> nom	varchar(20)	latin1_swedish_ci		Non			[Edit] [Delete] [Refresh] [Add] [Drop]
<input type="checkbox"/> adresse	varchar(30)	latin1_swedish_ci		Non			[Edit] [Delete] [Refresh] [Add] [Drop]
<input checked="" type="checkbox"/> num-telephone	varchar(10)	latin1_swedish_ci		Oui	NULL		[Edit] [Delete] [Refresh] [Add] [Drop]

Tout cocher / Tout décocher Pour la sélection : [Edit] [Delete] [Refresh] [Add] [Drop]

Version imprimable Suggérer des optimisations quant à la structure de la table

Ajouter 1 champ(s) En fin de Table En début de Table Après numero Exécute

Index:				
Nom de la clé	Type	Cardinalité	Action	Champ
PRIMARY	PRIMARY	0	[Edit] [Delete]	numero
nom	FULLTEXT	aucune	[Edit] [Delete]	nom

Créer une clef sur 1 colonne(s) Exécute

Espace utilisé:	
Type	Espace
Données	0 Octets
Index	1 024 Octets
Total	1 024 Octets

Statistiques:	
Information	Valeur
format	dynamique
Interclassement	latin1_swedish_ci
Enregistrements	0
Suivant Autoindex	
création	Dimanche 14 Janvier 2007 à 17:18
dernière modification	Dimanche 14 Janvier 2007 à 17:18

Raccourcis Clé primaire?

Modifier supprimer

# Interface graphique (phpMyAdmin) – 4 bis

The screenshot shows the phpMyAdmin interface for a MySQL database named 'lahire06-07'. The table 'vente' is selected, and its structure is displayed. A red box highlights the table's description: 'Les ventes de l'entreprise; (noDB free: 4095 KB; (ref-client) REFER (lahire06-07/client ('numero'); (ref-produit) REFER lahire06-07/produit ('numero'))'. A green box points to the 'Index' section, which lists the primary key 'numero' and foreign keys 'ref-produit' and 'ref-client'. Another green box points to the 'Statistiques' section, showing table statistics. The interface includes navigation buttons like 'Structure', 'Afficher', 'SQL', 'Rechercher', 'Insérer', 'Exporter', 'Opérations', 'Vider', and 'Supprimer'.

Severeur: localhost > Base de données: lahire06-07 > Table: vente

Structure Afficher SQL Rechercher Insérer Exporter Opérations Vider Supprimer

Les ventes de l'entreprise; (noDB free: 4095 KB; (ref-client) REFER (lahire06-07/client ('numero'); (ref-produit) REFER lahire06-07/produit ('numero'))

Champ	Type	Interclassement	Attributs Null	Défaut	Extra	Action				
<input type="checkbox"/>	numero	int(10)	unsigned	Non	auto_increment					
<input type="checkbox"/>	ref-produit	int(10)	unsigned	Non	0					
<input type="checkbox"/>	ref-client	int(10)	unsigned	Non	0					
<input type="checkbox"/>	date	date		Non	0000-00-00					

Tout cocher / Tout décocher Pour la sélection:

Version imprimable Gestion des relations Suggérer des optimisations pendant à la str

Ajouter 1 Champ(s) En fin de Table En début de Table Après numero

Index:					Espace utilisé:		Statistiques:	
Nom de la clé	Type	Cardinalité	Action	Champ	Type	Espace	Information	Valeur
PRIMARY	PRIMARY	0		numero	Données	16 384 Octets	format	fixe
ref-produit	INDEX	0		ref-produit	Index	32 768 Octets	Interclassement	latin1_swedish_ci
ref-client	INDEX	0		ref-client	Total	49 152 Octets	Suivant Autoindex	1

Créer une clef sur 1 colonne(s) Exécuter

Exécuter une ou des requêtes sur la base lahire06-07 Champs:

Terminé

Index pour Clés étrangères

Pour clés étrangères

# Intégrité référentielle

## ON UPDATE et ON DELETE

Si l'utilisateur tente de supprimer une ligne d'une table parente, alors qu'une ou plusieurs lignes existent dans une table enfant correspondante à la première, il y a plusieurs possibilités :

- CASCADE efface la ligne de la table parente et supprime automatiquement les lignes correspondants dans la table enfante
- SET NULL supprime la ligne de la table parente et met la (ou les) valeur(s) de la clé étrangère à NULL
- SET DEFAULT supprime la ligne de la table parente et met tous les composants de la clé étrangère à leur valeur par défaut dans la table enfante
- NO ACTION rejette l'opération de suppression dans la table parente

# Interface graphique (phpMyAdmin) - 5

The screenshot shows the phpMyAdmin web interface. The browser address bar indicates the URL is `localhost/lahire06-07`. The interface includes a sidebar on the left with the phpMyAdmin logo and a tree view of the database structure. The main area displays a table list for the database `lahire06-07`. Below the table list, there are options to create a new table, including fields for name and fields, and an 'Exécuter' button. The interface is annotated with green callout boxes and arrows pointing to various elements.

**Niveau de détail**

**Afficher, Rechercher, Insérer, vider**

Table	Action	Enregistrements	Type	Interclassement	Taille	Perte
<input type="checkbox"/> client		0	MYISAM	latin1_swedish_ci	1,0 Ko	-
<input type="checkbox"/> produit		0	MYISAM	latin1_swedish_ci	1,0 Ko	-
<input type="checkbox"/> vente		0	MYISAM	latin1_swedish_ci	1,0 Ko	-
3 table(s) Somme		0	-	latin1_swedish_ci	3,0 Ko	0 Octets

**Tables de la base**

**Ajouter, Modifier, Supprimer**

# Interface graphique (phpMyAdmin) - 6

The screenshot shows the phpMyAdmin interface for a MySQL database. The browser address bar shows 'euterpe.unice.fr / localhost / lahire06-07'. The interface includes a sidebar with the phpMyAdmin logo and a list of databases, with 'lahire06-07' selected. The main area displays the 'table des clients' with a table view. A green box labeled 'Niveau de détail' points to the table name 'clients'. Another green box labeled 'Equivalent en SQL' points to the SQL query: 'SELECT \* FROM client LIMIT 30'. A third green box labeled 'Paramétrage affichage' points to the 'Afficher' dropdown set to '30' and the 'en mode' dropdown set to 'horizontal'. A fourth green box labeled 'Contenu' points to the table data row: '1 Durand Nico 0610455412'. A fifth green box labeled 'Modification Suppression' points to the table's header row: 'numero nom adresse aux-telephone'. The table has two rows of data. The bottom of the interface shows a status bar with 'Terminé'.

