

DUT SRC – IUT de Marne-la-Vallée  
05/02/2014  
M2203 – Bases de données

## ***Cours 3***

# ***Le langage SQL***

# Sources

---

- Cours de Tony Grandame à l'IUT de Marne-la-Vallée en 2010-2011

- Cours de Mathieu Mangeot, IUT de Savoie

<http://jibiki.univ-savoie.fr/~mangeot/Cours/BasesDeDonnees.pdf>

- Cours de Fabrice Meuzeret, IUT de Troyes

<http://195.83.128.55/~fmeuzeret/vrac/>

- Livre de Laurent Audibert : *Bases de données - de la modélisation au SQL*

Version partielle sur :

<http://laurent-audibert.developpez.com/Cours-BD/html/index.php>

# Plan du cours 3 – Le langage SQL

---

- Résumé des épisodes précédents
- Introduction au langage SQL
- Langage de définition des données
- Intermède sur PHP
- Langage de manipulation des données
- SQL avancé : les jointures
- SQL avancé : les groupements
- SQL avancé : les transactions

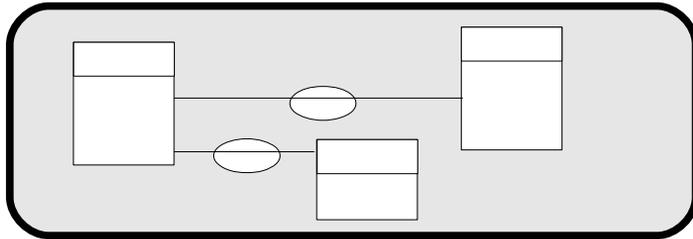
# Plan

---

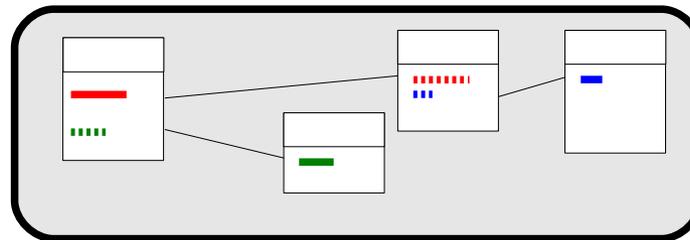
- Résumé des épisodes précédents
- Introduction au langage SQL
- Langage de définition des données
- Intermède sur PHP
- Langage de manipulation des données
- SQL avancé : les jointures
- SQL avancé : les groupements
- SQL avancé : les transactions

# Modèle physique des données

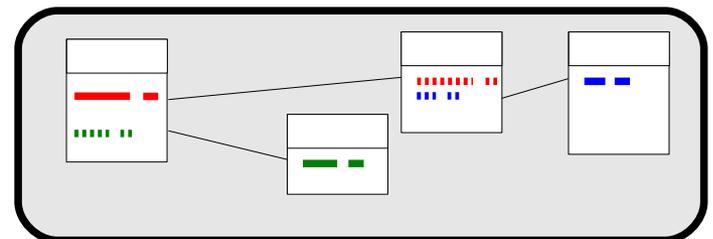
Modèle entité-association  
(modèle conceptuel des données)



Modèle logique des données



Modèle physique des données



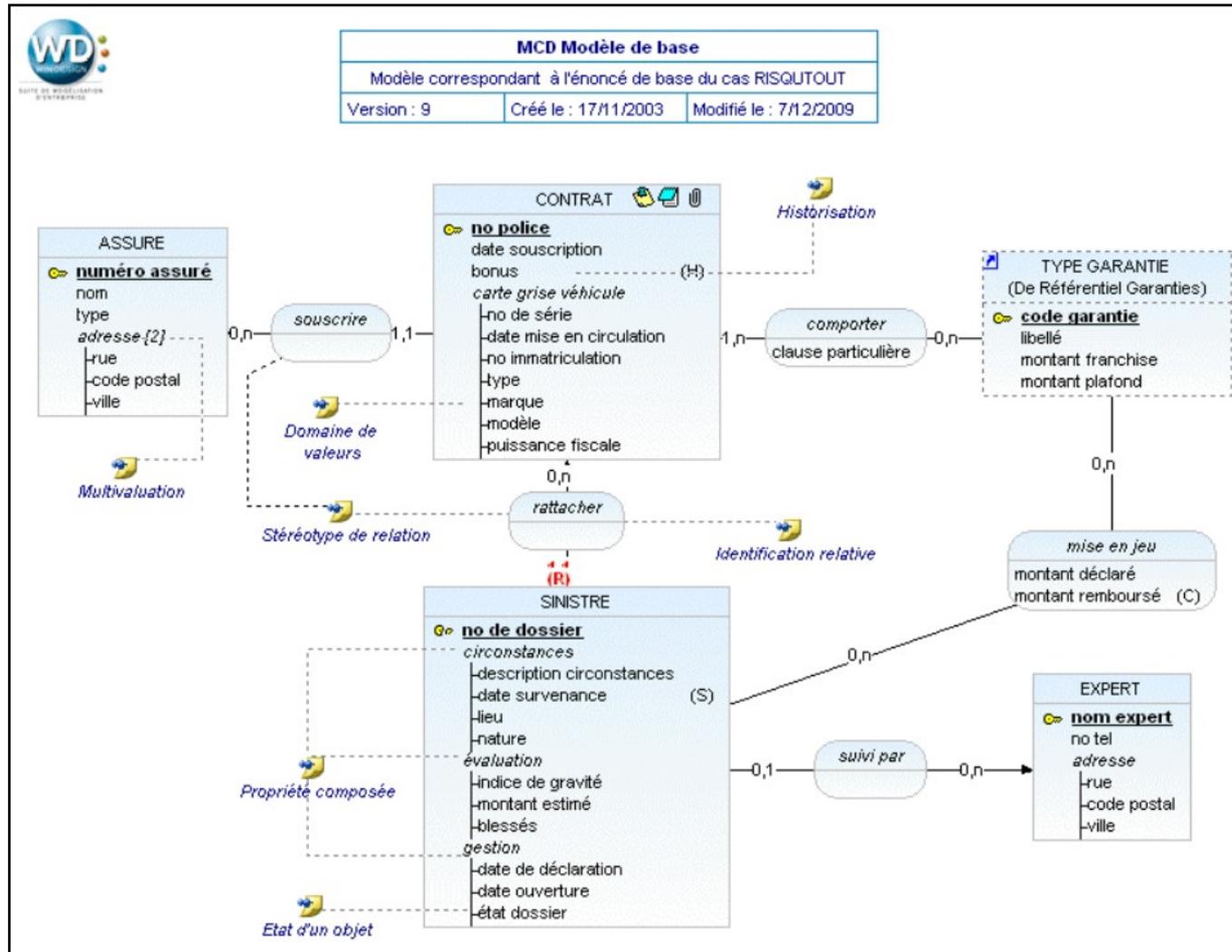
# Transformation vers le modèle logique des données

