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

Cours 3
Le langage SQL

UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

iut Marne-la-Vallée

Philippe Gambette

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

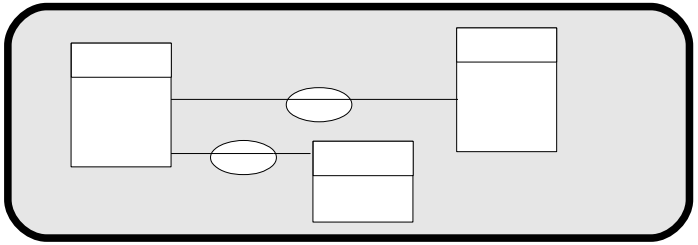
- Résumé des épisodes précédents
- Introduction au langage SQL
- Langage de définition des données
- Intermède sur PHP
- Langage de manipulation des données
- SQL avancé : les jointures
- SQL avancé : les groupements
- SQL avancé : les transactions

Plan

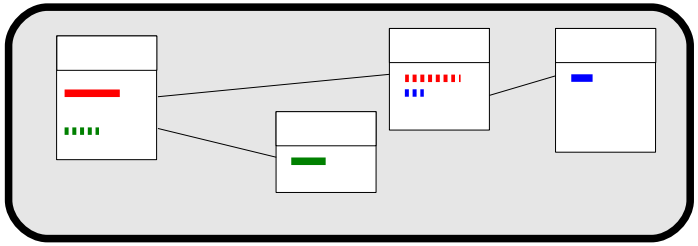
- Résumé des épisodes précédents
- Introduction au langage SQL
- Langage de définition des données
- Intermède sur PHP
- Langage de manipulation des données
- SQL avancé : les jointures
- SQL avancé : les groupements
- SQL avancé : les transactions

Modèle physique des données

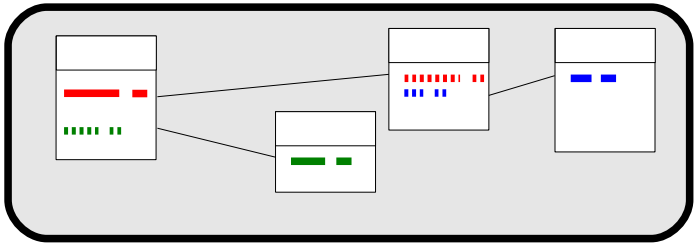
Modèle entité-association
(modèle conceptuel des données)