Niveau de détail

Equivalent en SQL

Paramétrage affichage

Contenu

Modification Suppression

# Interface graphique (phpMyAdmin) - 7

The screenshot shows the phpMyAdmin interface in a Mozilla Firefox browser window. The address bar shows the URL: `http://europe.unice.fr/phpmyadmin/index.php?lang=fr&utf8=server=1&collation_connection=utf8`. The page title is "europe.unice.fr / localhost / lahire06-07 | phpMyAdmin 2.6.2-Debian-ossange1 - Mozilla Firefox".

The main content area is titled "Schéma et données de la base" and shows the "Exporter" (Export) configuration screen for the "lahire06-07" database. The "Export" section on the left lists the tables: "client", "produit", and "vente". The "Options SQL" section on the right contains various checkboxes and options for the export process.

Annotations with green boxes and arrows point to specific elements:

- Niveau de détail**: Points to the "Options SQL" section.
- Configuration**: Points to the "Options SQL" section, specifically the "Structure" and "Données" options.
- Exportation**: Points to the "Options SQL" section, specifically the "Type d'exportation" dropdown.

The "Options SQL" section includes the following options:

- Commentaires mis en en-tête (n sépare les lignes):
- Utiliser le mode transactionnel:
- Désactiver la vérification des clés étrangères:
- Structure:**
  - Inclure des énoncés "DROP TABLE":
  - Ajouter "IF NOT EXISTS":
  - Inclure la valeur courante de l'AUTO\_INCREMENT:
  - Préciser les noms des tables et des champs par des \*\*\*:
- Inclure sous forme de commentaires**
  - Dates de création/modification/vérification:
- Compatibilité de l'assemblage: NONE
- Données:**
  - Insertions complètes:
  - Insertions étendues:
  - Insertions avec délai (DELAYED):
  - Ignorer les erreurs de doublons (INSERT IGNORE):
  - Encoder les champs binaires en hexadécimal:
- Type d'exportation: INSERT

# Interface graphique (phpMyAdmin) - 8

The screenshot shows the phpMyAdmin interface for a database named 'lahiro06-07'. The interface includes a sidebar with navigation options, a main toolbar with actions like 'Structure', 'Afficher', 'SQL', 'Rechercher', 'Insérer', 'Exporter', 'Opérations', 'Vider', and 'Supprimer'. The main content area displays the 'table des clients' with options to 'Choisir les champs à afficher' and 'Ordre d'affichage'. A search section is also visible.

**Champs**: Points to the 'Choisir les champs à afficher' section.

**Niveau de détail**: Points to the 'Afficher' button in the toolbar.

**Affichage**: Points to the 'Ordre d'affichage' dropdown menu.

**Critère**: Points to the search criteria table.

Champ	Type	Interclassement	Opérateur	Valeur
numero	smallint(5)	<input checked="" type="checkbox"/>	<input type="text"/>	<input type="text"/>
nom	varchar(20)	latin1_swedish_ci	LIKE	<input checked="" type="checkbox"/>
adresse	varchar(30)	latin1_swedish_ci	LIKE	<input checked="" type="checkbox"/>
num.telephone	varchar(10)	latin1_swedish_ci	LIKE	<input checked="" type="checkbox"/>

# Interface graphique (phpMyAdmin) – 8b

The screenshot shows the phpMyAdmin interface for a database named 'lahire06-07' and a table named 'Enseignant'. The 'Rechercher' (Search) tab is selected. The interface includes a sidebar with the database structure, a main area with search options, and a search criteria table.

**Recherche mono table**

**1...tous**

**ET**

**1**

Server: localhost → Base de données: lahire06-07 → Table: Enseignant

Structure Afficher SQL Rechercher Insérer Exporter Opérations Vider Supprimer

Relation Enseignant: InnoDB free: 3072 kB

Choisir les champs à afficher (au moins un)

- numero
- nom
- prenom
- age
- nb\_heures
- ville

DISTINCT

Nombre d'enregistrements par page : 30

Ordre d'affichage :  Croissant  Décroissant

Critères de recherche (pour l'énoncé "where"):  Exécuter

Ou Recherche par valeur (passepartout: "%")

Champ	Type	Interclassement	Opérateur	Valeur
numero	int(10)		>	1001
nom	varchar(20)	latin1_swedish_ci	LIKE	
prenom	varchar(20)	latin1_swedish_ci	LIKE	Philippe
age	tinyint(3)		=	
nb_heures	tinyint(3)		=	
ville	varchar(40)	latin1_swedish_ci	LIKE	

Exécuter

# Interface graphique (phpMyAdmin) - 9

The screenshot shows the phpMyAdmin web interface. The browser address bar indicates the URL is `http://euterpe.unice.fr/phpmyadmin/index.php?lang=fr-utf8&server=1&connection=...`. The interface is for a server named `localhost` and a database named `lahire06-07`. The main area displays a table structure for the `client` table, with columns `nom`, `adresse`, and `num-telephone`. The `Critère` (Criteria) field is set to `=prospér`. The `Requête SQL` (SQL Query) field contains the following query:

```
SELECT 'client'.nom,
       'client'.adresse,
       'client'.num-telephone
FROM client
WHERE ('client'.nom)='prospér'
```

Annotations with green boxes and arrows point to various elements:

- Niveau de détail**: Points to the browser address bar.
- Champs impliqués**: Points to the column selection dropdowns in the table structure view.
- Critère**: Points to the criteria input field.
- Requête SQL**: Points to the SQL query text area.
- Tables**: Points to the table selection dropdown in the 'Utiliser les tables' section.

