



# Formation Développeur Expert Java

SQL avec MySQL

# Module SQL

1. **Structure d'une base**
2. Installation de MySQL
3. Interrogation de données
4. Requêtes multi-tables
5. Requêtes complexes
6. Manipulation de données
7. Langage de définition de données
8. Développement de fonctions
9. Triggers
10. Transactions

- Qu'est-ce qu'une base de données
- Les objets d'une base de données
- Les tables
- Notion d'intégrité de données
- MySQL / MariaDB
- Les moteurs de MariaDB
- Les types de données

# 1. Structure d'une base

# Module SQL

Objectif :

- Décrire les composants d'un SGBDR
  - ◆ Qu'est-ce qu'une base de données relationnelle
  - ◆ Les objets d'une base de données relationnelle
  - ◆ Les tables et leurs caractéristiques
  - ◆ La notion d'intégrité de données

# Module SQL

Qu'est-ce qu'une base de données ?

- Permet de stocker la description des objets (par exemple les tables)
- Permet de stocker des données dans une ou plusieurs tables
- Permet de gérer l'accès aux données
- Permet d'assurer l'intégrité des données

# Module SQL

Les objets d'une base de données ?

- Tables
- Index
- Vues
- Séquences
- Fonctions & procédures
- Triggers

# Module SQL

## Les tables et leurs caractéristiques

- Permettent de stocker les données
- Les tables possèdent un ensemble de colonnes
- Chaque colonne comporte un nom et un type de données

# Module SQL

## MySQL / MariaDB

- MySQL a été racheté par Oracle en 2009
- MariaDB est un "fork" de MySQL suite à ce rachat



# Module SQL

Les moteurs de MySQL et MariaDB

MySQL possède plusieurs moteurs de base de données pour gérer les table.

- MyISAM
  - Très rapide
  - Inconvénient : pas de transactions
  - Inconvénient : Pas de clés étrangères
- InnoDB
  - Moteur par défaut sur MariaDB
  - Gère les transactions
  - Gère les clés étrangères et les contraintes d'intégrité
  - Système de récupération lors d'un crash par relecture de log
  - Plus lent

Le moteur est spécifié pour chaque table

# Module SQL

## Type de données

chaîne de caractères	<code>varchar(n)</code> <code>text</code>
Entier	<code>tinyint</code> (1 octet : -127 à 127 ) <code>smallint</code> (2 octets : -32768 à 32768) <code>mediumint</code> (3 octets -8388608 à 8388608) <code>int</code> (4 octets : -2 milliards à 2 milliards ) <code>bigint</code> (8 octets)
Numérique	<code>decimal</code> (nb_digit, nb_decimal) <code>float</code> (4 octets) : attention, stockage de valeur approximative <code>double</code> (8 octets) : attention, stockage de valeur approximative
Date & time	<code>date</code> <code>time</code> <code>datetime</code> <code>timestamp</code>

# Module SQL

## Notion d'intégrité de données

- Une colonne peut être NULL / NOT NULL
- Une colonne est définie par un type de données et une taille

# Module SQL

## Ordre de tri et comparaison (Collation)

- MySQL gère plusieurs jeux de caractère
  - latin1 : européen (1 caractère = 1 octet)
  - utf8 : international
- MySQL gère plusieurs ordre de tri
  - ci : case insensitive (pas de différences entre majuscules / minuscules / accents)
  - cs : case sensitive
  - bin : différence stricte
- Ces 2 éléments forment la collation :
  - latin1\_general\_ci
  - latin1\_general\_cs
  - utf8\_general\_ci
  - utf8\_bin

# Module SQL

1. Structure d'une base
2. **Installation de MySQL**
3. Interrogation de données
4. Requêtes multi-tables
5. Requêtes complexes
6. Manipulation de données
7. Langage de définition de données
8. Développement de fonctions
9. Triggers
10. Transactions

- Installation de MariaDB
- Installation de MySQLWorkbench
- Création d'une base

# 2. Installation de MySQL

# Module SQL

Objectif :

- Savoir installer MariaDB
- Savoir se connecter au serveur
- Savoir créer une base
- Savoir y importer des données

# Module SQL

Télécharger MariaDB sur le site de la fondation

→ <https://downloads.mariadb.org/>

→ Prendre la dernière version winx64.msi

Le package comprend le moteur MariaDB et l'outil HeidiSQL (permettant d'effectuer des requêtes



# Module SQL

## Exercice 1

- Installer MariaDB
  - ◆ Télécharger l'installer
  - ◆ Installer MariaDB
  - ◆ Laisser tous les paramètres par défaut

# Module SQL

## Exercice 2

- Se connecter avec HeidiSQL
- ◆ Nom de session : localhost
  - ◆ Hôte : 127.0.0.1
  - ◆ Utilisateur : root
  - ◆ Mot de passe :
  - ◆ Port :3306

# Module SQL

## Exercice 3

- Créer une base et y importer les données pour les TPs
  - ◆ Créer une base formation
    - ◆ *Collation : utf8\_bin*
  - ◆ Importer la structure de la base
    - ◆ *Fichier -> Exécuter un fichier SQL -> formation.sql*
  - ◆ Importer les données
    - ◆ *Fichier -> Exécuter un fichier SQL -> formation\_data.sql*

