



# MySQL

DÉCOUVERTE DU MODÈLE RELATIONNEL



# SOMMAIRE

**Introduction**

**Outils**

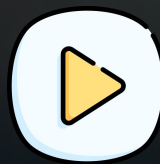
**Architecture de base de données**

**Lire des données**

**Écrire / Modifier / Supprimer des données**

**Altérer une base de données**

**Aller plus loin**



# INTRODUCTION

---



# QU'EST-CE QU'UNE DONNÉE ?

## DÉFINITION

Une donnée est une représentation conventionnelle d'une information en vue de son traitement informatique.

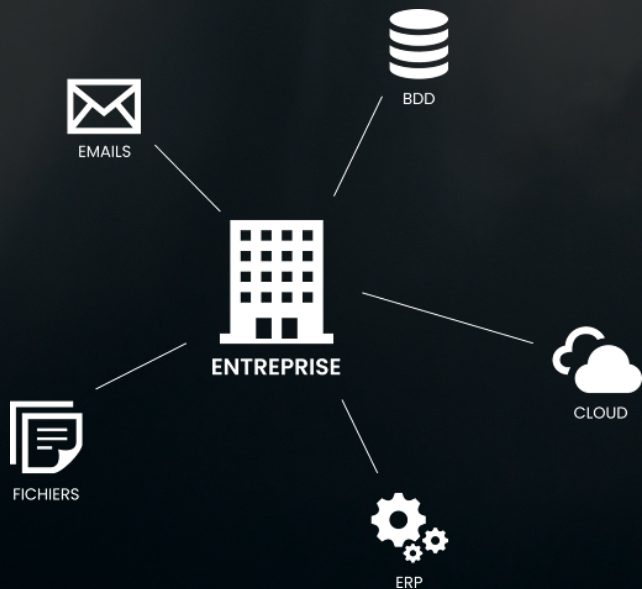
---

firstName	John
Clef	Valeur



# LES DONNÉES EN ENTREPRISE

## UN ÉLÉMENT CRUCIAL



Les données en entreprise augmentent exponentiellement, elles proviennent de sources diverses et sont formatées de manière hétérogène.

C'est pourquoi un S.I. Système d'information adapté doit être mis en place. Les bases de données relationnelles peuvent être une solution.

# LES ENJEUX

ILS SONT NOMBREUX

Aujourd'hui toutes les entreprises sont dotées d'un système d'information car ses avantages sont multiples.

Il est donc nécessaire d'avoir une bonne compréhension de celui-ci.



# UNE DONNÉE DE MAUVAISE QUALITÉ

CE QU'IL FAUT À TOUT PRIX ÉVITER

## # EXEMPLES

Données absentes	firstName : --
Données erronées	lastName : 29
Données avec une définition imprécise	fullName : John Doe / Doe John
Données obsolètes	--





# LES TYPES DE DONNÉES

## EN VERSION SIMPLIFIÉE

TYPES	EXEMPLES
Les chaînes de caractères	"Ceci est une chaîne de caractères"
Les nombres	314
Les tableaux	["john.d@free.fr", "jane.d@yahoo.fr"]
Les booléens	Vrai / Faux - 1/0
Les timestamps	1634232237

Timestamp : le nombre de secondes écoulées depuis le 1 janvier 1970.







# LES DONNÉES D'UN EMAIL

EN VERSION SIMPLIFIÉE

	CLEFS	VALEURS
	transmitter	"marc.doe@wanadoo.fr"
	recipients	["john.d@free.fr", "jane.d@yahoo.fr"]
	hiddenCopy	true
	object	"Ceci est l'objet de l'email"
	content	"Ceci est le contenu de l'email"
	created_at	16344232237

Timestamp : le nombre de secondes écoulées depuis le 1 janvier 1970.





# OUTILS

---

# LES OUTILS

## DE QUOI AVONS NOUS BESOIN ?



**Google Docs**

Google



**Google Sheets**

Google



**Draw.IO**

diagrams.net



**MySQL Workbench & Server**

MySQL

Ils existent bien d'autres logiciels permettant la création de base de données avec chacun leurs avantages et leurs inconvénients.