Modèle entité association



Modèle logique des données

Transformation automatique : exemple de WinDesign Database



# Transformation vers le modèle logique des données

Modèle entité association



Modèle logique des données

Transformation automatique : exemple de WinDesign Database

The screenshot shows the WinDesign Database interface. At the top, a metadata box for 'MCD Modèle de base' indicates it corresponds to the 'RISQUTOUT' case, with version 9, created on 17/11/2003, and modified on 7/12/2009. Below this, an Entity-Relationship diagram shows an 'ASSURE' entity with attributes 'numéro assuré', 'nom', 'type', and 'adresse {2}' (with sub-attributes 'rue', 'code postal', 'ville'). A 'CONTRAT' entity is also visible with a 'no police' attribute. The main window displays a table with two columns: 'Période' and 'Thème'. The 'Période' column contains 'J 2' and 'Matinée'. The 'Thème' column contains 'MODELE LOGIQUE DE DONNEES'. The table content is as follows:

Période	Thème
J 2 Matinée	<b>MODELE LOGIQUE DE DONNEES</b> <ul style="list-style-type: none"><li>▪ <b>Principes généraux de la modélisation logique des données</b></li><li>▪ <b>Concepts de base</b><ul style="list-style-type: none"><li>- table, attribut, types de données</li><li>- clé primaire, clé étrangère et contraintes référentielles, index</li></ul></li><li>▪ <b>Transformation MCD ⇒ MLD</b><ul style="list-style-type: none"><li>- préparation à la transformation,</li><li>- options de la transformation automatique, règles de nommage,</li></ul><i>Exercices d'application en continu sur la cas Risquetout</i></li><li>▪ <b>Optimisation du MLD</b><ul style="list-style-type: none"><li>- dénormalisation</li><li>- tables de valeurs codées</li><li>- index, choix d'implémentation de liens référentiels</li><li>- historique des suppressions</li><li>- mise en conformité du MCD</li></ul><i>Exercices d'application en continu sur la cas Risquetout</i></li><li>▪ <b>Concepts avancés</b><ul style="list-style-type: none"><li>- vue SQL</li><li>- règle (implémentation et codage),</li><li>- trigger (référentiel et utilisateur)</li><li>- implantation physique (storage et tablespace)</li></ul><i>Exercices d'application en continu sur la cas Risquetout</i></li></ul>

# Plan

---

- Résumé des épisodes précédents
- **Introduction au langage SQL**
- Langage de définition des données
- Intermède sur PHP
- Langage de manipulation des données
- SQL avancé : les jointures
- SQL avancé : les groupements
- SQL avancé : les transactions

# Introduction au langage SQL

## SQL

- Structured Query Language
- Langage standardisé pour effectuer des opérations sur des bases de données.
- **LDD : langage de définition de données**, pour gérer les structures de la base
- **LMD : langage de manipulation de données**, pour interagir avec les données.

Attention, certaines syntaxes ou fonctions sont propres au système de base de données utilisé.

# Introduction au langage SQL

## SQL

- Structured Query Language (“query” = “requête”)
- Langage standardisé pour effectuer des opérations sur des bases de données.
- **LDD** : langage de définition de données, pour gérer les structures de la base
- **LMD** : langage de manipulation de données, pour interagir avec les données.

Attention, certaines syntaxes ou fonctions sont propres au système de base de données utilisé.

**requête** : instruction demandant une action sur la base de données.

*Alternative au langage SQL : clic-clic-poët-poët avec PhpMyAdmin*

# Plan

---

- Résumé des épisodes précédents
- Introduction au langage SQL
- Langage de définition des données
- Intermède sur PHP
- Langage de manipulation des données
- SQL avancé : les jointures
- SQL avancé : les groupements
- SQL avancé : les transactions

# Langage de définition des données

## Bases

Une base regroupe toutes les données nécessaires pour un besoin fonctionnel précis : **une application ↔ une base de données.**

Possible de créer autant de bases de données que nécessaires, interaction entre les bases de données possible, mais alourdit la syntaxe SQL.