# Module SQL

1. Structure d'une base
2. Installation de MySQL
3. **Interrogation de données**
4. Requêtes multi-tables
5. Requêtes complexes
6. Manipulation de données
7. Langage de définition de données
8. Développement de fonctions
9. Triggers
10. Transactions

- Select
- Opérateurs arithmétiques
- Concaténation
- Clause where
- Agrégats
  - Group By
  - Fonctions d'agrégat
- Tri
- Fonctions de chaîne
- Conversion de type
- Fonctions de date
- Fonctions mathématiques
- Expression case

# 3. Interrogation de données

# Module SQL

Objectif :

- Connaître la syntaxe basique de l'ordre SELECT
- Réaliser des extractions mono-table
- Se servir des fonctions de base

# Module SQL

## L'ordre SELECT

- Syntaxe de l'ordre SELECT
  - ◆ `SELECT col1, col2, ... FROM table`
- Toutes les colonnes : \*
- Suppression des lignes identiques
  - ◆ `SELECT DISTINCT col1, col2, ... FROM table`
- Alias
  - ◆ `SELECT col1 AS alias1 FROM table`

# Module SQL

## Tris

- Les tris sont définis par ORDER BY
- Syntaxe
  - ◆ `SELECT col1, col2 FROM table ORDER BY col1 [ASC|DESC], col2 [ASC|DESC]`
  - ◆ `SELECT col1, col2 FROM table ORDER BY n1 [ASC|DESC], n2 [ASC|DESC]`
- Exemple
  - ◆ `SELECT nom, prenom FROM abonne ORDER BY nom ASC`
  - ◆ `SELECT nom, prenom FROM abonne ORDER BY 1 ASC`



# Module SQL

## Clause WHERE

- Permet de conditionner la sélection des lignes
- Syntaxe
  - ◆ `SELECT col1,col2 FROM table WHERE condition`

## Clauses LIMIT et OFFSET

- Permet de limiter le nombre de lignes renvoyées
- Syntaxe
  - ◆ `SELECT col1,col2 FROM table LIMIT n OFFSET n`

# Module SQL

## Opérateurs logiques

Egal	=
Différent	<> ou !=
Inférieur	<
Supérieur	>
Inférieur ou égal	<=
Supérieur ou égal	>=
Faisant parti d'une liste de valeur	IN (val1, val2, ..)
Comprise entre 2 valeurs	BETWEEN val1 AND val2
Nulle	IS NULL (et pas = NULL)
Recherche sur une partie d'un mot (joker % et _)	LIKE

# Module SQL

## Négation et combinaisons

- Négation : NOT
- Combinaisons
  - ◆ Et : AND
  - ◆ Ou : OR
- ◆ Utilisation des parenthèses pour prioriser les combinaisons

# Module SQL

## Exercice 4

- Récupérer tous les abonnés habitant à MONTPELLIER et les trier par ordre alphabétique de nom et prénom.
- Afficher les colonnes nom, prénom et ville

# Module SQL

## Exercice 5

→ Lister tous les prénoms différents qui commencent par la lettre L

# Module SQL

## Exercice 6

→ Lister tous les prénoms différents qui commencent par la lettre L ou la lettre M

# Module SQL

## Exercice 7

- Récupérer tous les abonnés habitant à MONTPELLIER dont le prénom commence par J et les trier par ordre alphabétique de nom et prénom.
- Afficher les colonnes nom, prenom et ville

# Module SQL

## Opérateurs arithmétiques

- Addition : +
- Soustraction : -
- Multiplication : \*
- Division : /



# Module SQL

## Opérateurs de concaténation de chaîne

- `concat(val1, val2)`
- `concat(val1, val2, val3, valn)`

# Module SQL

## Fonctions de chaîne

Longueur de chaîne	length(chaine)
Conversion en minuscule	lower(chaine)
Conversion en majuscule	upper(chaine)
Partie de chaîne	substring(chaine, debut, nombre)
Début d'une chaîne	left(chaine, nombre)
Fin d'une chaîne	right(chaine, nombre)
Suppression des espaces de début et fin	trim(chaine)
Position d'une sous-chaîne	strpos(chaine, sous-chaine)
Remplacement d'une sous-chaîne	replace(chaine, sous-chaine, nouv_sous-chaine)

# Module SQL

## Exercice 8

- Récupérer tous les abonnés habitant à Montpellier sans tenir compte des majuscules et minuscules et les trier par ordre alphabétique.
- Afficher le résultat avec une seule colonne avec sous la forme "NOM Prénom"

# Module SQL

## Fonctions de date

Date & heure système	<code>now()</code> : date/heure courante <code>curdate()</code> : date courante <code>current_date</code> / <code>current_date()</code> : synonyme de <code>curdate()</code> <code>curtime()</code> : heure courante <code>current_time</code> / <code>current_time()</code> : synonyme de <code>curtime()</code>
Partie d'une date	<code>date(datetime)</code> : extrait la date d'un datetime <code>time(datetime)</code> : extrait l'heure dun datetime <code>extract (partie from date)</code> ex : <code>extract( year from now() )</code> <code>year(datetime)</code> : année de la date <code>month(datetime)</code> : mois de la date <code>day(datetime)</code> : jour de la date <code>weekofyear(datetime)</code> : semaine de la date