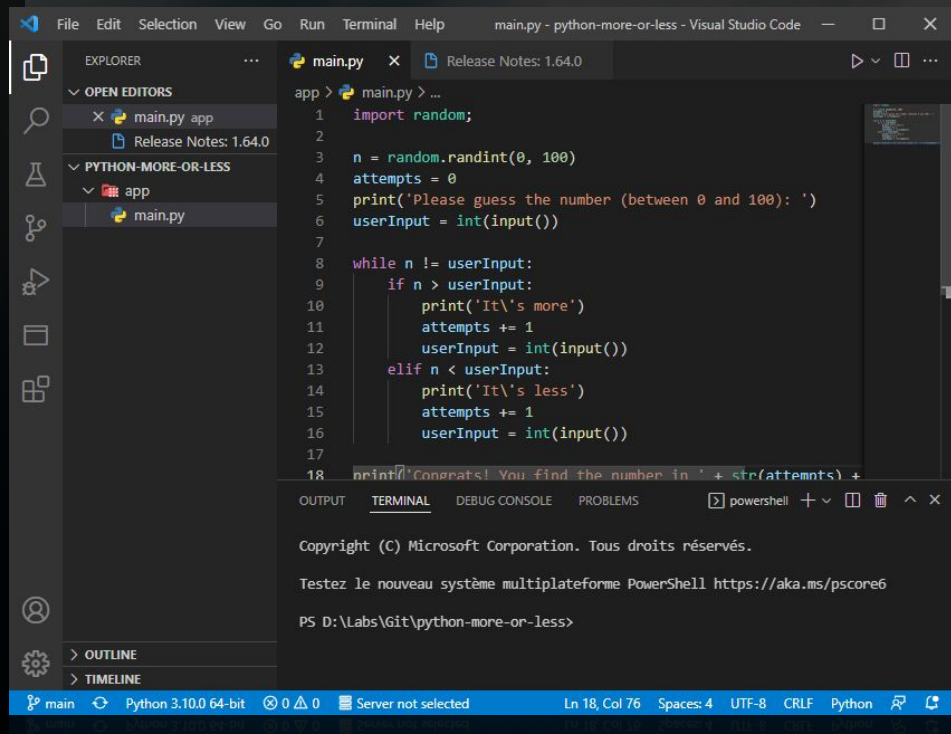
Je vous invite à réaliser une veille technologique afin de connaître les meilleurs outils.

---



# VISUAL STUDIO CODE

## INTERFACE & RACCOURCIS



**Gratuit**

**Système d'extensions**

**Connecté au système de versionning**

**Le plus populaire**

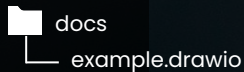
**Et bien plus...**




# DRAW.IO

APP. DE BUREAU OU EXTENSION VS. CODE

**mysql-course**



1. Créer et ouvrir le dossier mysql-course à l'aide de VS. Code.
2. Créer un dossier docs.
3. Installer l'extension draw.io. 
4. Créer le fichier example.drawio dans le dossier docs.



# MySQL INSTALLER

## WORKBENCH & SERVER

<b>Windows (x86, 32-bit), MSI Installer</b> (mysql-installer-web-community-8.0.29.0.msi)	8.0.29 MD5: 4f735569267527dec28d9e8d977f33d1   <a href="#">Signature</a>	2.3M	<a href="#">Download</a>
<b>Windows (x86, 32-bit), MSI Installer</b> (mysql-installer-community-8.0.29.0.msi)	8.0.29 MD5: 3f4def7aef5e2e030e2dd62e784f246   <a href="#">Signature</a>	439.6M	<a href="#">Download</a>

<https://dev.mysql.com/downloads/installer/>

1. Se rendre sur le lien ci-contre.

**mysql-installer-web-community-8.0.29.0.msi**

2. Choisir la version web-community

3. Installer MySQL Workbench et MySQL Server.







# ARCHITECTURE DE BASE DE DONNÉES

---

# SGBDR

## SYSTÈME DE GESTION DE BASES DE DONNÉES RELATIONNELLES

Un système de gestion de base de données relationnelles (SGBDR) est un logiciel standard qui repose sur les principes du modèle relationnel.

### LES TROIS PRINCIPALES FONCTIONNALITÉS D'UN SGBDR

1. La définition des données sous forme de relations.
2. La manipulation des données via un langage déclaratif.
3. L'administration des données.