# Interface graphique (phpMyAdmin) - 10

The screenshot shows the phpMyAdmin interface for a database named 'lahiro06-07'. The interface is annotated with several callouts and numbers:

- 1**: Points to the left sidebar menu.
- 2**: Points to the 'Mise-à-jour de la requête' button at the bottom of the SQL editor.
- 3**: Points to the 'Afficher' (Display) section of the query editor, where checkboxes for 'UE: libelle' and 'UE: responsable' are checked.
- 4**: Points to the 'Critère' (Criteria) section of the query editor.
- Sélection**: A green box pointing to the 'Où:' (Where) clause in the query editor.
- Niveau de détail**: A green box pointing to the 'UE: libelle' and 'UE: responsable' fields in the 'Afficher' section.
- Champs impliqués**: A green box pointing to the 'UE: libelle' and 'UE: responsable' fields in the 'Afficher' section.
- Requête SQL**: A green box pointing to the SQL query text in the editor.
- Tables**: A green box pointing to the 'Utiliser les tables:' (Use tables:) section on the left.

The SQL query displayed in the editor is:

```
SELECT 'Enseignant'.nom,  
'UE'.libelle  
FROM Enseignant UE  
WHERE (( 'Enseignant'.nom  
='lahie') AND ('UE'.responsable' =  
Enseignant.surnom))  
(( 'Enseignant'.nom = 'menez') AND  
( 'UE'.responsable' =
```

# Interface graphique (phpMyAdmin) - 11

The screenshot shows the phpMyAdmin interface for a MySQL database. The browser address bar shows the URL: `http://eutsrpe.unice.fr/phpmyadmin/index.php?lang=fr&utf8server=1&collation_connection=utf8mb4_unicode_ci`. The interface includes a sidebar with the phpMyAdmin logo and a navigation menu for the database 'lahire06-07'. The main content area displays the 'Enseignant' table structure and a query result.

**Niveau de détail**: A green box pointing to the top navigation bar containing buttons like 'Structure', 'Afficher', 'SQL', 'Rechercher', 'Insérer', 'Exporter', 'Opérations', 'Vider', and 'Supprimer'.

**Champs impliqués**: A green box pointing to the 'WHERE' clause of the SQL query: `WHERE ( enseignant.nom = 'lahire' ) AND ( 'E' = 'responsable' = Enseignant.nom )`.

**Requête SQL**: A green box pointing to the entire SQL query text: `SELECT Enseignant.nom, UE.libelle FROM Enseignant UE WHERE ( enseignant.nom = 'lahire' ) AND ( 'E' = 'responsable' = Enseignant.nom ) LIMIT 30`.

**Résultat**: A green box pointing to the table header 'nom: libelle' and the first row of data 'Lahire Base de Données'.

# Interface graphique (phpMyAdmin) - 12

The screenshot shows the phpMyAdmin web interface. The browser address bar indicates the URL `http://euterpe.unice.fr/phpmyadmin/index.php?lang=fr-utf8&server=18collibon_connection`. The interface includes a navigation menu on the left with the database `lahire06-07` selected. The main content area displays the results of a SQL query. A green box labeled "Niveau de détail" points to the browser's address bar. Another green box labeled "Affichage" points to the "Afficher" controls (30 lines) and the "en mode" dropdown (horizontal). A third green box labeled "Requête SQL" points to the SQL query text: `SELECT client_nom, client_adresse, client_num_telephone FROM CLIENT WHERE (client_nom = 'prosp')`. A fourth green box labeled "Résultat" points to the table header `nom adresse num_telephone` and the single row of data: `Prosper Paris`. The interface also features buttons for "Structure", "SQL", "Exporter", "Rechercher", "Requête", "Opérations", and "Supprimer".

# Interface graphique (phpMyAdmin) - 13

The screenshot shows the phpMyAdmin interface for a database named 'lahire05-07'. The 'client' table is selected, and its structure is displayed. The table has the following columns:

Champ	Type	Fonction	Null	Valeur
numero	smallint(5) unsigned			
nom	varchar(20)	ASCII CHAR MDS SHA1 ENCRYPT RAND	<input checked="" type="checkbox"/>	
adresse	varchar(30)	LAST_INSERT_ID UNIX_TIMESTAMP		
num-telephone	varchar(10)	COUNT AVG SUM	<input checked="" type="checkbox"/>	

Annotations in the image:

- Champs**: Points to the column headers.
- Niveau de détail**: Points to the 'client' table name in the breadcrumb.
- insertion**: Points to the 'Insérer' button.
- Valeurs pour le tuple**: Points to the input fields for the 'numero' and 'nom' columns.
- Attribut calculé**: Points to the 'Fonction' dropdown menu.

At the bottom of the interface, there are buttons for 'Exécuter' and 'Régénérer les valeurs'.

# Résumé sur le modèle relationnel (1)

- Le système de gestion de base de données relationnelle est actuellement le logiciel de traitement de données le plus fréquemment utilisé
- Une **relation** mathématique est un sous-ensemble du produit cartésien de deux ensembles ou plus. En termes de base de données, une relation est n'importe quel sous-ensemble du produit cartésien des domaines des attributs
- Les relations sont représentées de manière physique par des **tables**, dont les lignes correspondent aux tuples individuels et les colonnes aux attributs.

## Résumé sur le modèle relationnel (2)

- Les propriétés d'une base de données sont les suivantes :
  - chaque cellule contient exactement une valeur atomique
  - les noms d'attributs sont distincts les uns des autres
  - l'ordre des attributs est immatériel
  - l'ordre des tuples est immatériel
  - il n'existe pas de tuples en double
- Dans un tuple, un **null** représente une valeur d'un attribut, inconnue à l'heure actuelle ou qui ne s'applique pas à ce tuple

## Résumé sur le modèle relationnel (3)

- Une **clé candidate** est un ensemble minimum d'attributs qu'identifie les tuples d'une relation de façon unique
- Une **clé primaire** est la clé candidate choisie pour servir à l'identification de tuples
- Une **clé étrangère** est un ensemble d'attributs au sein d'une relation qui constitue une clé candidate d'une autre relation
- l'**intégrité d'entité** établit que, dans une relation de base, aucun attribut qui fait partie de la clé primaire ne peut être nul
- l'**intégrité référentielle** établit que les valeurs d'une clé étrangère doivent correspondre à une valeur d'une clé candidate d'un tuple dans la relation de référence de la clé candidate ou être complètement nulles