## Création d'une base de données

```
CREATE DATABASE [IF NOT EXISTS] db_name  
[create_specification]
```

Les spécifications permettent notamment de définir l'encodage de caractères de la base :

```
CREATE DATABASE db_name DEFAULT CHARACTER SET latin1  
COLLATE latin1_swedish_ci;
```

# Langage de définition des données

## Bases

Une base regroupe toutes les données nécessaires pour un besoin fonctionnel précis : **une application ↔ une base de données.**

Possible de créer autant de bases de données que nécessaires, interaction entre les bases de données possible, mais alourdit la syntaxe SQL.

## Suppression d'une base de données

```
DROP DATABASE [IF EXISTS] db_name
```

# Langage de définition des données

## Bases

Une base regroupe toutes les données nécessaires pour un besoin fonctionnel précis : **une application ↔ une base de données.**

Possible de créer autant de bases de données que nécessaires, interaction entre les bases de données possible, mais alourdit la syntaxe SQL.

## Modification d'une base de données

```
ALTER DATABASE db_name alter_specification [,  
alter_specification] ...
```

# Langage de définition des données

## Tables

Rappel : Une table correspond à une entité.

Une base de données contient une ou plusieurs tables.

### Création d'une table :

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name  
[(create_definition,...)] [table_options]
```

- 1. `create_definition` représente la liste des champs avec leur type et leurs éventuelles options.
- 2. `table_option` permet de préciser notamment le système d'encodage des caractères, et le moteur de la table (`ENGINE`).

# Introduction au langage SQL

## SQL

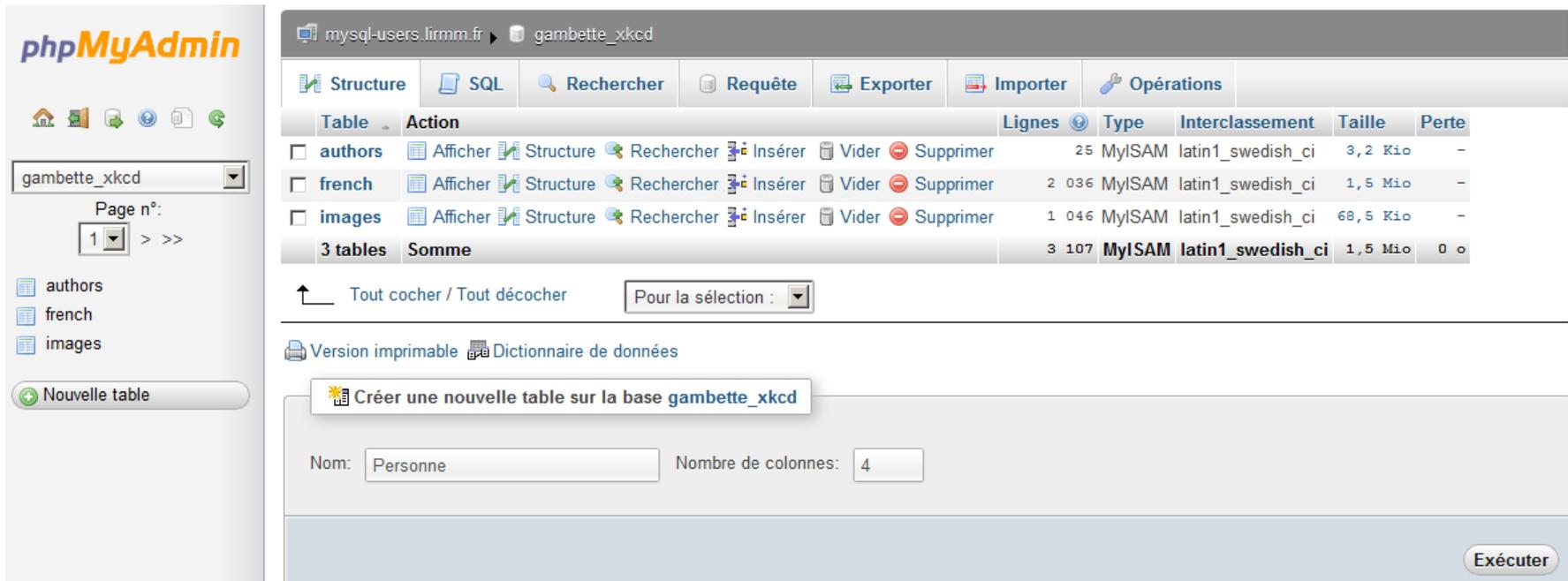
- Structured Query Language (“query” = “requête”)
- Langage standardisé pour effectuer des opérations sur des bases de données.
- **LDD : langage de définition de données**, pour gérer les structures de la base
- **LMD : langage de manipulation de données**, pour interagir avec les données.

Attention, certaines syntaxes ou fonctions sont propres au système de base de données utilisé.

**requête** : instruction demandant une action sur la base de données.

*Alternative au langage SQL : clic-clic-poët-poët avec PhpMyAdmin*

# Création d'une table avec PhpMyAdmin



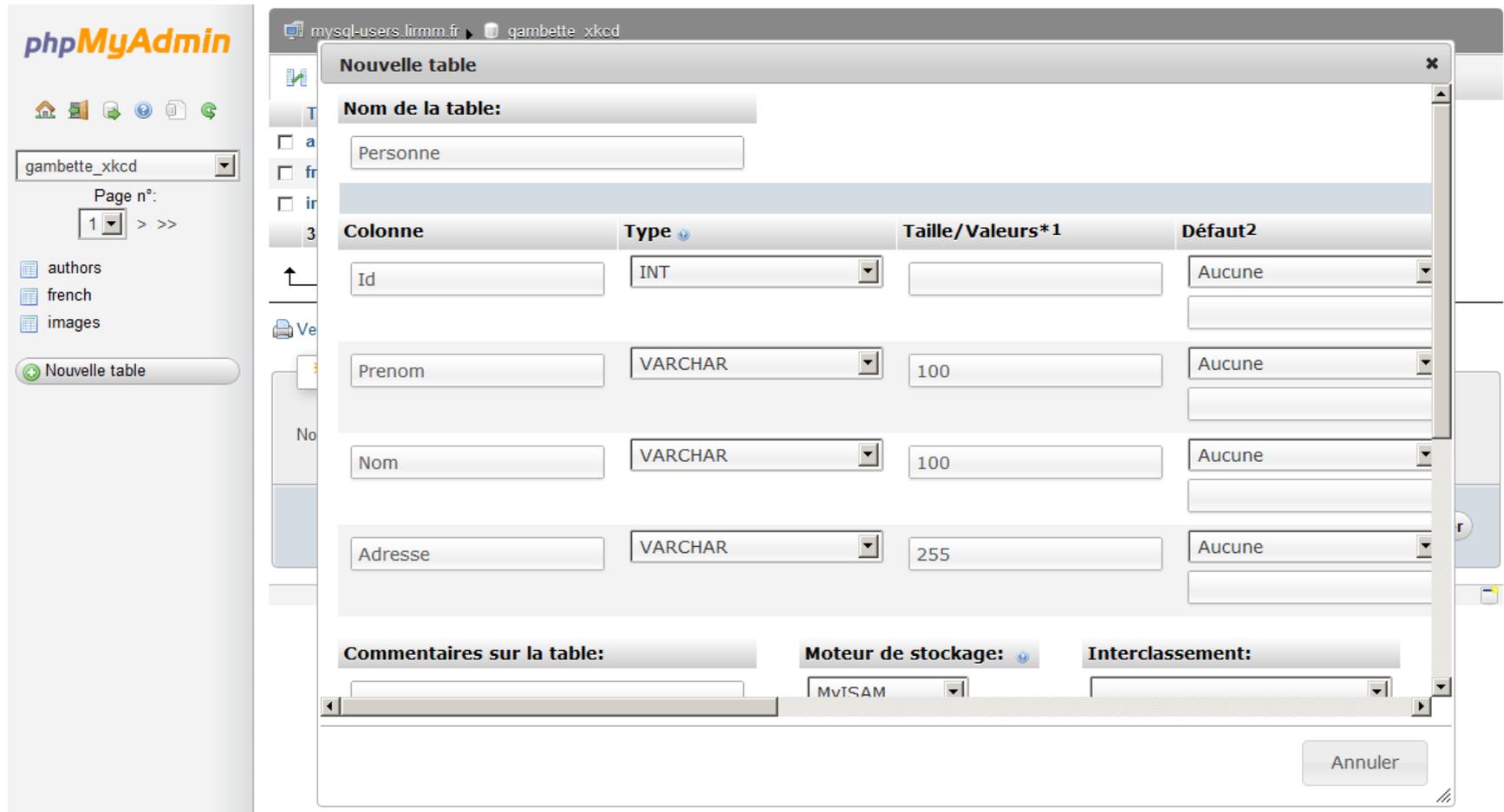