# Module SQL

## Conversion de date

Date vers numérique	<code>to_days(date)</code> : converti en nombre de jour depuis l'an 0 <code>to_seconds(date)</code> : converti en nombre de secondes depuis l'an 0 <code>unix_timestamp(date)</code> : converti la date en timestamp
Numérique vers date	<code>from_days(valeur)</code> : converti un nombre de jour depuis l'an 0 en date <code>from_unixtime(valeur)</code> : converti un timestamp en date

# Module SQL

## Calculs sur les dates

- MySQL ne permet pas de faire des calculs directement sur les dates
- On peut par contre faire des calculs en convertissant la date en jour, secondes ou timestamp
- Exemple :
  - ◆ `SELECT to_days(current_date) + 5`
    - *Retourne la date courante + 5 jours en nombre de jour depuis l'an 0*
  - ◆ `SELECT from_days(to_days(current_date) + 5)`
    - *Retourne la date courante + 5 jours*
  - ◆ `SELECT to_days(current_date) - to_days('2018-11-01')`
    - *Retourne le nombre de jour depuis le 01/11/2018*

# Module SQL

## Fonctions de calculs sur des dates

Ajout / suppression à une date	<code>date_add(date, interval valeur unité)</code> <code>date_sub(date, interval valeur unité)</code> ex : <code>date_add(now(), interval 10 day)</code>
Différence entre 2 dates	<code>datediff(date1, date2)</code> : nombre de jours entre les 2 dates <code>timestampdiff(unit, date1, date2)</code> nombre de unit entre les 2 dates ex : <code>timestampdiff(month, date1, now())</code>

# Module SQL

## Exercice 9

- Récupérer tous les abonnés dont l'abonnement est valide (date\_fin\_abo supérieure ou égale à la date courante).



# Module SQL

## Exercice 10

→ Récupérer tous les abonnés ayant moins de 20 ans

# Module SQL

## Exercice 11

→ Récupérer tous les abonnés dont l'anniversaire tombe le mois courant

# Module SQL

## Exercice 12

→ Récupérer tous les abonnés dont l'anniversaire tombe demain

# Module SQL

## Exercice 13

- ➔ Récupérer tous les abonnés qui se sont abonnés la semaine dernière (de lundi de la semaine dernière à dimanche dernier)

# Module SQL

## Agrégats

- Utilisation de GROUP BY pour regrouper les données
- Syntaxe
  - ◆ `SELECT col1, fonction(col2) FROM table GROUP BY col1`
- Explication
  - ◆ Retourne le résultat de fonction sur la colonne 2 pour toutes les valeurs distinctes de col1

# Module SQL

## Fonctions d'agrégat

- Moyenne : AVG
- Nombre d'enregistrements : COUNT
- Valeur la plus grande : MAX
- Valeur la plus petite : MIN
- Total : SUM
- Exemple
  - ◆ `select count(*) from abonne`
  - ◆ `select ville, count(id) from abonne group by ville`
  - ◆ Renvoi la liste des villes avec pour chacune le nombre d'abonné

# Module SQL

## Condition Having

- Permet de filtrer des résultats en appliquant un filtre sur un agrégat
- Exemple
  - ◆ `select ville, count(id) from abonne group by ville having count(id) > 10`
  - ◆ Renvoi la liste des villes ayant plus de 10 abonné avec pour chacune le nombre d'abonné

# Module SQL

## Exercice 14

→ Compter le nombre d'abonnés



# Module SQL

## Exercice 15

→ Compter le nombre d'abonnés entre 30 et 40 ans

# Module SQL

## Exercice 16

→ Afficher le nombre d'abonné pour chaque ville et trier par ordre descendant

# Module SQL

## Exercice 17

→ Limiter le résultat précédant aux villes ayant au moins 20 abonnés

# Module SQL

## Expression Case

→ Permet de conditionner la valeur renvoyée

→ Syntaxe :

- ◆ CASE expression

  - WHEN valeur1 THEN resultat1

  - WHEN valeur2 THEN resultat2

  - ELSE resultat

  - END

- ◆ CASE

  - WHEN condition1 THEN resultat1

  - WHEN condition2 THEN resultat2

  - ELSE resultat

  - END

# Module SQL

## Expression Case

### → Exemples :

- ◆ 

```
SELECT CASE sexe
  WHEN 'F' THEN concat('Madame ', nom)
  WHEN 'M' THEN concat('Monsieur ', nom)
  ELSE nom END
FROM personnes ;
```
- ◆ 

```
SELECT CASE
  WHEN sexe = 'F' THEN concat('Madame ', nom)
  WHEN sexe = 'M' THEN ('Monsieur ', nom)
  ELSE nom END
FROM personnes ;
```

# Module SQL

## Conversion de type

- MySQL effectue de conversion implicite quand il le peut
  - ◆ exemple `select concat(1, ' mot')` retournera une chaîne
- Fonction CAST
- Syntaxe
  - ◆ `cast(valeur as type)`
- Exemples
  - ◆ `SELECT cast('2015-01-15' as date)`