Modèle logique des données



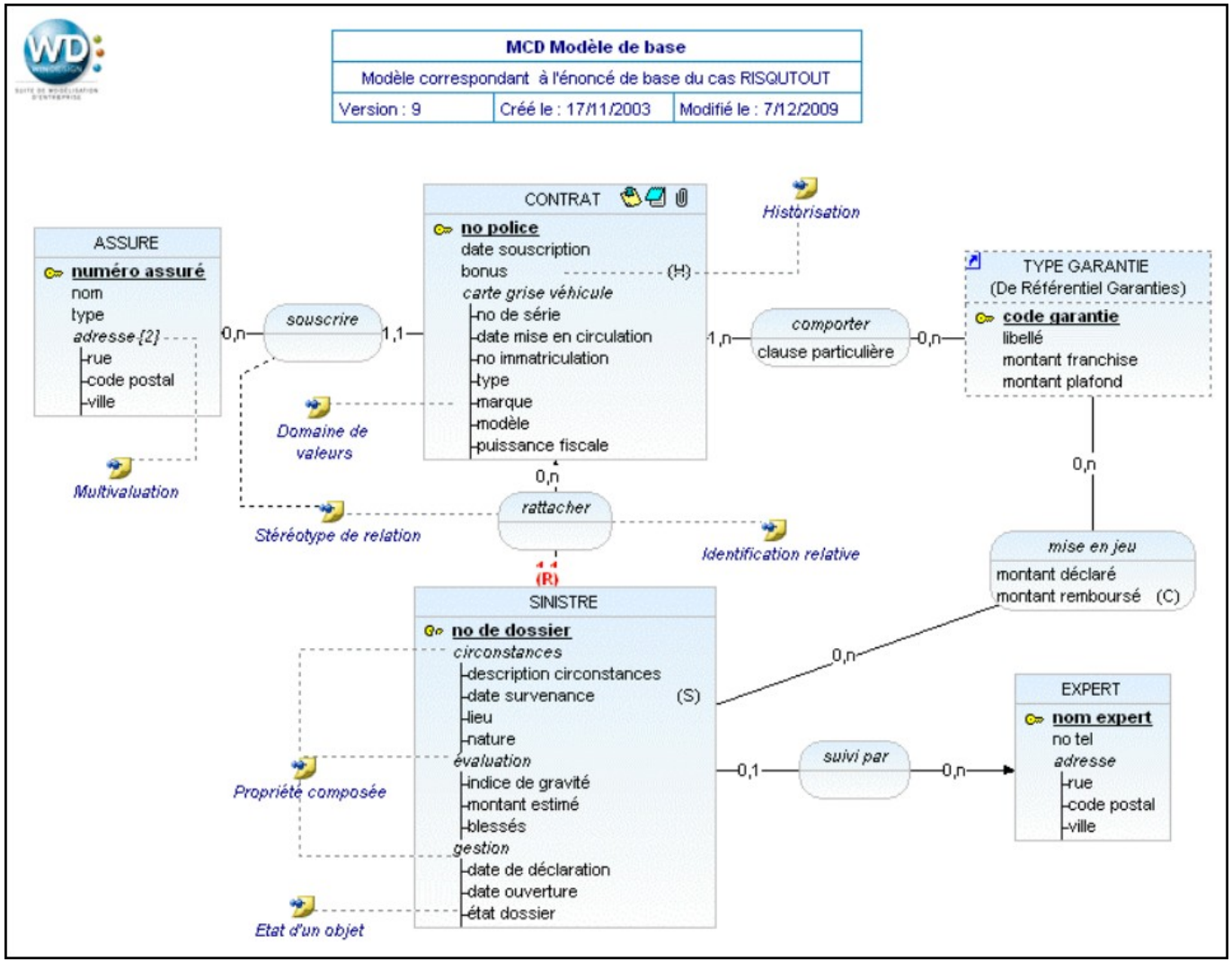
Modèle physique des données



Transformation vers le modèle logique des données