**2ème PARTIE :**

**Le langage SQL**

# Introduction

## Objectifs de SQL

- Créer la structure de la base de données et de ses table
- Exécuter les tâches de base de la gestion des données, telle que l'insertion, la modification et la suppression de données des tables
- Effectuer des requêtes simples ou complexes

## Langage orienté transformation

## Origine

---

### Structured Query Language

- caractéristiques des langages déclaratifs
- origine : IBM, System R, milieu des années 70
- implémenté dans de nombreux SGBD

### Plusieurs versions :

- SQL1 initial : ANSI\* —1986
- SQL1 avec intégrité référentielle, ANSI —1989
- SQL2 ANSI —1992
- SQL3 ANSI — 1999 incorpore la notion d'objet

\* ANSI = American National Standard Institute

## Caractéristiques de SQL

---

- Fonctionnalités :
  - Définition des objets de la base de données (LDD)
  - Manipulation de données (LMD)
  - Contrôle des accès aux données
  - Gestion de transactions
- Utilisé par : DBA, développeurs, quelques utilisateurs



Parallèle avec mySQL

## Principales Instructions

---

- Définitions (LDD)  
*CREATE, DROP, ALTER*
- Mises à jour (LMD)  
*INSERT, UPDATE, DELETE*
- Interrogations (LMD)  
*SELECT*
- Contrôle d'accès aux données  
*GRANT, REVOKE*
- Gestion de transactions  
*COMMIT, ROLLBACK*



## Consultation des données

---

- Hypothèse:
  - un schéma de base de données est créé
  - Une base de données a été remplie
- La création a été faite par une interface QBE (mySQL)
- On expérimente la consultation avec SQL
- On expérimentera la création du schéma et le remplissage de la base avec SQL plus tard

- ➡ **Langage algébrique en SQL**
- ➡ **Requêtes mono et multi table(s)**

# Exemple pour les requêtes

---

## CLIENT

numéro	nom	adresse	numéro_téléphone
101	Durand	NICE	0493456743
108	Fabre	PARIS	NULL
110	Prosper	PARIS	NULL
125	Antonin	MARSEILLE	NULL

## PRODUIT

référence	marque	Prix HT
153	BMW	1000
589	PEUGEOT	1800
158	TOYOTA	1500

## VENTE

numéro	référence_produit	numéro_client	date
00102	153	101	12/10/04
00809	589	108	20/01/05
11005	158	108	15/03/05
12005	589	125	30/03/05

## Format des requêtes

- SELECT
  - FROM
  - WHERE
  - GROUP BY
  - HAVING
  - ORDER BY
- 
- FROM spécifie la table ou les tables à utiliser
  - WHERE filtre les lignes selon une condition donnée
  - GROUP BY forme des groupes de lignes de même valeur de colonne
  - HAVING filtre les groupes sujets à une certaine condition
  - SELECT spécifie les colonnes qui doivent apparaître dans les résultats
  - ORDER BY spécifie l'ordre d'apparition des données dans le résultat

## Requêtes simples (SELECT-FROM)

- Afficher le nom et l'adresse des clients

```
SELECT nom , adresse
```

```
FROM Client ;
```

- Afficher toutes les informations des clients

```
SELECT *
```

```
FROM Client ;
```

# Elimination des doublons

---

- Afficher l'adresse de tous les clients

```
SELECT adresse  
FROM Client ;
```

```
SELECT ALL adresse  
FROM Client;
```

- Afficher toutes les adresses existantes (sans doublons)

```
SELECT DISTINCT adresse  
FROM Client ;
```

Projection

Par défaut



Relationnel : On doit pouvoir distinguer chaque tuple !  
SQL : le résultat peut ne pas être une relation

# Sélection de colonne (clause WHERE)

## Les conditions fondamentales de recherche

- *comparaison*  
(salaire > 10000, ville = 'Paris')
- *étendue ou intervalle*  
(salaire BETWEEN 20000 and 30000)
- *appartenance à un ensemble*  
(couleur IN ('red', 'vert'))
- *correspondence à un masque*  
(adresse LIKE '%Montréal%')
- *nul*  
(adresse IS NULL)

# Opérateur Sélection

- Quels sont les clients dont l'adresse est *Paris*

```
SELECT *  
FROM Client  
WHERE adresse = 'Paris';
```

- Quels sont les produits dont le prix TTC est supérieur à *1000*

```
SELECT *  
FROM Produit  
WHERE prix_HT + prix_HT * 0.195 > 1000;
```

# Opérations possibles (mysql)

---

- Booléennes  
and, or, xor, =, !=, <, >, <=, >=
- Arithmétiques  
+, -, \*, /  
+, - *opérateurs unaires*
- Fonctions numériques  
abs, log, cos, sin, mod, power ...
- Arithmétiques sur date  
+, -
- Fonctions sur chaînes  
length, concat, ...

Un grand nombre



## Précédence des opérateurs

- :=
- ||, OR, XOR
- &&, AND
- BETWEEN, CASE, WHEN, THEN, ELSE
- =, <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
- |
- &
- <<, >>
- -, +
- \*, /, DIV, %, MOD
- ^
- - (unary minus), ~ (unary bit inversion)
- !, NOT
- BINARY, COLLATE

+ faible

```
SELECT 1+2*3;
```

7

+ forte

## Utilisation des opérateurs

---

- **SELECT** ABS (-32);      32      *Bien sur on peut utiliser des attributs*
- **SELECT** FLOOR (1.23);      1
- **SELECT** MOD (234, 10);      4      *Peut se trouver dans Where*
- **SELECT** 253 % 7;      1      **WHERE** 3 / 5 < 1;
- **SELECT** ROUND (1.298, 1);      1,3
- **SELECT** ROUND (1.298, 0);      1
- **SELECT** SIGN (234) SIGN (-32) SIGN (0);      1 / -1 / 0
- **SELECT** 3 / 5;      0,60

## Utilisation des opérateurs (Chaînes)

---

- **SELECT** **CONCAT** ('My', 'S', 'QL');    'MySQL'
- **SELECT** **CHAR\_LENGTH** ('MySQL');    5
- **SELECT** **LOCATE** ('bar', 'foobarbar');    4
- **SELECT** **LOCATE** ('bar', 'foobarbar', 5);    7
- **SELECT** **INSERT** ('Quadratic', 3, 4, 'What');    'QuWhattic'
- **SELECT** **LOWER** ('MySQL');    'mysql'
- **SELECT** **SUBSTRING** ('Quadratically',5,6);    'ratica'
- **SELECT** 'David!' **LIKE** 'David\_';    1
- **SELECT** 'David!' **LIKE** '%D%v%';    1
- **SELECT** **STRCMP** (S1, S2);    -1,0,1