# Module SQL

## Fonctions de formatage

Date vers texte	<code>date_format( date, format )</code>
Numérique vers texte	<code>format( number, nb_decimal )</code>
Texte vers date	<code>str_to_date( text, format )</code>
Vers un timestamp	<code>to_timestamp(text, format)</code> <code>to_timestamp(number)</code>

→ format est une chaîne contenant un format d'affichage

→ Exemple :

◆ `str_to_date( '01/12/2015', '%d/%m/%Y' )`

◆ `date_format( now(), '%H:%i' )`

# Module SQL

## Exercice 18

- Récupérer les abonnés dont le nom commence par A et afficher leur statut d'abonnement sous la forme "abonné jusqu'au dd/mm/yyyy" ou "expiré"



# Module SQL

## Exercice 19

→ Compter le nombre de membre de chaque famille (même nom)

# Module SQL

## Exercice 20

- Compter le nombre de membre de chaque famille (même nom) et retourner pour chacune la date de naissance du plus jeune et du plus vieux

# Module SQL

## Exercice 21

- Compter le nombre de membre de chaque famille (même nom) et retourner pour chacune l'âge du plus jeune et du plus vieux en années

# Module SQL

## Traitement de la valeur nulle

→ Fonction COALESCE

→ Syntaxe

- ◆ COALESCE( val1, val2, val3, ...)

- ◆ Renvoi la 1ere valeur non nulle

→ Exemple

- ◆ `SELECT COALESCE( libelle_long, libelle, 'vide') FROM table1`

# Module SQL

## Fonctions mathématiques

Modulo	% mod mod()
Valeur absolue	abs()
Division entière	div
Arrondi	round(valeur, precision) ex : round(42.9876, 2) : 42.99
Troncature	truncate(valeur, precision) ex : truncate(42.9876, 2) : 42.98
Valeur aléatoire	random()
Signe	sign(valeur) ex : sign(-12345) : -1 sign(123345) : 1

# Module SQL

## Exercice 22

→ Calculer le nombre d'abonné par tranche d'âge (10-20 ans, 20-30 ans, ...)

# Module SQL

1. Structure d'une base
2. Installation de MySQL
3. Interrogation de données
4. **Requêtes multi-tables**
5. Requêtes complexes
6. Manipulation de données
7. Langage de définition de données
8. Développement de fonctions
9. Triggers
10. Transactions

- Notion de jointure
- Produit cartésien
- Jointure
- Jointure externe

# 4. Requêtes multi-tables



# Module SQL

Objectif :

- Comprendre la notion de jointure
- Réaliser des extractions sur plusieurs tables

# Module SQL

## Relation entre les tables

→ Relation 1 à plusieurs

table : edition	
<u>noEdition</u>	clé primaire
edition	
adresse	
telephone	

relation  
1 à n



- une maison d'édition peut avoir plusieurs livre
- un livre à qu'une seule maison d'édition

table : livre	
<u>no</u>	clé primaire
titre	
sujet	
auteur	
pages	
noEdition	clé étrangère

# Module SQL

## Relation entre les tables

table : eleve	
<u>noEleve</u>	clé primaire
nom	
prenom	
annee	

relation  
m à n



- un élève a plusieurs cours
- un cours a plusieurs élèves

table : cours	
<u>cote</u>	clé primaire
titreCours	
description	

table : eleve	
<u>noEleve</u>	clé primaire
nom	
prenom	
annee	

2 relations 1 à n

1 à n

n à 1



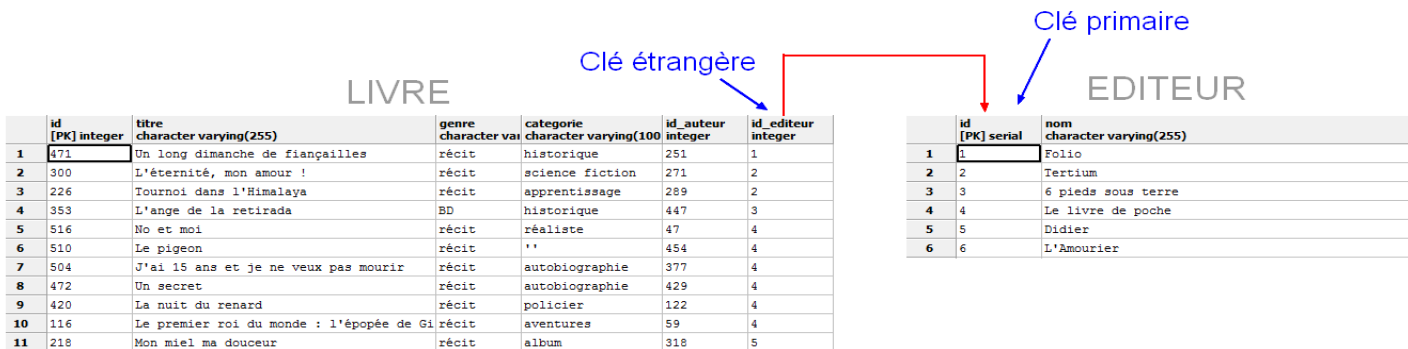
table : suitCours	
<u>noEleve</u>	clé primaire
<u>cote</u>	clé primaire

table : cours	
<u>cote</u>	clé primaire
titreCours	
description	

# Module SQL

## Notion de jointure

→ Une jointure permet de “lier” 2 tables



→ Tous les éditeurs sont stockés dans la table “editeur”