Modèle entité association → **Modèle logique des données**

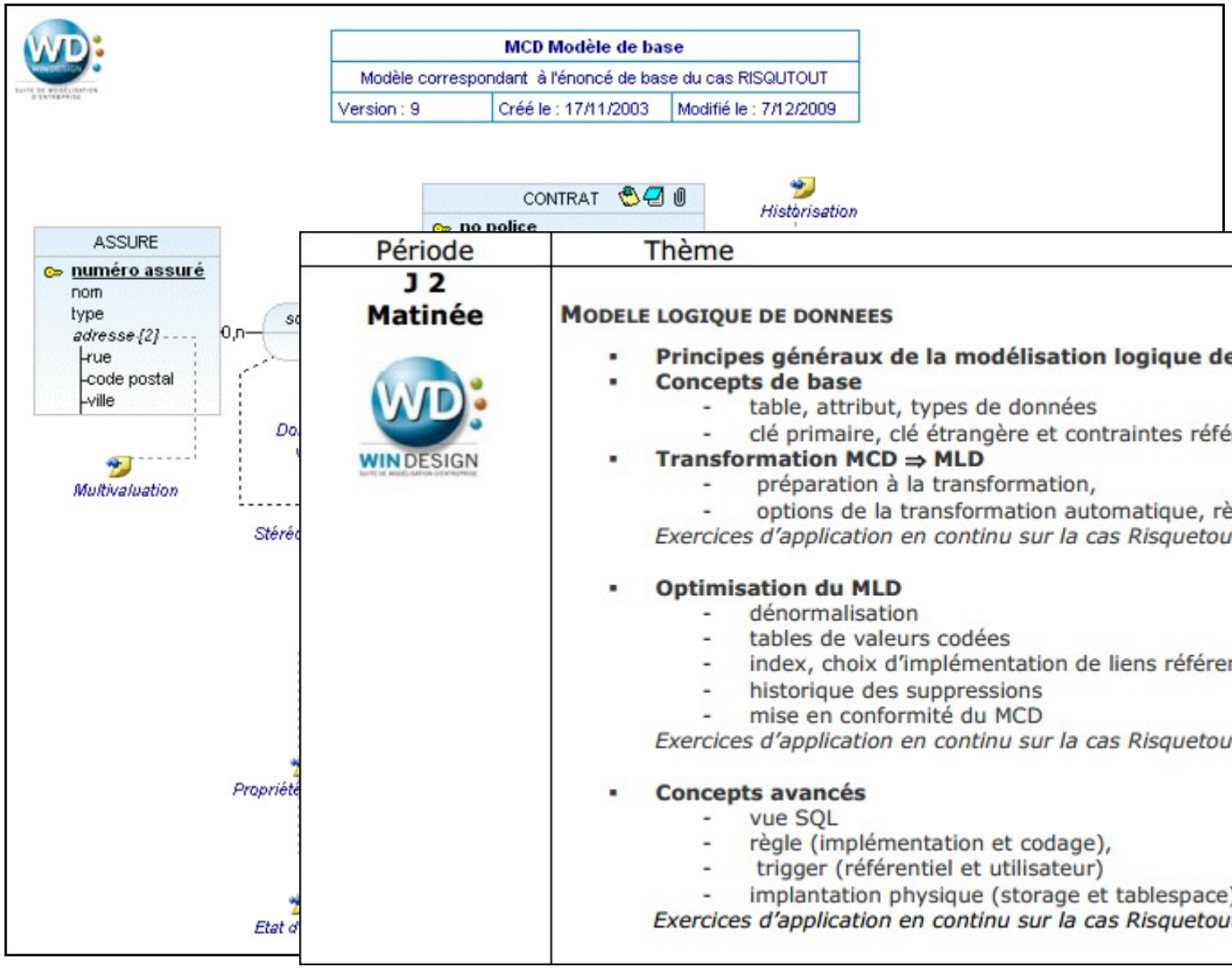
Transformation automatique : exemple de WinDesign Database