<p style="text-align: center;">_ et %</p> <p>0 (false), 1 (true)</p>
--

## Combinaison Sélection + projection

---

**SELECT** liste d'attributs ou \*

*Attributs recherchés / calculés*

**FROM** liste des relations

**WHERE** condition ;

*Si plusieurs: il faut une jointure*

*Test sur chaque tuple: résultat vrai ou faux*

*Utilise des fonctions, opérateurs...*

*Autres clauses possibles*

## Requêtes simples (1)

---

- *Y a-t-il des produits dont le nom est « XBOX »*

```
SELECT *  
FROM Produit  
WHERE nom = 'XBOX' ;
```

- *Quels sont les ventes réalisés il y a plus de 30 jours?*

```
SELECT *  
FROM Vente  
WHERE CURRENT_DATE () > 30 + date ;
```

- *Quels sont les ventes faites après le 1er janvier 2007?*

```
SELECT *  
FROM Vente  
WHERE date > DATE ('2007-01-01') ;
```

## Requêtes simples (2)

---

- *Quels sont les ventes dont le montant HT est entre 1000 et 3000 euros et dont le client n'est pas le numéro 101?*

**SELECT \***

**FROM** Vente

**WHERE** (prix\_ht between 1000 and 3000) and  
(numero != 101);

- *Quels sont les clients dont le nom est soit Prosper, soit Durand, soit Anthonin ?*

**SELECT \***

**FROM** Client

**WHERE** nom in ('Prosper', 'Durand', 'Anthonin');

## Requêtes simples (3)

---

- *Quels sont les clients dont le nom commence par 'P'*

**SELECT \***

**FROM** *Employe*

**WHERE** *nom* **LIKE** 'P%';

- *Quels sont les clients dont le nom commence par 'P' et a un 'S' comme 4<sup>ème</sup> lettre*

**SELECT \***

**FROM** *Client*

**WHERE** *nom* **LIKE** 'P\_\_S%';

## Requêtes et valeurs nulles

---

- Quels sont les ventes dont la date de réalisation est inconnue?

**SELECT** \*

**FROM** Vente

**WHERE** date is null ;

Ou : IS NOT NULL

### Attention :

**WHERE** date = NULL



Whatever comparé avec NULL  
→ Ni vrai ni faux

**COUNT, MIN, SUM**



Ignore les valeurs NULL  
→ sauf COUNT (\*)

Toute opération appliquée à NULL donne pour résultat NULL

## Les autres clauses (tri)

---

```
SELECT attribut1 ..  
FROM  
WHERE expression  
ORDER BY attribut1 [ASC] [DESC]...
```

Affichage!

- Donner le numero, le prix HT et la marque des produits selon l'ordre décroissant des marques et l'ordre croissant des prix HT

```
SELECT marque, prix_ht, numero  
FROM Produit  
ORDER BY marque DESC, prix_ht ASC;
```

Mais aussi:

```
ORDER BY 1 DESC, 2 ASC;
```

Ordre d'affichage!

## Requêtes multi- tables (Opérateur *Jointure*)

---

### Deux points de vue:

- exécuter des **boucles imbriquées** (une table par boucle)  
appliquer la clause WHERE dans les boucles
- calculer le **produit cartésien** (une nouvelle table)  
Appliquer la clause WHERE sur chaque ligne

*Donner le nom d'un produit et le montant de la vente*

```
SELECT Produit.nom, Vente.prix_HT  
FROM Produit , Vente  
WHERE Vente.reference_produit = Produit.numero ;  
Critères de jointure
```

***Il faut joindre les tables (jointure)***

## Définition de la jointure

---

```
SELECT Client.nom, Vente.prix_ht  
FROM Client, Vente ;
```

→ tous les tuples (*nom, prix\_ht*)

→ nom est un nom de client et prix\_ht est un prix de vente

**Pas de critère de jointure → Produit cartésien**

**Intérêt?**



**Mots-clés pour exprimer le critère de sélection**

## Jointure (compléments)

---

- Donner la marque des produits dont le prix HT est supérieur à celui d'une BMW

```
SELECT Produit.marque
FROM Produit Produit_reference Produit
WHERE Produit_reference.marque = 'BMW' AND
       Produit.prix_HT > Produit_reference.prix_HT ;
```

Renommage

Ou de même prix HT:

```
SELECT Produit.marque
FROM Produit Produit_reference, Produit
WHERE Produit_reference.marque = 'BMW' AND
       Produit.prix_ht = Produit_reference.prix_ht AND
       Produit.marque != Produit_reference.marque ;
```

## Quelques jointures (compléments)

---

### Que deviennent les tuples non sélectionnés de R1 ou R2?

- dans la jointure (interne **INNER JOIN**) les tuples qui ne peuvent pas être joints sont éliminés du résultat
- dans la jointure externe (**OUTER JOIN**) les tuples qui ne peuvent pas être joints sont conservés dans le résultat

**Pour les tuples de la relation de gauche (R1) et / ou de droite (R2)**

Défaut

R1 **FULL OUTER JOIN** R2 : Remplit R1.\* et R2.\*

R1 **LEFT OUTER JOIN** R2 : Remplit R1.\*

R1 **RIGHT OUTER JOIN** R2 : Remplit R2.\*

avec NULL si nécessaire.

## Exercice (1)

---

### Relations :

- Journal (code-j, titre, prix, type, périodicité)
- Dépôt (no-dépôt, nom-dépôt, adresse)
- Livraison (no-dépôt, code-j, date-liv, quantité-livrée)

### Requêtes : donner...

- le prix des journaux livrés le 15/01/07 ?
- tous le nom des hebdomadaires reçus par le dépôt de Paris..
- les titre des journaux livrés à Nice.
- le nom des dépôts qui reçoivent des hebdomadaires dont la quantité livrée excède 100.