The screenshot shows the phpMyAdmin interface for the 'gambette\_xkcd' database. The left sidebar contains navigation icons and a list of tables: 'authors', 'french', and 'images'. A 'Nouvelle table' button is visible at the bottom of the sidebar. The main area displays a table list with columns: Table, Action, Lignes, Type, Interclassement, Taille, and Perte. The tables listed are 'authors' (25 lines), 'french' (2 036 lines), and 'images' (1 046 lines). Below the table list, there are options to 'Tout cocher / Tout décocher' and a dropdown for 'Pour la sélection'. A 'Version imprimable' and 'Dictionnaire de données' link are also present. The 'Créer une nouvelle table sur la base gambette\_xkcd' dialog is open, showing a form with 'Nom: Personne' and 'Nombre de colonnes: 4'. An 'Exécuter' button is located at the bottom right of the dialog.

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> authors	Afficher Structure Rechercher Insérer Vider Supprimer	25	MyISAM	latin1_swedish_ci	3,2 Kio	-
<input type="checkbox"/> french	Afficher Structure Rechercher Insérer Vider Supprimer	2 036	MyISAM	latin1_swedish_ci	1,5 Mio	-
<input type="checkbox"/> images	Afficher Structure Rechercher Insérer Vider Supprimer	1 046	MyISAM	latin1_swedish_ci	68,5 Kio	-
<b>3 tables</b>	<b>Somme</b>	<b>3 107</b>	<b>MyISAM</b>	<b>latin1_swedish_ci</b>	<b>1,5 Mio</b>	<b>0</b>

Nom:  Nombre de colonnes:

Exécuter

# Création d'une table avec PhpMyAdmin



The screenshot shows the 'Nouvelle table' (New Table) dialog in phpMyAdmin. The table name is 'Personne'. The columns are defined as follows:

Colonne	Type	Taille/Valeurs*1	Défaut2
Id	INT		Aucune
Prenom	VARCHAR	100	Aucune
Nom	VARCHAR	100	Aucune
Adresse	VARCHAR	255	Aucune

At the bottom, the storage engine is set to 'MyISAM' and the sorting is set to 'Interclassement'. An 'Annuler' (Cancel) button is visible in the bottom right corner.

# Création d'une table avec PhpMyAdmin

The screenshot displays the 'Nouvelle table' (New Table) window in phpMyAdmin. The window title is 'mysql-users.lirmm.fr - gambette\_xkcd'. The main area is titled 'Nouvelle table' and contains the following fields:

- Prenom:** VARCHAR, 100, Aucune
- Nom:** VARCHAR, 100, Aucune
- Adresse:** VARCHAR, 255, Aucune

Below the column definitions, there are three sections:

- Commentaires sur la table:** A text input field.
- Moteur de stockage:** MyISAM (selected)
- Interclassement:** A dropdown menu.

At the bottom, there is a section for 'Définition de PARTITION:' with a text input field. Below this, a summary bar shows 'Sauvegarder' (Save) or 'Ajouter' (Add) 1 colonne(s) (column(s)). The 'Exécuter' (Execute) button is circled in red. An 'Annuler' (Cancel) button is located at the bottom right.

# Langage de définition des données

## Tables

Rappel : Une table correspond à une entité.

Une base de données contient une ou plusieurs tables.

### Création d'une table :

- 1. La liste des champs doit être précisée :