Transformation vers le modèle logique des données

Modèle entité association → **Modèle logique des données**

Transformation automatique : exemple de WinDesign Database



Période	Thème
J 2 **Matinée**	**MODELE LOGIQUE DE DONNEES** ▪ **Principes généraux de la modélisation logique des données** ▪ **Concepts de base** - table, attribut, types de données - clé primaire, clé étrangère et contraintes référentielles, index ▪ **Transformation MCD ⇒ MLD** - préparation à la transformation, - options de la transformation automatique, règles de nommage, *Exercices d'application en continu sur la cas Risquetout* ▪ **Optimisation du MLD** - dénormalisation - tables de valeurs codées - index, choix d'implémentation de liens référentiels - historique des suppressions - mise en conformité du MCD *Exercices d'application en continu sur la cas Risquetout* ▪ **Concepts avancés** - vue SQL - règle (implémentation et codage), - trigger (référentiel et utilisateur) - implantation physique (storage et tablespace) *Exercices d'application en continu sur la cas Risquetout*

http://www.win-design.com/en/Documentation/WD_DataBase.pdf

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.

http://dev.mysql.com/doc/refman/5.0/fr/

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

http://dev.mysql.com/doc/refman/5.0/fr/

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.

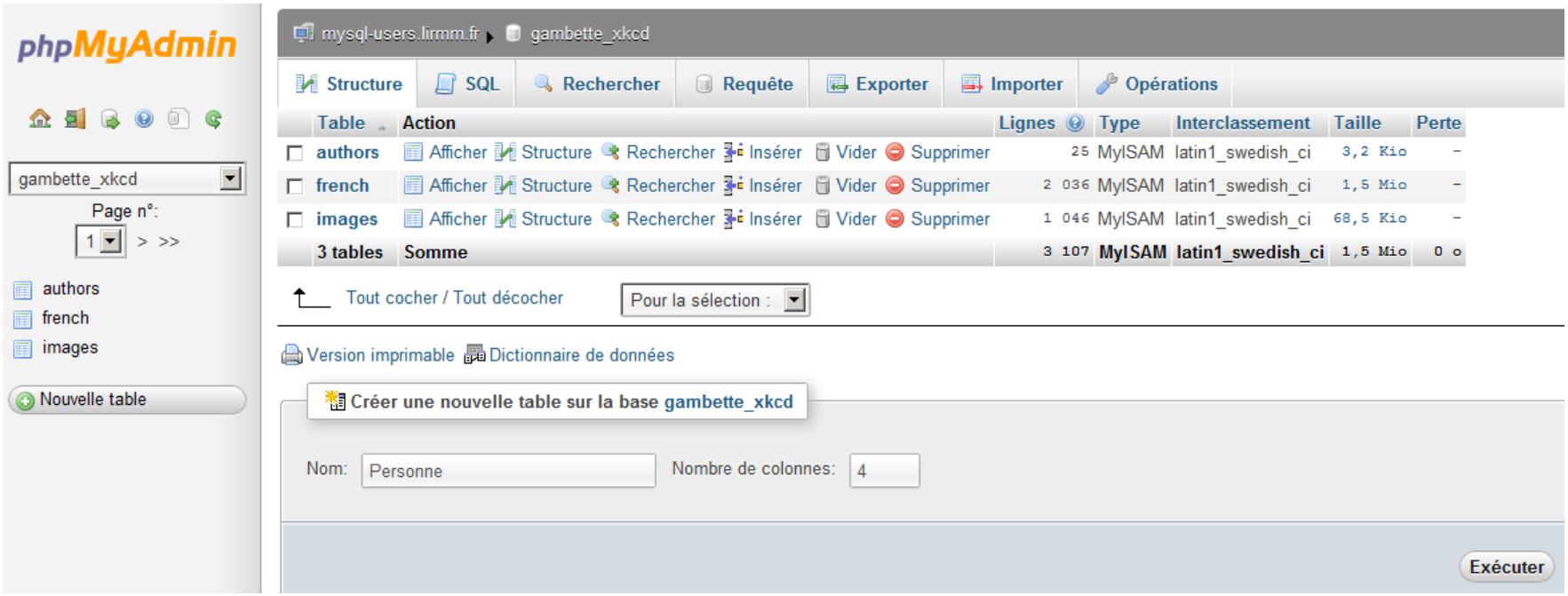
Attention, certaines syntaxes ou fonctions sont propres au système de base de données utilisé.