## Exercice (2)

---

```
SELECT *  
FROM LIVRAISON RIGHT JOIN (DEPOT, JOURNAL)  
ON (LIVRAISON.no-depot = DEPOT.no-depot AND  
LIVRAISON.code-j = JOURNAL.code-j)
```

*Toutes les lignes de DEPOT et JOURNAL seront présentes  
Avec éventuellement rien pour la partie livraison*

```
SELECT *  
FROM LIVRAISON, DEPOT, JOURNAL  
WHERE LIVRAISON.no-depot = DEPOT.no-depot AND  
LIVRAISON.code-j = JOURNAL.code-j)
```

*Seulement les lignes de DEPOT et JOURNAL qui  
correspondent à une livraison*

## Opérateur *Union*

---

*Afficher la liste des numéros d'employé des responsables de département et des directeurs*

*Département*

numéro-département	...	responsable
--------------------	-----	-------------

*Employé*

numéro-employé	...	fonction
----------------	-----	----------

```
SELECT responsable  
FROM Département  
UNION  
SELECT numéro-employé  
FROM Employé  
WHERE fonction = 'Directeur';
```

## Opérateurs *Intersection, Différence*

---

*Afficher les numéros d'employé des responsables de département qui sont aussi des directeurs*



```
SELECT responsable  
FROM Département  
INTERSECT  
SELECT numéro-employé  
FROM Employé  
WHERE fonction = 'Directeur';
```

```
SELECT responsable  
FROM Département  
EXCEPT  
SELECT numéro-employé  
FROM Employé  
WHERE fonction = 'Directeur';
```



*Afficher les numéros d'employé des responsables de département  
Sauf ceux qui sont aussi des directeurs*

## Fonctions d'agrégat

- SUM (*nom d'attribut*)
- COUNT(\*)
- COUNT(DISTINCT *nom d'attribut*)
- MAX (*nom d'attribut*)
- MIN (*nom d'attribut*)
- AVG (*nom d'attribut*)
- AVG (DISTINCT *nom d'attribut*)



Dans les clauses :

- SELECT
- HAVING TO

## Clause Group by

---

**SELECT** *attributs recherchés*  
**FROM** *liste des relations*  
**WHERE** *condition*  
**GROUP BY** *attributs de regroupement*  
**[HAVING** *condition sur le groupe* **];**

**SELECT** COUNT (\*)  
**FROM** *Produit*  
**GROUP BY** *marque*

**SELECT** AVG (*prix\_ht*), ~~*nom*~~  
**FROM** *Produit*  
**GROUP BY** *marque*

**SELECT** COUNT (\*)  
**FROM** *Produit*  
**WHERE** *marque* <> 'BMW'  
**GROUP BY** *marque*  
**HAVING** AVG (*prix\_ht*) > 10.5

## Utilisation des fonctions statistiques

---

- Donner la moyenne des prix HT et les prix min. et max.

```
SELECT AVG (prix_ht), MIN (prix_ht), MAX (prix_ht)
FROM Produit
```

- Même chose mais par marque

```
SELECT AVG (prix_ht), MIN (prix_ht), MAX (prix_ht)
FROM Produit
GROUP BY marque
```

- Même chose mais par marque si la moyenne > 10,5

```
SELECT AVG (prix_ht), MIN (prix_ht), MAX (prix_ht)
FROM Produit
GROUP BY marque HAVING AVG (prix_ht) > 10.5
```

# Requêtes imbriquées (1)

---

**SELECT** *liste d'attributs recherchés*  
**FROM** *liste\_des\_relations*  
**WHERE** *bloc SFW dans la condition*

Intérêt: indiquer qu'un attribut doit prendre ses valeurs dans une liste de valeurs définies par un autre bloc *SFW*

- Itérations imbriquées
- autre façon de faire certaines formes de jointure
- Sous-requêtes indépendantes ou pas

## Requêtes imbriquées (2)

---

Principaux connecteurs: = != > < IN EXISTS ANY ALL

**EXISTS**  $R$  : retourne TRUE si  $R$  n'est pas vide, FALSE sinon

$t$  **IN**  $R$  : retourne TRUE si  $t$  appartient à  $R$ , FALSE sinon

valeur comp **ANY**  $R$  : retourne TRUE si la comparaison avec **au moins un** des tuples de  $R$  renvoie TRUE

valeur comp **ALL**  $R$  : retourne TRUE si la comparaison avec **tous** les tuples de  $R$  renvoie TRUE

## Sous-requêtes indépendantes

---

Quelles sont les références produit dont le prix HT est supérieur au prix HT moyen des produits ?

**SELECT** référence

**FROM** *Produit*

**WHERE** *Produit.prix\_ht* > (**SELECT** AVG(*P.prix\_ht*)  
**FROM** *Produit P*);



Le bloc SFW imbriqué peut être évalué séparément du bloc principal

# Sous-requêtes indépendantes (1)

---

Quels sont les marques de produits vendus le 1/1/2007 ?

```
SELECT marque
FROM Produit, Vente
WHERE Vente.ref-produit = Produit.reférence
      AND date = 1/1/2007;
```

ou

```
SELECT Produit.marque           Jointure ou requête imbriquée
FROM Produit
WHERE Produit.reférence IN
      (SELECT Vente.ref-produit
       FROM Vente
       WHERE date = 1/1/2007);
```

## Sous-requêtes indépendantes (2)

---

*Quelles sont les marques de produits qui n'ont pas le prix HT le plus élevé ?*

```
SELECT marque  
FROM Produit < SELECT max (P.prix_ht) ...  
WHERE Produit.prix-ht < ANY  
      (SELECT P.prix-ht  
       FROM Produit P;)
```

*Quelles sont les marques de produits qui ont le prix HT le plus élevé ?*

```
SELECT marque > SELECT max (P.prix_ht) ...  
FROM Produit  
WHERE Produit.prix_ht >= ALL  
      (SELECT P.prix_ht FROM Produit P;)
```

## Sous-requêtes corrélées (1)

---

Quels produits dont le prix HT est supérieur au prix HT moyen des produits de la même marque ?

```
SELECT Produit.référence  
FROM Produit  
WHERE Produit.prix-ht > (SELECT AVG(P.prix-ht)  
                                FROM Produit P  
                                WHERE Produit.marque = P.marque);)
```