```
col_name type [NOT NULL | NULL] [DEFAULT  
default_value] [AUTO_INCREMENT] [[PRIMARY] KEY]  
[reference_definition]
```

Seuls le nom et le type sont obligatoires.

Par défaut un champ est défini en NULL.

Les champs sont séparés par des virgules.

# Langage de définition des données

## Tables

Rappel : Une table correspond à une entité.

Une base de données contient une ou plusieurs tables.

### Création d'une table :

- 1. L'option `AUTO_INCREMENT` permet de confier la gestion du champ par le moteur de base de données.

A chaque insertion dans la table, la valeur du champ sera automatiquement incrémentée.

Cette option n'est possible que sur des champs de type entier.

Le type `SERIAL` est un raccourci pour définir un champ `UNSIGNED BIGINT AUTO_INCREMENT UNIQUE`.

# Langage de définition des données

## Tables

Rappel : Une table correspond à une entité.

Une base de données contient une ou plusieurs tables.

## Création d'une table :

- 2. Les options facultatives de la tables permettent de préciser (en outre) :
  - le moteur de la table :
    - MyIsam (par défaut)
    - InnoDB (gère les transactions)
    - Memory (chargée en mémoire)
  - Le système d'encodage de caractères, par défaut `latin1_swedish_ci` correspondant à ISO-8859.

# Langage de définition des données

## Tables

Rappel : Une table correspond à une entité.

Une base de données contient une ou plusieurs tables.

## Exemples de création d'une table

```
CREATE TABLE IF NOT EXISTS Coord (Id int(11) NOT NULL
auto_increment, Name varchar(255) collate
latin1_general_ci NOT NULL, Type varchar(255) collate
latin1_general_ci NOT NULL, Coord varchar(255)
collate latin1_general_ci NOT NULL, Url varchar(255)
collate latin1_general_ci NOT NULL, PRIMARY KEY
(Id)) ENGINE=MyISAM DEFAULT CHARSET=latin1
COLLATE=latin1_general_ci AUTO_INCREMENT=201 ;
```

# Langage de définition des données

## Index

Un index permet au moteur d'**accéder rapidement à la donnée recherchée**.

Si vous recherchez un champ ayant une valeur donnée et qu'il n'y a pas d'index sur ce champ, le moteur devra parcourir toute la table.

Index à utiliser avec parcimonie : pénalisent les temps d'insertion et de suppression des données dans la table.

Une clé primaire est par définition un index unique sur un champ non nul.  
Un index peut être nul.

```
CREATE TABLE IF NOT EXISTS Personne(Id int NOT NULL  
primary key auto_increment, Nom varchar(100) not  
null, Prenom varchar(100), Annee_naiss year default  
"1950") ENGINE=InnoDB
```

# Langage de définition des données

## Index

Un index permet au moteur d'**accéder rapidement à la donnée recherchée**.

Si vous recherchez un champ ayant une valeur donnée et qu'il n'y a pas d'index sur ce champ, le moteur devra parcourir toute la table.

Index à utiliser avec parcimonie : pénalisent les temps d'insertion et de suppression des données dans la table.

Une clé primaire est par définition un index unique sur un champ non nul.  
Un index peut être nul.

```
CREATE TABLE IF NOT EXISTS Personne(Id int NOT NULL  
primary key auto_increment, Nom varchar(100) not  
null, Prenom varchar(100), Annee_naiss year default  
"1950") ENGINE=InnoDB
```

```
CREATE TABLE IF NOT EXISTS Personne(Nom varchar(100)  
not null, Prenom varchar(100), Annee_naiss year  
default "1950", primary key (Nom, Prenom), index  
personne_anne (Annee_naiss)) ENGINE=InnoDB
```

# Langage de définition des données

## Modification d'une table

```
CREATE TABLE tbl_name
ADD [COLUMN] column_definition [FIRST | AFTER
col_name ]
| ADD INDEX [index_name] [index_type]
(index_col_name,...)
| ADD PRIMARY KEY [index_type] (index_col_name,...)
| ALTER [COLUMN] col_name {SET DEFAULT literal | DROP
DEFAULT}
| ALTER TABLE tbl_name
| ADD FOREIGN KEY [index_name] (index_col_name,...)
| CHANGE [COLUMN] old_col_name column_definition
| DROP [COLUMN] col_name
| DROP PRIMARY KEY
| DROP INDEX index_name
| DROP FOREIGN KEY fk_symbol
```

# Langage de définition des données

## Modification d'une table

Renommage d'une table :

```
RENAME TABLE nom_de_table TO nouveau_nom_de_table
```

Suppression d'une table :