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

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

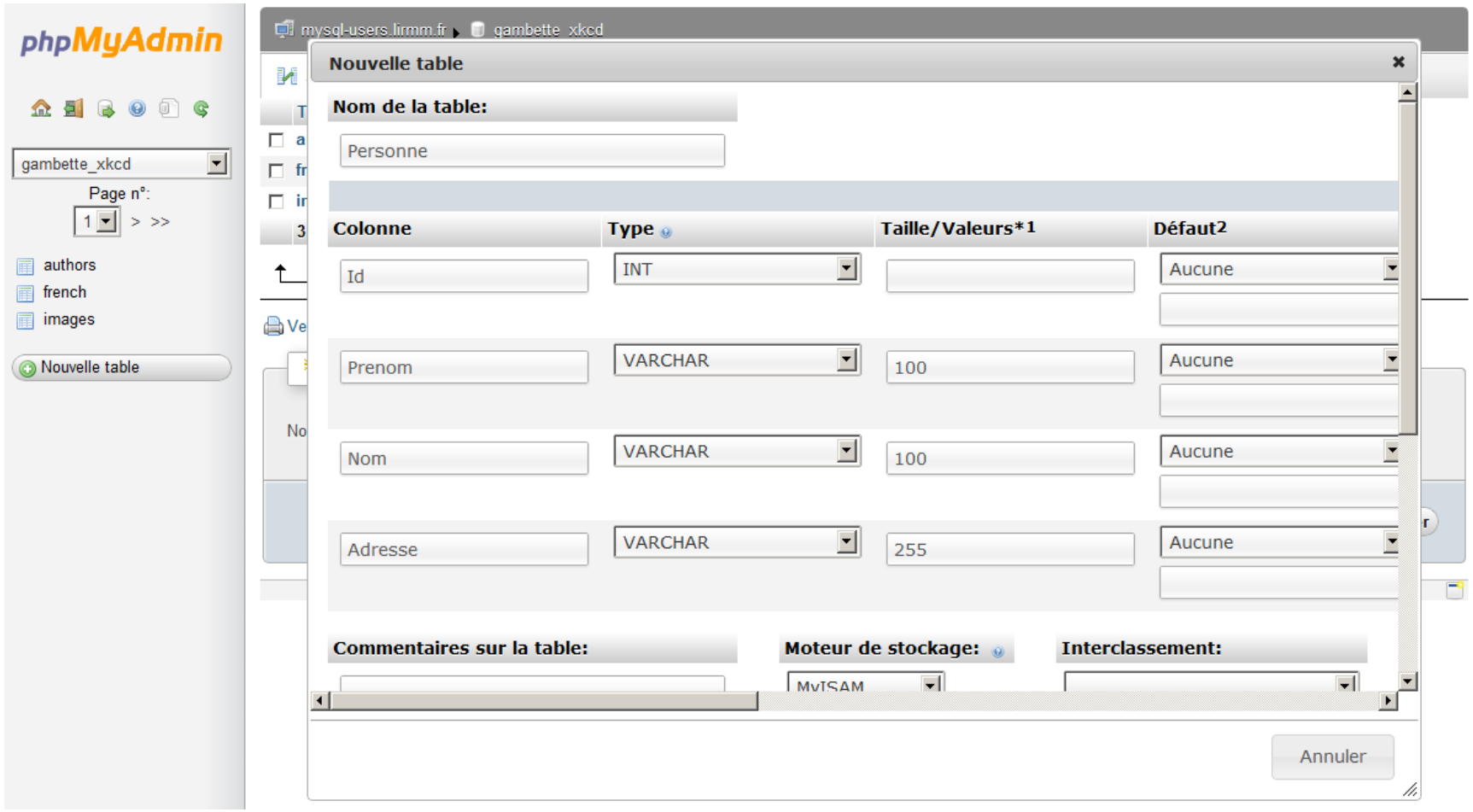
http://dev.mysql.com/doc/refman/5.0/fr/

Création d'une table avec PhpMyAdmin



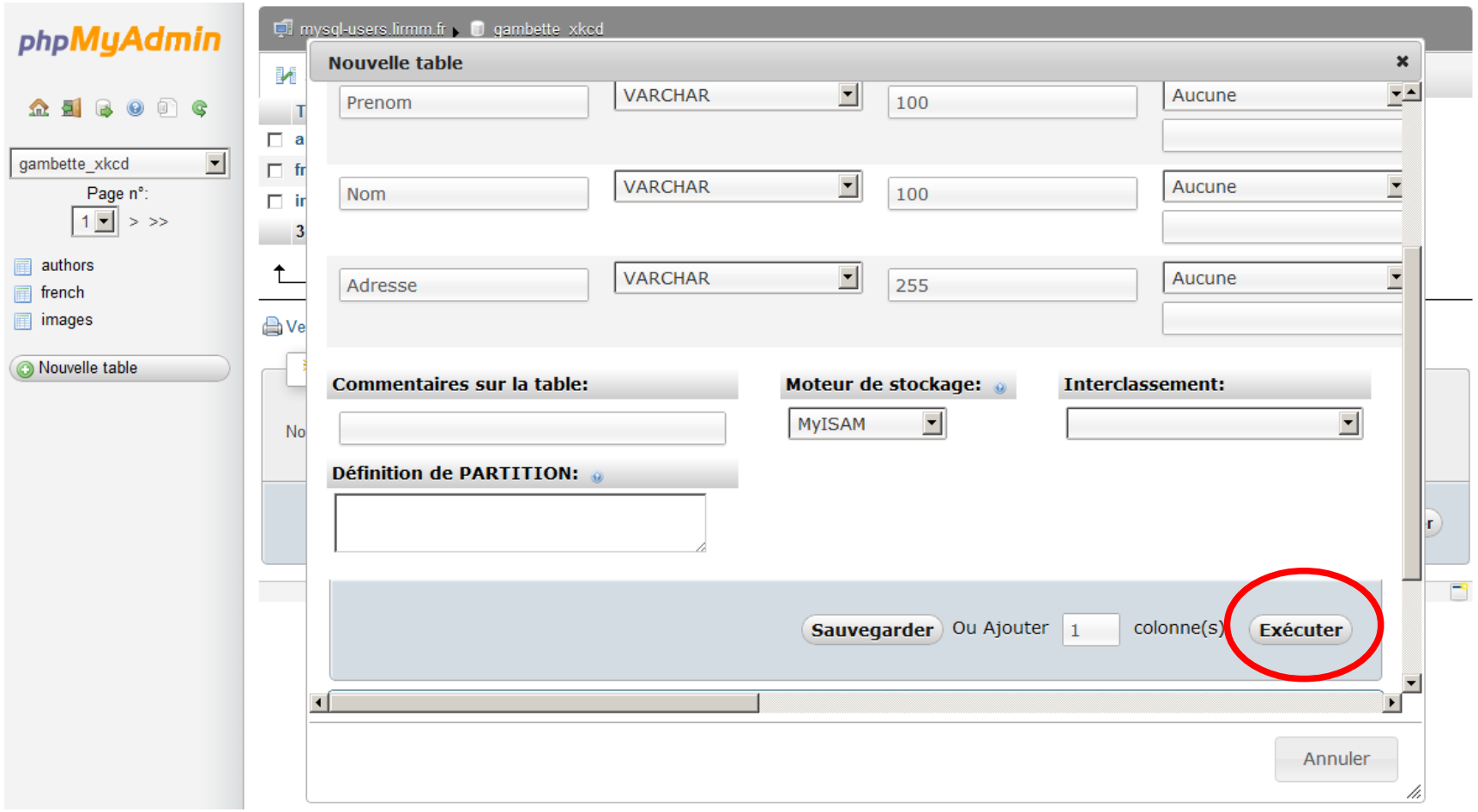
http://dev.mysql.com/doc/refman/5.0/fr/

Création d'une table avec PhpMyAdmin



http://dev.mysql.com/doc/refman/5.0/fr/

Création d'une table avec PhpMyAdmin