Le bloc *SFW* principal et le bloc *SFW* imbriqué doivent être évalués simultanément

## Sous-requêtes corrélées (2)

---

*Quels sont les marques dont aucun produit n'a été vendu le 1/01/2007 ?*

```
SELECT Produit.marque  
FROM Produit  
WHERE NOT EXISTS  
  (SELECT Vente.ref-produit  
   FROM Vente, Produit P  
   WHERE Vente.ref-produit = P.référence and  
         P.marque = Produit.marque and  
         date = 1/01/2007;)
```

# Exercices

---

Donner le nom des employés qui ont le même salaire que Dupont

*Employé*

numéro-employé	nom	...	salaire
----------------	-----	-----	---------

```
SELECT Y.nom  
FROM Employé X , Employé Y  
WHERE  
    X.nom='Dupont'  
    and Y.salaire = X.salaire  
    and X.nom != Y.nom;
```

*Remplacer la jointure par une requête imbriquée*

# Manipulation des structures de données

---

- Schéma logique / Base de données
  - CREATE/DROP SCHEMA/DATABASE ...
- Schéma de **tables** (*relations*) et de leur contenu
  - CREATE/ALTER/DROP TABLE ...
- Définition des **contraintes** qui assurent des contrôles sur l'intégrité des données
- Mais encore :
  - les index
  - les utilisateurs et les privilèges

# Création/Suppression d'un schéma

---

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name  
DROP DATABASE [IF EXISTS] db_name
```

- Dans la norme SQL: création de schéma
- MySQL: *schema* = *database*
- Il faut avoir les droits de le faire (administrateur?)
- Création d'un compte s'il n'existe pas
- Association des droits

*Pas de message  
d'erreur si la BD  
existe!*

```
CREATE DATABASE lahire06-07
```

```
GRANT ALL PRIVILEGES  
ON lahire06-07.*  
TO 'lahire06-07'@localhost  
IDENTIFIED BY 'lahire06-07';
```

```
DROP DATABASE lahire06-07
```

*Attention !*

## Types de la norme ANSI

INTEGER	Entiers relatifs	4 octets
SMALLINT	Entiers relatifs	2 octets
BIGINT	Entiers relatifs	8 octets
FLOAT	Flottants	4 octets
DOUBLE	Flottants	8 octets
REAL	Flottants	4 ou 8 octets
NUMERIC(N,D)/ DECIMAL(N,D)	Décimaux à précision fixe	N octets

CHAR(M)	Chaînes de longueur fixe	M octets
VARCHAR(M)	Chaînes de longueur variable	au plus M octets
BIT VARYING	Chaînes d'octets	longueur de la chaîne
DATE	date(hour, mois, an)	4 octets
TIME	heure(h, mn, sh)	4 octets
DATETIME	date et heure	8 octets
YEAR	année	2 octets

# Création et gestion des relations

---

- Structures de données (relations, attributs, tuples)

relation	→	table
attribut	→	column
tuple	→	row

- Instructions sur les relations

<b>CREATE TABLE</b>	<i>créer une relation</i>
<b>DROP TABLE</b>	<i>supprimer une relation</i>
<b>ALTER TABLE</b>	<i>modifier la structure d'une relation</i>

# Création de table/relation (1)

---

```
CREATE TABLE Produit (  
référence CHAR(4),  
nom VARCHAR(20),  
marque VARCHAR(15),  
constructeur CHAR(20)  
);
```

```
CREATE TABLE Client (  
numéro INT(10),  
nom VARCHAR(20),  
adresse VARCHAR(30),  
téléphone CHAR(10)  
);
```

```
CREATE TABLE Vente (  
numéro INT(10),  
ref-produit INT(10),  
ref-client INT(10),  
date DATE  
);
```

- Compléter la description
- Voir avec les types MySQL

## Création de table (valeurs nulles, défaut)

---

- NOT NULL : l'attribut correspondant doit *toujours* avoir une valeur  
nom VARCHAR(20) NOT NULL
- DEFAULT : définir la valeur d' un attribut par défaut  
adresse VARCHAR (30) DEFAULT 'Inconnue'

```
CREATE TABLE Produit (  
référence CHAR(4) NOT NULL,  
nom VARCHAR(20) NOT NULL,  
marque VARCHAR(15),  
constructeur CHAR(20) DEFAULT 'Renault'  
);
```

# Création de table (clé primaire)

---

## Identification des tuples

**PRIMARY KEY** (*att*)

**PRIMARY KEY** (*att1, ..., attn*)

→ mono attribut ou multi attributs

→ Attributs figurant dans une clé : déclaration NOT NULL

```
CREATE TABLE Produit (  
référence CHAR(4) NOT NULL,  
  nom VARCHAR(20) NOT NULL,  
  marque VARCHAR(15),  
  constructeur CHAR(20) DEFAULT 'Renault',  
  PRIMARY KEY (référence)  
);
```

# Création de table (unicité)

---

## Unicité des valeurs dans une colonne

**UNIQUE** (nom, prenom)

→ Principalement pour les clés secondaires

```
CREATE TABLE Produit (  
référence CHAR(4) NOT NULL,  
    nom VARCHAR(20) NOT NULL,  
    marque VARCHAR(15),  
    constructeur CHAR(20),  
    PRIMARY KEY (référence),  
    UNIQUE (nom)  
);
```

NOT NULL non spécifié

UNIQUE ne s'applique pas aux valeurs nulles

# Création de table (clé étrangère)

---

**FOREIGN KEY** (att) **REFERENCES** ...

**FOREIGN KEY** (att1, ..., attn) **REFERENCES** ...

**FOREIGN KEY**(ref-produit) **REFERENCES** *Produit*

→ ref-produit référence la clé primaire de la table **Produit**

```
CREATE TABLE Vente (  
  numéro INT(10) NOT NULL,  
  ref-produit INT(10) NOT NULL,  
  ref-client INT(10) NOT NULL,  
  date DATE,  
  PRIMARY KEY (numéro),  
  FOREIGN KEY (ref-produit) REFERENCES Produit,  
  FOREIGN KEY (ref-client) REFERENCES Client  
);
```

# Gestion de l'intégrité référentielle

---

## Vérification, si insertion, suppression, mise à jour ...

*Les valeurs prises par la clé étrangère correspondent-elles à la clé?*

Action par défaut : **rejet** de l'opération

## Autres actions si celle par défaut ne convient pas

- **ON UPDATE** (*en cas de mise à jour*)
- **ON DELETE** (*en cas de suppression*)

## Objectif: faire respecter la contrainte

- **SET NULL** clé étrangère mise à NULL
- **CASCADE** application de la même opération
- **SET DEFAULT** valeur par défaut pour la clé étrangère

## Intégrité référentielle (exemple)

---