→ Chaque livre est lié à un éditeur

# Module SQL

## Produit cartésien

- Il s'agit d'une jointure sans condition
- Syntaxe :
  - ◆ `SELECT champ1, champ2`  
`FROM table1 AS alias1, table2 AS alias2, ...`
- le résultat est la fusion des 2 tables :
  - ◆ chaque ligne de table1 est répétée autant de fois qu'il y a de ligne dans table2

# Module SQL

## Jointure avec condition

→ Syntaxe non normalisée :

- ◆ `SELECT champ1, champ2`  
`FROM table1, table2`  
`WHERE table1.colonne = table2.colonne ;`

→ Syntaxe normalisée :

- ◆ `SELECT champ1, champ2`  
`FROM table1 JOIN table2 ON table1.colonne = table2.colonne`

# Module SQL

## Jointure externe

- Cela permet d'afficher les lignes d'une table qui ne correspondent pas à la condition de jointure.
- Par exemple quand le champ de jointure est vide
- Syntaxe
  - ◆ `SELECT champ1, champ2`  
`FROM table1 LEFT OUTER JOIN table2 ON table1.colonne = table2.colonne`

# Module SQL

## Exercice 23

→ Lister tous les livres avec leur auteur



# Module SQL

## Exercice 24

→ Lister tous les livres avec leur auteur et leur éditeur

# Module SQL

## Exercice 25

→ Afficher la liste des éditeurs avec le nombre de livre de chacun

# Module SQL

## Exercice 26

→ Lister tous les livres actuellement empruntés avec leur emprunteur

# Module SQL

## Exercice 27

→ Trouver les abonnés ayant emprunté un livre depuis plus de 2 mois et ne l'ayant pas rendu

# Module SQL

## Exercice 28

→ Lister tous les livres dont la dernière date d'emprunt date de plus de 2 ans

# Module SQL

## Exercice 29

→ Trouver les 10 abonnés ayant emprunté le plus de livres (rendus et non rendus)

# Module SQL

## Exercice 30

→ Lister les catégories les plus lues

# Module SQL

## Exercice 31

→ Trouver les abonnés ayant plusieurs livres en leur possession



# Module SQL

## Exercice 32

→ Trouver le nombre de livre empruntés pour chaque année

# Module SQL

## Exercice 33

→ Trouver l'âge moyen des abonnés

# Module SQL

## Exercice 34

→ Trouver la ville ayant la moyenne d'âge la plus petite

# Module SQL

## Exercice 35

→ Lister tous les abonnés qui ont des homonyme (même nom et prénom)

# Module SQL

## Exercice 36

→ Trouver l'âge moyen des lecteurs de chaque catégorie

# Module SQL

## Exercice 37

→ Trouver l'âge moyen des lecteurs des livres de chaque éditeur

# Module SQL

## Exercice 38

→ Classer les catégories de la plus lue à la moins lue pour chaque année

# Module SQL

1. Structure d'une base
2. Installation de MySQL
3. Interrogation de données
4. Requêtes multi-tables
- 5. Requêtes complexes**
6. Manipulation de données
7. Langage de définition de données
8. Développement de fonctions
9. Triggers
10. Transactions

- Opérateurs ensemblistes
- Sous-requêtes mono-ligne
- Sous-requêtes multi-lignes
- Sous-requêtes dans la clause from



# 5. Requêtes complexes

# Module SQL

Objectif :

- Écrire des requêtes utilisant les opérateurs ensemblistes
- Comprendre la syntaxe des sous-requêtes
- Écrire des requêtes utilisant des sous-requêtes mono-lignes
- Écrire des requêtes utilisant des sous-requêtes multi-lignes
- Écrire des requêtes utilisant des sous-requêtes dans la clause FROM

# Module SQL

## Les opérateur ensemblistes

### → UNION

- ◆ Permet d'obtenir un ensemble de lignes provenant de 2 requêtes
- ◆ requête1 UNION requête2

<i>nom</i>	<i>prénom</i>
Chose	Jules
Machin	Pierre
Truc	Patrick

Table1

union

<i>nom</i>	<i>prénom</i>
Pouf	Jean
Chose	Jules

Table2

=

<i>nom</i>	<i>prénom</i>
Chose	Jules
Machin	Pierre
Pouf	Jean
Truc	Patrick

Résultat

# Module SQL

## Les opérateur ensemblistes

### → INTERSECT

- ◆ Permet d'obtenir un ensemble de lignes commun à 2 requêtes
- ◆ requête1 INTERSECT requête2

<i>nom</i>	<i>prénom</i>
Chose	Jules
Machin	Pierre
Truc	Patrick

**inter**

<i>nom</i>	<i>prénom</i>
Pouf	Jean
Chose	Jules

**=**

<i>nom</i>	<i>prénom</i>
Chose	Jules

# Module SQL

## Les opérateurs ensemblistes

### → EXCEPT

- ◆ Permet d'obtenir un ensemble de lignes présent dans le résultat d'une requête, mais pas dans l'autre
- ◆ requête1 EXCEPT requête2

<i>nom</i>	<i>prénom</i>	<b>Except</b>	<i>nom</i>	<i>prénom</i>	<b>=</b>	<i>nom</i>	<i>prénom</i>
Chose	Jules		Pouf	Jean		Machin	Pierre
Machin	Pierre		Chose	Jules		Truc	Patrick
Truc	Patrick						

# Module SQL

## Les opérateur ensemblistes

- Les différentes requêtes doivent avoir le même nombre de champs
- Les champs doivent être de même type
- Les doublons sont supprimé (DISTINCT implicite)
- La clause Order By doit utiliser les numéros de colonne.