http://dev.mysql.com/doc/refman/5.0/fr/

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

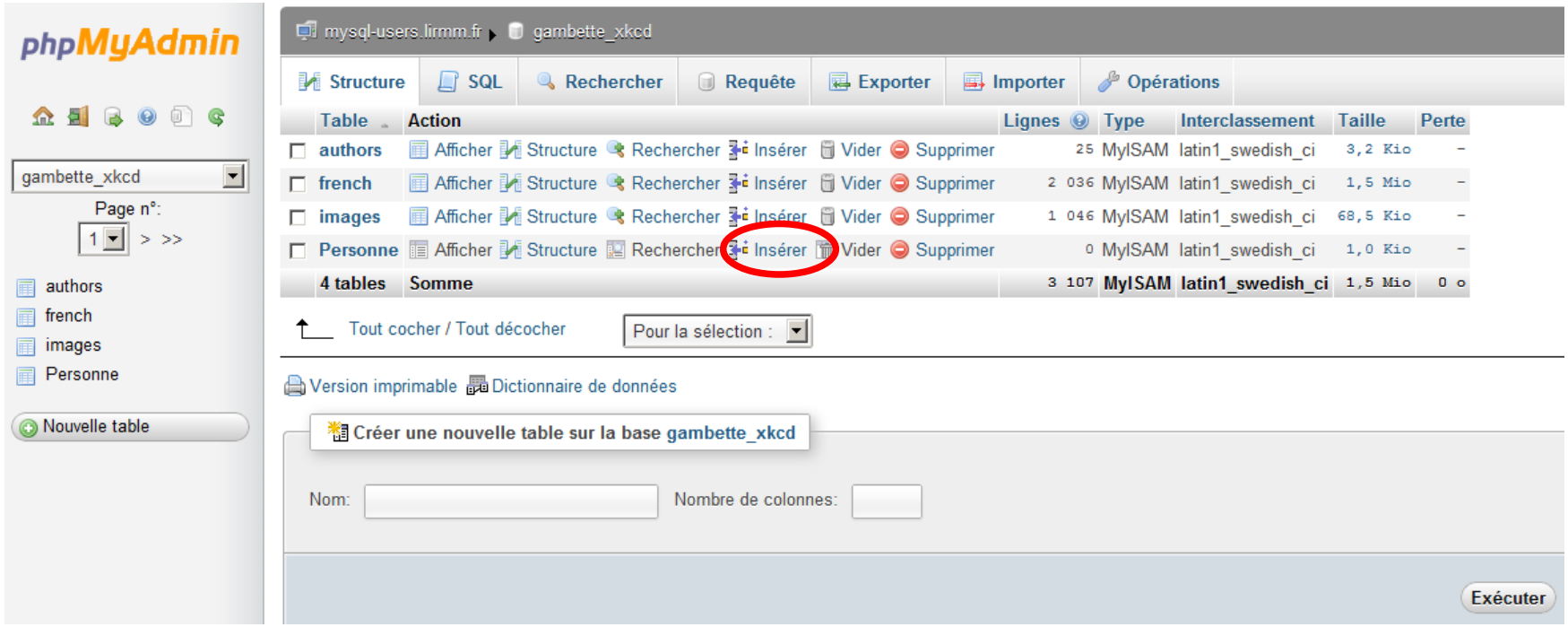
- Résumé des épisodes précédents
- Introduction au langage SQL
- Langage de définition des données
- Intermède sur PHP
- Langage de manipulation des données
- SQL avancé : les jointures
- SQL avancé : les groupements
- SQL avancé : les transactions

Langage de manipulation des données

Les commandes principales sont :