```
CREATE TABLE Vente (  
numéro INT(10) NOT NULL,  
  ref-produit INT(10) NOT NULL,  
  ref-client INT(10) NOT NULL,  
  date DATE,  
  PRIMARY KEY (numéro),  
  FOREIGN KEY (ref-produit) REFERENCES Produit  
  ON DELETE SET NULL ON UPDATE CASCADE,  
  FOREIGN KEY (ref-client) REFERENCES Client  
  ON DELETE SET NULL ON UPDATE CASCADE  
);
```

- Si suppression dans *Produit* : ref-produit = NULL
- Si mise à jour dans *Produit* : ref-produit = mise à jour

# Mise à jour d'un schéma de BD

---

- Suppression d'une relation

`DROP TABLE` *nom de relation* ;

- Modification d'une table

`ALTER TABLE` *nom de \_relation* **ACTION** *description*

ADD, MODIFY, ALTER, DROP, RENAME

*description* commande

Mais aussi :

`ALTER TABLE` Vente TYPE = *innodb*

## Modification d'une table (structure)

---

```
ALTER TABLE Produit MODIFY nom VARCHAR(25);
```

```
ALTER TABLE Produit ADD quantité-stock VARCHAR(20);
```

```
ALTER TABLE Client ALTER adresse SET DEFAULT 'NICE';
```

```
ALTER TABLE Client DROP adresse;
```

```
ALTER TABLE Client  
  ADD CONSTRAINT CT_UN UNIQUE (nom) ;
```

```
ALTER TABLE Client  
  ADD CONSTRAINT CT_PR PRIMARY KEY (numéro) ;
```

```
ALTER TABLE Vente ADD CONSTRAINT CT_ET  
  FOREIGN KEY (ref-client) REFERENCES Client ;
```

## Mises à jour des données

---

- **INSERT INTO**  
pour insérer des tuple
- **DELETE FROM**  
pour supprimer des tuples
- **UPDATE**  
pour mettre à jour des tuples

## Insertion de tuples (1)

---

### Insertion avec désignation explicite des colonnes

```
INSERT INTO Produit (nom, marque, constructeur)  
VALUES ('mercedes 300 SL', 'mercedes benz',  
         'daimler');
```

### Insertion dans l'ordre des colonnes

```
INSERT INTO Produit  
VALUES (DEFAULT, 'mercedes 300 SL', 'mercedes  
         benz', 'daimler');
```

## Insertion de tuples (2)

---

Insertion à partir d'une autre table

```
INSERT INTO Produit (nom, marque, constructeur)  
SELECT P.nom, P.marque, p.constructeur  
FROM OldProduit P  
WHERE P.marque = 'Peugeot'
```

➡ **Evolution du schéma relationnel**

# Suppression de tuples

---

Suppression de tous les tuples

**DELETE FROM** *Produit*

ou

**DELETE FROM** *Produit* **WHERE** 1 > 0

Suppression conditionnelle

**DELETE FROM** *Produit*

**WHERE** *marque* = 'peugeot'

**DELETE FROM** *VENTE* **WHERE** ref-produit **IN**

(**SELECT** *référence*

**FROM** *Produit*

**WHERE** *Produit.marque* = 'peugeot');

# Modification des tuples

---

```
UPDATE Produit  
SET nom = 'mercedes 300 SLK'  
WHERE numero =3;
```

*Modification d'un tuple*

```
UPDATE Produit  
SET prix-ht = prix-ht * 0,9  
WHERE référence NOT IN  
  (SELECT ref-produit  
   FROM VENTE  
   WHERE quantité > 10);
```

*Modification d'une  
collection de tuple*

## Compléments: Définition de vues

Une vue est une table virtuelle dérivée de tables de base :

- On stocke seulement la définition de vue
- Son contenu est généré dynamiquement

```
CREATE VIEW ProduitPhare  
AS  
SELECT *  
FROM Vente V, Produit P  
WHERE  
  V.ref-produit = P.référence  
AND V.quantité > 100 ;
```

```
CREATE VIEW VenteInfo  
  (référence, date, marque)  
AS  
SELECT P.référence, V.date,  
         P.marque  
FROM Vente V, Produit P  
WHERE P.référence = V.ref-produit;
```

**Evolution du schéma relationnel**



**Requêtes fréquentes**

*DROP VIEW VenteInfo*

## Compléments: Création de contraintes

---

**CREATE ASSERTION**

**CHECK** *pour décrire des contraintes générales*

**CREATE ASSERTION** *QuantiteVendue*

**CHECK NOT EXISTS** (

**SELECT** *V.quantité*

**FROM** *Vente V, Produit P*

**WHERE**

*V.ref-produit = P.référence AND*

*P.quantité-vendue < V.quantité);*

*On suppose que la table Produit  
contient la colonne quantité-vendue*

## Compléments: Création d'index

---

### **CREATE INDEX**

Un index offre un chemin d'accès aux lignes d'une table

```
CREATE INDEX index1 ON Client(nom,prenom);
```

un index est systématiquement défini sur la clé primaire de chaque table

pour chaque clause UNIQUE utilisée dans la création de la table, un index permet de vérifier rapidement, au moment d'une insertion, que la clé n'existe pas déjà

## Compléments: Gestion des privilèges

---

SHOW DATABASES ;

SHOW PRIVILEGES ;

GRANT *privilèges* ON ... TO ...

REVOKE *privilèges* ON ... FROM ...

ALL [PRIVILEGES]	Tous les droits sauf WITH GRANT OPTION
ALTER	Autorise l'utilisation de ALTER TABLE
CREATE	Autorise l'utilisation de CREATE TABLE
DELETE	Autorise l'utilisation de DELETE
DROP	Autorise l'utilisation de DROP TABLE
...	...