```
DROP TABLE tbl_name
```

Attention, cette action est irréversible, toutes les données contenues dans la table sont évidemment supprimées.

# Plan

---

- Résumé des épisodes précédents
- Introduction au langage SQL
- Langage de définition des données
- **Intermède sur PHP**
- Langage de manipulation des données
- SQL avancé : les jointures
- SQL avancé : les groupements
- SQL avancé : les transactions

# Plan

---

- Résumé des épisodes précédents
- Introduction au langage SQL
- Langage de définition des données
- Intermède sur PHP
- **Langage de manipulation des données**
- SQL avancé : les jointures
- SQL avancé : les groupements
- SQL avancé : les transactions

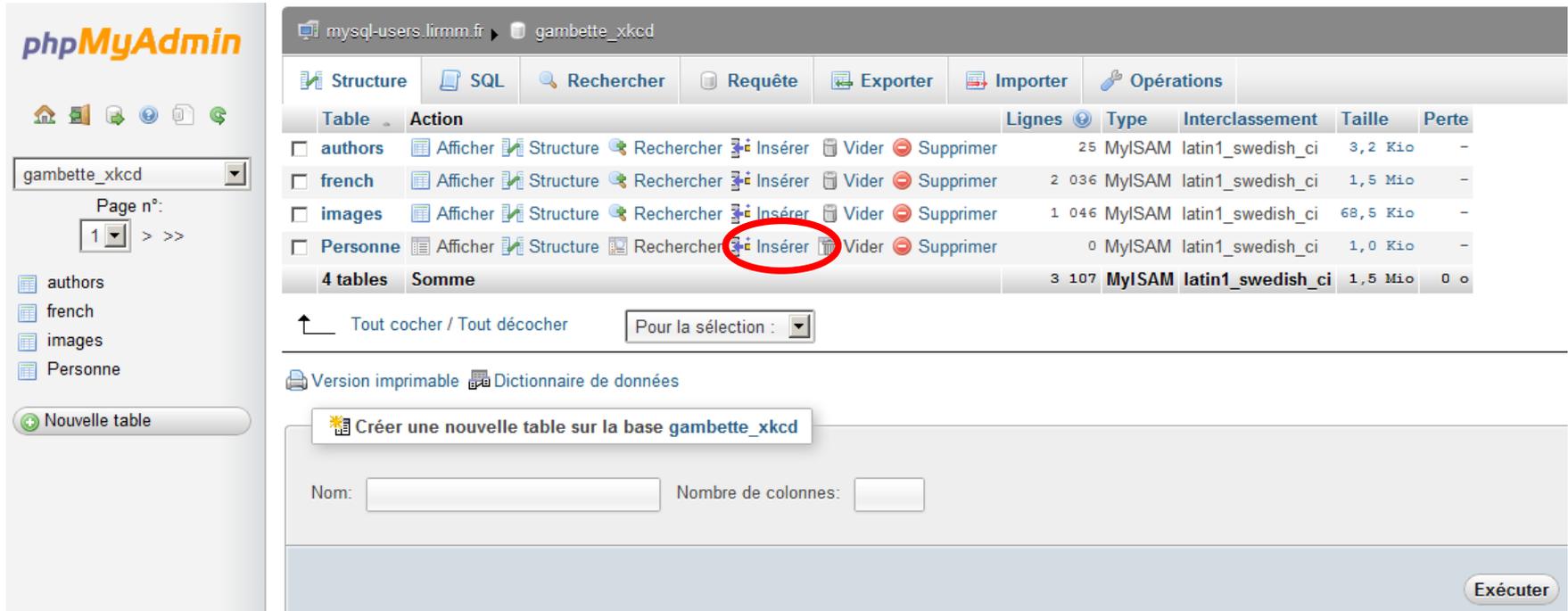
# Langage de manipulation des données

---

Les commandes principales sont :

- INSERT pour **ajouter** les données
- UPDATE pour **modifier** les données
- DELETE pour **supprimer** les données
- SELECT pour **consulter** les données

# Insertion d'occurrences dans une table avec PhpMyAdmin



The screenshot shows the phpMyAdmin interface for a MySQL database named 'gambette\_xkcd'. The left sidebar contains a list of tables: 'authors', 'french', 'images', and 'Personne'. The main area displays a table overview for 'Personne', with the 'Insérer' button circled in red. Below the table overview, there is a form to create a new table on the 'gambette\_xkcd' database, with fields for 'Nom' and 'Nombre de colonnes', and an 'Exécuter' button.

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> authors	Afficher Structure Rechercher Insérer Vider Supprimer	25	MyISAM	latin1_swedish_ci	3,2 Kio	-
<input type="checkbox"/> french	Afficher Structure Rechercher Insérer Vider Supprimer	2 036	MyISAM	latin1_swedish_ci	1,5 Mio	-
<input type="checkbox"/> images	Afficher Structure Rechercher Insérer Vider Supprimer	1 046	MyISAM	latin1_swedish_ci	68,5 Kio	-
<input type="checkbox"/> <b>Personne</b>	Afficher Structure Rechercher <b>Insérer</b> Vider Supprimer	0	MyISAM	latin1_swedish_ci	1,0 Kio	-
<b>4 tables</b>	<b>Somme</b>	<b>3 107</b>	<b>MyISAM</b>	<b>latin1_swedish_ci</b>	<b>1,5 Mio</b>	<b>0 0</b>

Version imprimable Dictionnaire de données

Créer une nouvelle table sur la base gambette\_xkcd

Nom:  Nombre de colonnes:

Exécuter



# Insertion d'occurrences dans une table avec PhpMyAdmin

The screenshot shows the phpMyAdmin interface for a MySQL database. The breadcrumb path is 'mysql-users.lirmm.fr > gambette\_xkcd > Personne'. The top navigation bar includes 'Structure', 'SQL', 'Rechercher', 'Requête', 'Exporter', 'Importer', and 'Opérations'. A green message box indicates '1 ligne insérée. Identifiant de la ligne insérée : 1'. The SQL editor contains the following query:

```
INSERT INTO `gambette_xkcd`.`Personne` (  
  `id` ,  
  `Prenom` ,  
  `Nom` ,  
  `Adresse`  
)  
VALUES (  
  NULL , 'Philippe' , 'Gambette' , '49T rue Haquenot'
```

At the bottom right of the SQL editor, there are links: [En ligne] [Modifier] [Créer source PHP]. Below the SQL editor is a table listing the database tables:

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> authors	<a href="#">Afficher</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	25	MyISAM	latin1_swedish_ci	3,2 Kio	-
<input type="checkbox"/> french	<a href="#">Afficher</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	2 036	MyISAM	latin1_swedish_ci	1,5 Mio	-
<input type="checkbox"/> images	<a href="#">Afficher</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	1 046	MyISAM	latin1_swedish_ci	68,5 Kio	-
<input type="checkbox"/> Personne	<a href="#">Afficher</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	1	MyISAM	latin1_swedish_ci	2,0 Kio	-
<b>4 tables</b>	<b>Somme</b>	<b>3 108</b>	<b>MyISAM</b>	<b>latin1_swedish_ci</b>	<b>1,5 Mio</b>	<b>0 0</b>

Below the table, there are controls for 'Tout cocher / Tout décocher' and a dropdown menu for 'Pour la sélection :'. At the bottom, there are links for 'Version imprimable' and 'Dictionnaire de données'.

# Langage de manipulation des données - INSERT

## Insérer des données dans une table :

```
INSERT [INTO] tbl_name [(col_name, ...)]  
VALUES ({expr | DEFAULT}, ...)
```

Le nombre de `col_name` doit correspondre au nombre d'`expr`.

Le fait de préciser les champs est optionnel mais impose en cas de non indication de donner les expressions de chaque colonne dans l'ordre.

Pour les champs ayant l'option `AUTO_INCREMENT`, il est possible :

- soit de ne pas préciser le champ dans la liste,
- soit de passer la valeur `NULL`.

Le système se chargera d'attribuer automatiquement une valeur.

# Langage de manipulation des données - UPDATE

## Modifier des données dans une table :

```
UPDATE tbl_name  
SET col_name1=expr1 [,col_name2=expr2 ...]  
[WHERE where_definition] [LIMIT row_count]
```

Le `SET` permet d'attribuer une nouvelle valeur au champ.

Il est possible de mettre à jour plusieurs champs en même temps.

Le `WHERE` permet de préciser quelles données on désire mettre à jour.

Son fonctionnement sera détaillé avec la commande `SELECT`.

Sans clause `WHERE`, toutes les données de la table sont mises à jour.

La `LIMIT` permet de limiter le nombre de lignes à modifier.

# Langage de manipulation des données - DELETE

## Supprimer des données dans une table :

```
DELETE FROM table_name  
[WHERE where_definition] [LIMIT row_count]
```

Le `WHERE` permet de préciser quelles données on désire supprimer.

Sans clause `WHERE`, toutes les données de la table sont supprimées. On préfère alors utiliser la commande spéciale `TRUNCATE TABLE`.

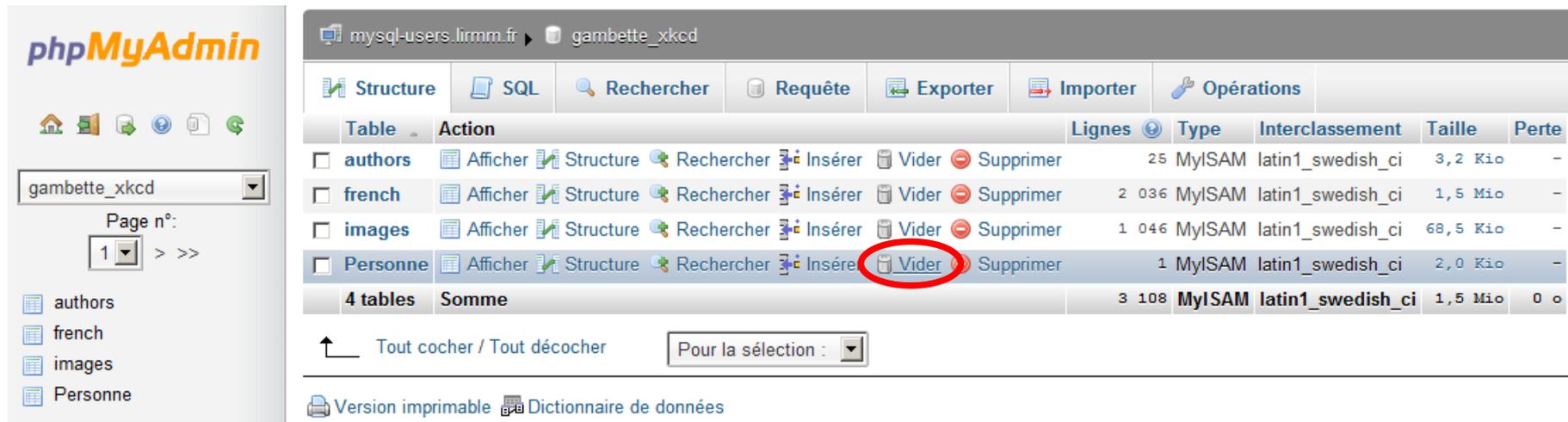
# Langage de manipulation des données - DELETE

## Supprimer des données dans une table :