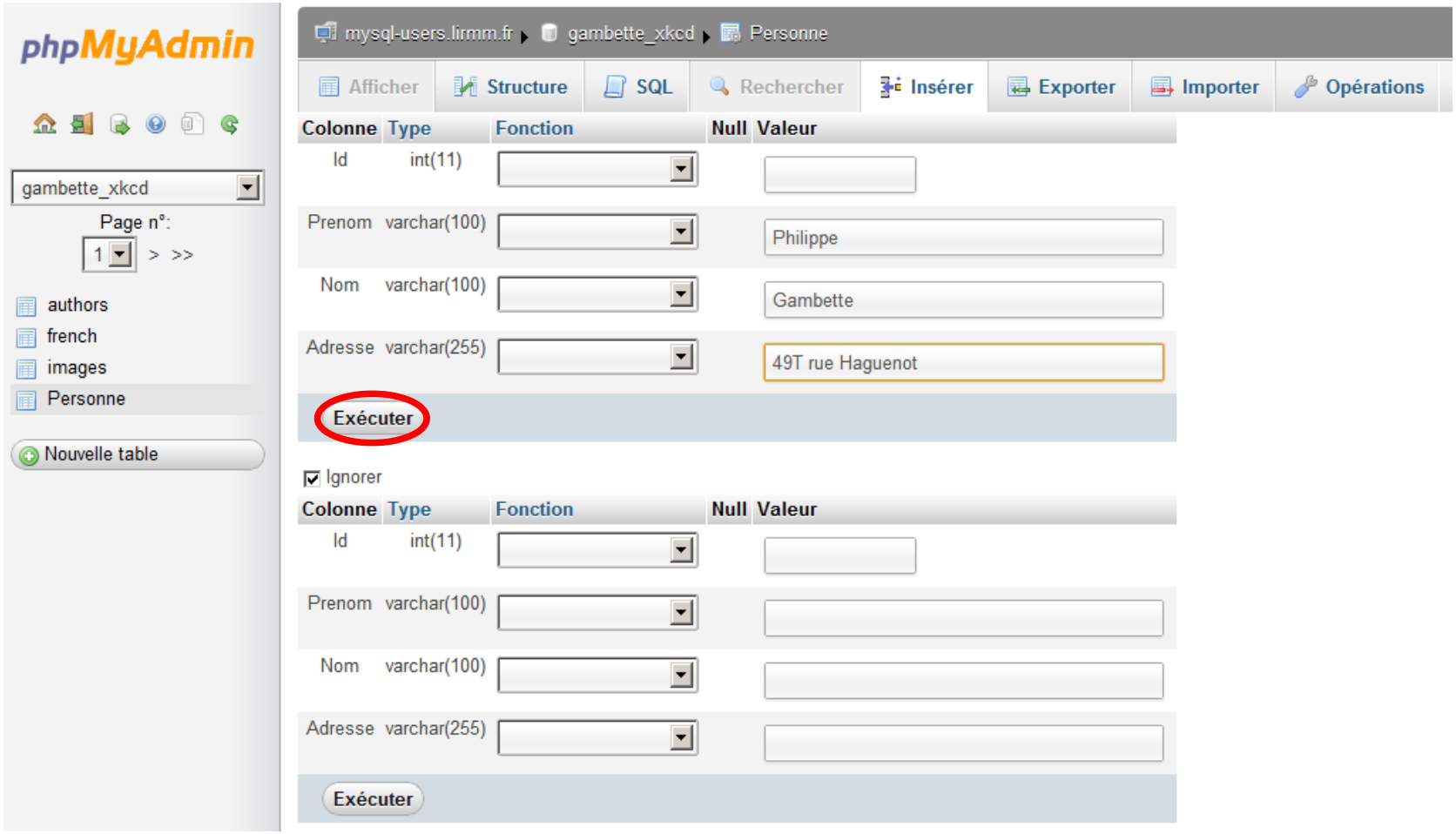
- `INSERT` pour **ajouter** les données
- `UPDATE` pour **modifier** les données
- `DELETE` pour **supprimer** les données
- `SELECT` pour **consulter** les données