# Module SQL

## Sous requête mono-ligne

- Utilisation d'un Select renvoyant une ligne unique
- Peut être utilisé dans
  - ◆ Comme champ dans la clause SELECT
  - ◆ Comme valeur de condition dans la clause WHERE
- Syntaxe :
  - ◆ SELECT champ1, champ2, (sous-requête)  
FROM table
  - ◆ SELECT champ1, champ2  
FROM table  
WHERE champ opérateur (sous-requête)

# Module SQL

## Exercice 39

→ Lister les 10 livres les plus empruntés (avec le nombre d'emprunt)



# Module SQL

## Exercice 40

- Lister tous les abonnés avec le dernier livre qu'ils ont empruntés même s'ils n'ont jamais emprunté de livre (sous select dans le SELECT)

# Module SQL

## Sous requête multi-lignes

- Utilisation d'un Select renvoyant plusieurs ligne dans la clause WHERE
- Utilisation des opérateur
  - ◆ IN
  - ◆ NOT IN
  - ◆ EXISTS
  - ◆ NOT EXISTS
- Syntaxe :
  - ◆ SELECT champ1, champ2
  - FROM table
  - WHERE champ [NOT] IN (sous-requête)

# Module SQL

## Sous requête dans la clause FROM

- Permet d'imbriquer un SELECT dans une clause FROM
- Syntaxe :
  - ◆ SELECT champ1, champ2  
FROM (sous-requête)  
WHERE condition

# Module SQL

## Exercice 41

→ Lister les livres qui n'ont jamais été empruntés

# Module SQL

## Exercice 42

→ Trouver la ou les catégories les plus lues pour chaque année

# Module SQL

## Exercice 43

- Lister tous les abonnés avec le dernier livre qu'ils ont emprunté même s'ils n'ont jamais emprunté de livre (sous-select dans la jointure)

# Module SQL

## Exercice 44

→ Lister tous les livres des auteurs "PY Olivier", "UBAC Claire", "REES Celia"

# Module SQL

## Exercice 45

→ Compter le nombre de livre pour chaque genre



# Module SQL

## Exercice 46

→ Rechercher la catégorie la plus lue pour chaque tranche d'âge

# Module SQL

1. Structure d'une base
2. Installation de MySQL
3. Interrogation de données
4. Requêtes multi-tables
5. Requêtes complexes
- 6. Manipulation de données**
7. Langage de définition de données
8. Développement fonctions
9. Triggers
10. Transactions

- La commande INSERT
- La commande UPDATE
- La commande DELETE

# 6. Manipulation de données

# Module SQL

Objectif :

- Savoir insérer des données dans la base
- Savoir modifier des données dans la base
- Savoir supprimer des données de la base

# Module SQL

## La commande INSERT

- Permet d'insérer des données dans la base
- Toutes les colonnes
  - ◆ INSERT INTO table VALUES( valeur1, valeur2, ...)
- Colonne spécifiques
  - ◆ INSERT INTO table (col1, col2, ...) VALUES( valeur1, valeur2, ...)
  - ◆ Les colonnes non alimentées ne doivent pas être obligatoire (NOT NULL)
- Plusieurs lignes
  - ◆ INSERT INTO table (col1, col2, ...) VALUES  
(valeur1, valeur2, ...),  
(valeur1b, valeur2b, ...)

# Module SQL

## Insertion à partir d'une requête

- Permet d'insérer des lignes dans une table à partir de données provenant d'une requête
- Syntaxe :
  - ◆ `INSERT INTO table (col1, col2, ...) SELECT col1b, col2b, ... FROM table2 ;`

# Module SQL

## Exercice 47

→ Ajouter une ligne dans la table abonné avec votre nom

# Module SQL

## Exercice 48

→ Ajouter 3 personnes dans la table abonné en une seule requête



# Module SQL

## La commande UPDATE

- Permet de modifier les données d'une table
- Syntaxe :
  - ◆ UPDATE table SET col1 = valeur1, col2 = valeur2, ...  
WHERE condition ;
- Les valeurs peuvent être des valeurs fixes ou des valeurs issues de sous requêtes

# Module SQL

## Exercice 49

→ Mettre à jour la date de fin de votre abonnement à la date du jour + 1 an

# Module SQL

## La commande DELETE

- Permet de supprimer des lignes d'une table
- Syntaxe :
  - ◆ DELETE FROM table  
WHERE condition ;

# Module SQL

## Exercice 50

→ Supprimer la ligne correspondant à votre nom dans la table abonné

# Module SQL

## La commande TRUNCATE

- Permet de supprimer TOUTES les lignes d'une table
- Syntaxe :
  - ◆ TRUNCATE table ;