```
DELETE FROM table_name  
[WHERE where_definition] [LIMIT row_count]
```

Le WHERE permet de préciser quelles données on désire supprimer.

Sans clause WHERE, toutes les données de la table sont supprimées. On préfère alors utiliser la commande spéciale TRUNCATE TABLE.



The screenshot shows the phpMyAdmin interface for a MySQL database named 'gambette\_xkcd'. The 'Personne' table is selected, and the 'Vider' (Empty) button is circled in red. The table's structure is as follows:

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> authors	Afficher Structure Rechercher Insérer Vider Supprimer	25	MyISAM	latin1_swedish_ci	3,2 Kio	-
<input type="checkbox"/> french	Afficher Structure Rechercher Insérer Vider Supprimer	2 036	MyISAM	latin1_swedish_ci	1,5 Mio	-
<input type="checkbox"/> images	Afficher Structure Rechercher Insérer Vider Supprimer	1 046	MyISAM	latin1_swedish_ci	68,5 Kio	-
<input checked="" type="checkbox"/> <b>Personne</b>	Afficher Structure Rechercher Insérer <b>Vider</b> Supprimer	1	MyISAM	latin1_swedish_ci	2,0 Kio	-
<b>4 tables</b>	<b>Somme</b>	<b>3 108</b>	<b>MyISAM</b>	<b>latin1_swedish_ci</b>	<b>1,5 Mio</b>	<b>0 o</b>

At the bottom of the interface, there are options for 'Tout cocher / Tout décocher' and 'Pour la sélection :'. The 'Version imprimable' and 'Dictionnaire de données' links are also visible.

# Langage de manipulation des données - SELECT

## Lire des données dans une ou plusieurs tables :

```
SELECT [DISTINCT] select_expression, ...
FROM table_references
    [WHERE where_definition]
    [ORDER BY {unsigned_integer | nom_de_colonne}
            [ASC | DESC] , ...]
    [LIMIT [offset,] lignes]
```

`select_expression` indique la colonne à lire, une constante, ou une valeur calculée.

Le `DISTINCT` permet de ne lire que des valeurs distinctes.

Le `FROM` permet de lister les tables à utiliser dans la recherche des données.

Le `ORDER BY` permet de trier le résultat de la requête (`ASC` : croissant, `DESC` : décroissant).

# Langage de manipulation des données - SELECT

## Exemples

On désire lire les noms rangés par ordre alphabétique de toutes les personnes qui se prénomment Lisa.

Personne	
<u>ID</u>	<u>int</u>
Nom	varchar(30)
Prenom	varchar(30)
Adress#	int

# Langage de manipulation des données - SELECT

## Exemples

On désire lire les noms rangés par ordre alphabétique de toutes les personnes qui se prénomment Lisa.

```
SELECT Nom FROM Personne  
WHERE Prenom = 'Lisa' ORDER BY 1
```

On désire lire tous les noms et prénoms associés dans un champ séparés par un espace.

Personne	
<u>ID</u>	<u>int</u>
Nom	varchar(30)
Prenom	varchar(30)
Adress#	int

# Langage de manipulation des données - SELECT

## Exemples

On désire lire les noms rangés par ordre alphabétique de toutes les personnes qui se prénomment Lisa.

```
SELECT Nom FROM Personne  
WHERE Prenom = 'Lisa' ORDER BY 1
```

On désire lire tous les noms et prénoms associés dans un champ séparés par un espace.

```
SELECT concat(Nom, ' ', Prenom) as Gens  
FROM Personne ORDER BY 1
```

On désire lire les ID de toutes les personnes ayant une adresse renseignée.

Personne	
<u>ID</u>	<u>int</u>
Nom	varchar(30)
Prenom	varchar(30)
Adress#	int

# Langage de manipulation des données - SELECT

## Exemples

On désire lire les noms rangés par ordre alphabétique de toutes les personnes qui se prénomment Lisa.

```
SELECT Nom FROM Personne
WHERE Prenom = 'Lisa' ORDER BY 1
```

On désire lire tous les noms et prénoms associés dans un champ séparés par un espace.

```
SELECT concat(Nom, ' ', Prenom) as Gens
FROM Personne ORDER BY 1
```

On désire lire les ID de toutes les personnes ayant une adresse renseignée.

```
SELECT ID FROM Personne
WHERE Adress IS NOT NULL
```

Personne	
<u>ID</u>	<u>int</u>
Nom	varchar(30)
Prenom	varchar(30)
Adress#	int

# Langage de manipulation des données - SELECT

Le `WHERE` permet de préciser les critères de recherche et d'associer les tables entre elles.

Tous les opérateurs `=`, `<=>`, `<`, `>`, `!=`, `>=`, `<=`, `<>`, `BETWEEN`, `IN`, `NOT IN`, `IS NULL`, `IS NOT NULL`, ... sont supportés.

Pour chercher des données contenues dans une table ainsi que dans une autre table liées par le biais d'une clé étrangère, indispensable de préciser l'égalité entre les 2 champs.

**Attention** : si toutes les tables listées dans la clause `FROM` ne sont pas associées dans la clause `WHERE`, le moteur effectuera un produit cartésien des tables non liées.

Ainsi si 3 tables de 500, 1000, et 2500 lignes sont appelées dans le `FROM` sans association dans la clause `WHERE`, le résultat sera de :

$500 * 1000 * 2500 = 1\ 250\ 000\ 000$  lignes.