Insertion d'occurrences dans une table avec PhpMyAdmin



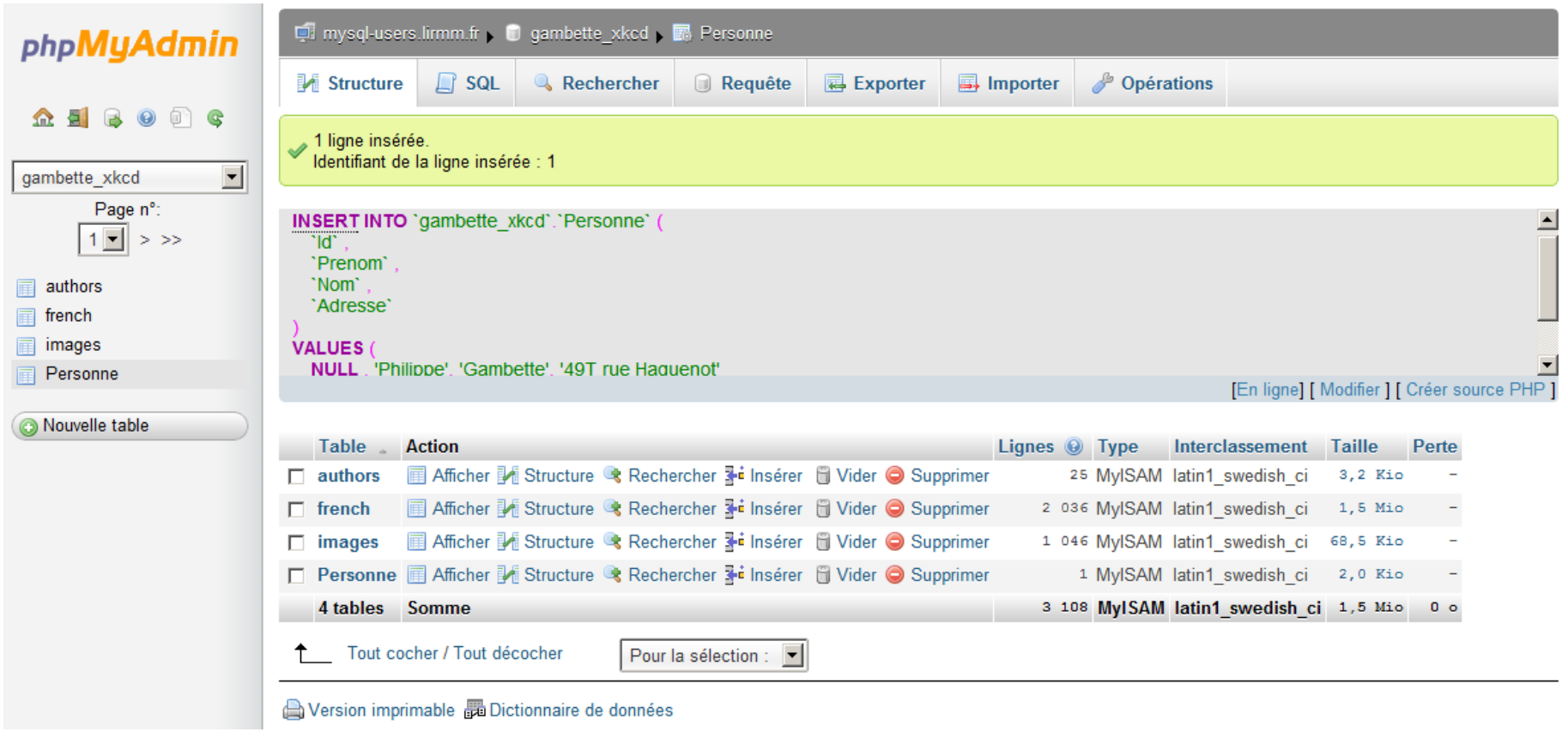
http://dev.mysql.com/doc/refman/5.0/fr/

Insertion d'occurrences dans une table avec PhpMyAdmin



http://dev.mysql.com/doc/refman/5.0/fr/

Insertion d'occurrences dans une table avec PhpMyAdmin



http://dev.mysql.com/doc/refman/5.0/fr/

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`.

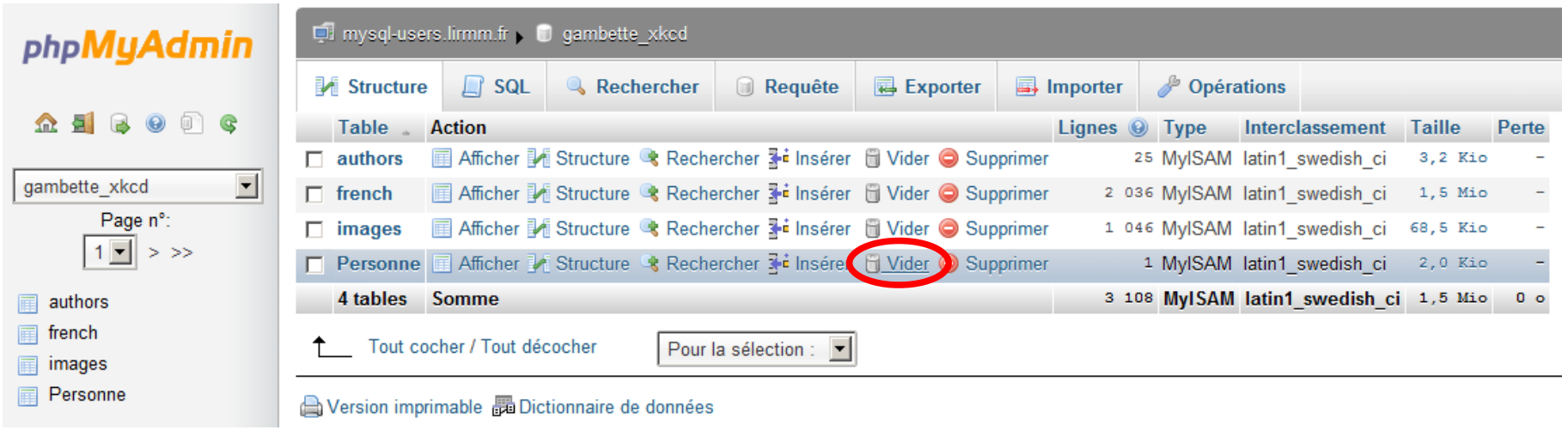
Langage de manipulation des données - DELETE

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`.



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	
ID	int
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
```

Personne	
ID	int
Nom	varchar(30)
Prenom	varchar(30)
Adress#	int

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

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
```

Personne	
ID	int
Nom	varchar(30)
Prenom	varchar(30)
Adress#	int

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.

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
```

Personne	
ID	int
Nom	varchar(30)
Prenom	varchar(30)
Adress#	int

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
```

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.