# Module SQL

## Exercice 51

- Essayer d'insérer une ligne dans la table livre avec un id\_auteur qui n'existe pas dans la table auteur

# Module SQL

## Exercice 52

→ Remplir la table genre à partir des différents genre présents dans la table livre

# Module SQL

## Exercice 53

- Mettre à jour la colonne `id_genre` de la table `livre` en fonction de la valeur contenue dans la colonne `genre`



# Module SQL

## Exercice 54

→ Essayer de supprimer une ligne de la table genre

# Module SQL

## Exercice 55

- Créer un nouveau genre
- Créer plusieurs livres ayant ce nouveau genre
- Supprimer tous les livres ayant ce genre (Sous-select dans la clause where)

# Module SQL

1. Structure d'une base
2. Installation de MySQL
3. Interrogation de données
4. Requêtes multi-tables
5. Requêtes complexes
6. Manipulation de données
- 7. Langage de définition de données**
8. Développement de fonctions
9. Triggers
10. Transactions

- Création de table
- Modification de table
- Suppression de table
- Les vues
- Les séquences
- Les indexes

# 7. Langage de définition de données (LDD)

# Module SQL

Objectif :

→ Connaître les commandes de manipulation de structure de table

# Module SQL

## Créer une table

→ Utilisation de la commande CREATE TABLE

→ Syntaxe :

- ◆ CREATE TABLE table  
(  
    coll type\_de\_la\_colonne contrainte\_de\_colonne,  
    ...  
);

# Module SQL

## Les contraintes de colonne

→ Se placent après le type d'une colonne

◆ NOT NULL

- *interdit la valeur nulle*

◆ AUTO\_INCREMENT

- *La valeur de la colonne sera automatiquement incrémentée*

◆ DEFAULT valeur

- *valeur par défaut du champ*

◆ CONSTRAINT nom PRIMARY KEY

- *clé primaire*

◆ CONSTRAINT nom REFERENCES table (refcol)

- *Clé étrangère*

# Module SQL

## Les contraintes de table

→ Le placent après la liste des colonnes

- ◆ `CONSTRAINT nom PRIMARY KEY (col, ...)`
- ◆ `CONSTRAINT nom FOREIGN KEY (col) REFERENCES table (col)`



# Module SQL

## Les tables temporaires

- Une table temporaire jusqu'à ce que la session expire.
- Elle est visible uniquement à partir de la session qui l'a créée.
- Syntaxe :
  - ◆ `CREATE TEMP TABLE table`  
`(`  
`coll type_de_la_colonne contrainte_de_colonne,`  
`...`  
`);`

# Module SQL

## Exercice 56

- Créer une table categorie
  - ◆ comportant un identifiant nommé id de type integer et auto\_increment et une colonne nom\_categorie de type varchar(60)
  - ◆ l'identifiant sera la clé primaire de la table

# Module SQL

## Modifier une table

- Ajout d'une colonne
  - ◆ ALTER TABLE table ADD nom\_de\_la\_colonne type\_de\_la\_colonne;
- Modification d'une colonne
  - ◆ ALTER TABLE table ALTER COLUMN nom\_de\_la\_colonne TYPE nouveau\_type;
- Suppression d'une colonne
  - ◆ ALTER TABLE table DROP nom\_de\_la\_colonne ;
- Ajout d'une contrainte
  - ◆ ALTER TABLE table ADD CONSTRAINT ...

# Module SQL

## Exercice 57

- Ajouter une colonne `id_categorie` dans la table `livre` et créer une foreign key vers la colonne `id` de la table `categorie`

# Module SQL

## Exercice 58

→ Supprimer la colonne genre de la table livre

# Module SQL

## Exercice 59

- Créer une colonne `tranche_age` de type `varchar(5)` dans la table `abonne`
- Remplir cette colonne avec la tranche d'âge de chaque abonné

# Module SQL

Supprimer une table

→ `DROP TABLE table ;`

# Module SQL

## Les vues

- Objet logique auquel est rattaché une requête SELECT
- A chaque utilisation de la vue, la requête associée est exécutée
- Les données de la vue ne sont pas stockées
- Syntaxe :
  - ◆ CREATE VIEW nom\_vue  
(col1, col2, ...)  
AS SELECT ...
- Suppression d'une vue :
  - ◆ DROP VIEW nom\_vue ;



# Module SQL

## Exercice 60

- Créer une vue renvoyant les livres, leur éditeur et leur genre
- ◆ id
  - ◆ titre
  - ◆ nom\_editeur
  - ◆ nom\_genre

# Module SQL

## Les séquences

- Compteur permettant de générer des numéros uniques
- Syntaxe :
  - ◆ CREATE SEQUENCE nom\_sequence  
START WITH valeur\_initiale  
INCREMENT val ;
- Utilisation :
  - ◆ SELECT nextval('nom\_sequence')

# Module SQL

## Les index

- S'applique à une ou plusieurs colonnes
- Permettent un accès plus rapide aux données.
- Permettent de forcer l'unicité
- Syntaxe :
  - ◆ `CREATE [UNIQUE] INDEX nom_index  
ON table (coll, ...);`

# Module SQL

## Démonstration

La base formation0 contient 3,3 M de lignes

- Faire une recherche sur un abonné par son nom et noter le temps d'exécution
  - ◆ 1,6 seconde
- Créer un index sur la colonne nom de la table abonne
  - ◆ Cela va prendre environ 10 s
- Refaire la recherche et comparer le temps d'exécution
  - ◆ 0 s

# Module SQL

## Exercice 61

- Créer un index unique sur le nom, prénom, date de naissance de la table abonne
- Essayer d'ajouter une donnée déjà existante

# Module SQL

1. Structure d'une base
2. Installation de MySQL
3. Interrogation de données
4. Requêtes multi-tables
5. Requêtes complexes
6. Manipulation de données
7. Langage de définition de données
- 8. Développement de fonctions**
9. Triggers
10. Transactions

➤ Notions sur les fonctions

# 8. Développement de fonctions

# Module SQL

Objectif :

→ Avoir des notions sur le développement de fonctions



# Module SQL

## Les fonctions personnalisées

- MySQL permet la création de fonctions personnalisées et des procédures stockées
- Reçoivent des arguments en entrée
- Les fonctions retournent une valeur
- Les procédures stockées ne renvoient rien
- Les fonctions personnalisées s'utilisent comme les fonctions natives (left, date\_part, ...)

# Module SQL

1. Structure d'une base
2. Installation de MySQL
3. Interrogation de données
4. Requêtes multi-tables
5. Requêtes complexes
6. Manipulation de données
7. Langage de définition de données
8. Développement de fonctions
- 9. Triggers**
10. Transactions

- Qu'est-ce qu'un trigger
- Types de triggers

# 9. Triggers

# Module SQL

Objectif :

→ Savoir créer des triggers

# Module SQL

## Principes

- Un trigger la possibilité d'exécuter une fonction lors d'un évènement sur la table :
  - ◆ insert
  - ◆ update
  - ◆ delete
- Le déclenchement peut être effectué avant ou après l'évènement
- Le déclenchement peut se faire une fois pour l'ensemble des lignes modifiées, ou une fois pour chaque ligne modifiée

# Module SQL

→ Syntaxe :

- ◆ CREATE TRIGGER nom\_trigger  
BEFORE or AFTER  
event  
ON table  
FOR EACH ROW  
commande sql

"event "peut être UPDATE, INSERT ou DELETE

# Module SQL

→ Syntaxe pour plusieurs lignes de commande :

◆ delimiter //

```
CREATE TRIGGER nom_trigger  
BEFORE or AFTER  
event  
ON table  
FOR EACH ROW  
BEGIN  
    commande sql ;  
    commande sql ;  
END ; //  
  
delimiter ;
```

# Module SQL

- Les mots clés NEW et OLD permettent d'accéder nouvelle et anciennes valeurs des colonnes de la ligne modifiée
- Sur un trigger "insert" seul NEW existe
- Sur un trigger "delete", seul OLD existe
- Exemple
  - NEW.titre
- Sur un trigger "before", on peut modifier la ligne en cours avec le mot clé SET
- Exemple
  - $\text{SET NEW.col2} = \text{NEW.col1} + 2$



# Module SQL

→ Dans un bloc BEGIN - END on peut utiliser des blocs de condition IF THEN END IF

# Module SQL

## Exercice 62

→ Créer un trigger sur la table `abonne` permettant de garder à jour la colonne `tranche_age`

# Module SQL

1. Structure d'une base
2. Installation de MySQL
3. Interrogation de données
4. Requêtes multi-tables
5. Requêtes complexes
6. Manipulation de données
7. Langage de définition de données
8. Développement de fonctions
9. Triggers
- 10. Transactions**

- Qu'est qu'une transaction
- Gestion du rollback
- Verrous

# 10. Transactions

# Module SQL

Objectif :

- Connaître le principe des transactions
- Savoir les gérer
- Connaître l'impact sur les verrous

# Module SQL

## Notion de transaction

- Une transaction permet de regrouper plusieurs ordre sql de mise à jour en un ensemble cohérent
- Toutes les mises à jour effectuées sont validées ensemble à la fin de la transaction (Commit)
- Si la transaction est annulée les actions déjà effectué sont annulées (Rollback)

# Module SQL

## Utilisation

→ Syntaxe :

◆ BEGIN ;

<sql>

<sql>

COMMIT ;    ou    ROLLBACK ;

# Module SQL

## Verrouillage

- Lorsqu'un enregistrement est mis à jour, il est verrouillé (Lock) jusqu'à ce que la transaction soit terminée
- Les transactions doivent être courtes



# Module SQL

## Exercice 63

- Effectuer plusieurs mises à jour de la table abonne après avoir démarré une transaction puis effectuer un rollback

# QCM



**Restons en contact.**

**DIGINAMIC**

Lionel Cabon, Directeur  
[contact@diginamic.fr](mailto:contact@diginamic.fr)



Nos coordonnées : 04 34 09 04 60  
[contact@iocean.fr](mailto:contact@iocean.fr) - [www.iocean.fr](http://www.iocean.fr)

N° Déclaration OF : 91 34 08867 34

Nantes, Paris, Montpellier

[www.diginamic.fr](http://www.diginamic.fr)