# SGBDR

## LES TECHNOLOGIES

### ORACLE

Oracle Database est un système de gestion de base de données relationnelle (SGBDR) qui depuis l'introduction du support du modèle objet dans sa version 8 peut être aussi qualifié de système de gestion de base de données relationnel-objet (SGBDRO).



Microsoft SQL Server est un système de gestion de base de données en langage SQL incorporant entre autres un SGBDR développé et commercialisé par la société Microsoft.

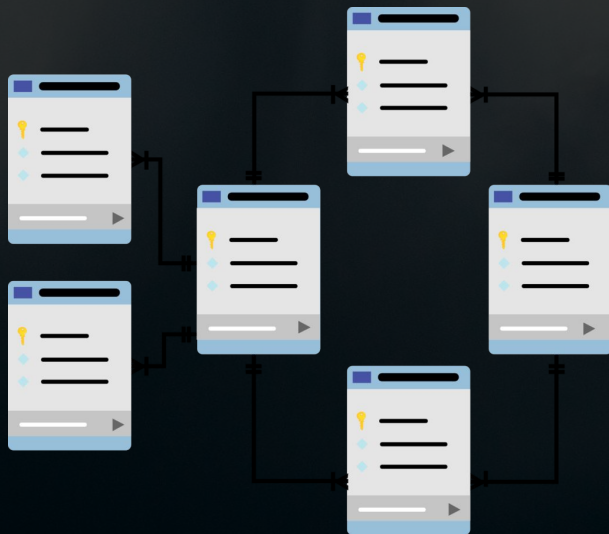


MySQL est un système de gestion de bases de données relationnelles. Il est distribué sous une double licence GPL et propriétaire.



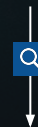
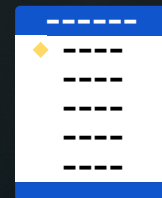
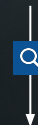
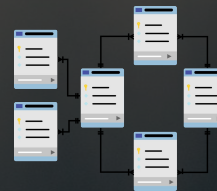
# SCHÉMA

## LA VUE D'ENSEMBLE



Un schéma de base de données représente la configuration logique de tout ou partie d'une base de données relationnelle.

Il peut se présenter à la fois sous la forme d'une représentation visuelle et d'un ensemble de formules, appelées contraintes d'intégrité, qui régissent une base de données.



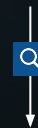
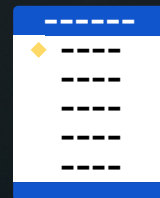
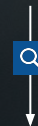
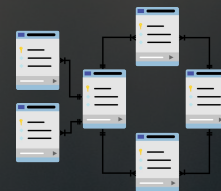

# ENTITÉ / TABLE

UN ÉLÉMENT SE RATTACHANT À LA RÉALITÉ

articles	
◆ id	
title	
content	
created_at	
updated_at	

Une entité / table est un objet du réel, concret ou abstrait dont on s'accorde à reconnaître une existence propre : doit présenter un intérêt pour la compréhension de la réalité.

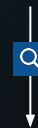
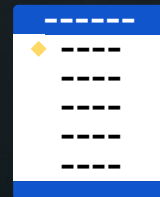
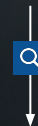
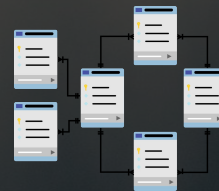
---




# COLONNES & ENREGISTREMENTS

EXEMPLE DE LA TABLE ARTICLES AVEC LA REPRÉSENTATION DES DONNÉES

	COLONNE 1	COLONNE 2	COLONNE 3	COLONNE 4	COLONNE 5
	id	title	content	created_at	updated_at
ENREGISTREMENT 1	1	Les voit...	[...]	16344232236	16344232236
ENREGISTREMENT 2	2	Sous le...	[...]	16344269842	16344259985
ENREGISTREMENT 3	3	Un jeun...	[...]	16389852358	16389852358








# LA MÉTHODE MERISE

## LES ÉTAPES À SUIVRE POUR CRÉER UNE BASE DE DONNÉES RELATIONNELLE

Merise est une méthode de conception de systèmes d'information, notamment des bases de données relationnelles.

Nous verrons à travers la réalisation de cinq étapes clefs, comment créer une base de données relationnelle propre.

---

1. Définir une convention de nommage.
2. Création du dictionnaire de données.
3. Création du modèle conceptuel de données – MCD.
4. Création du modèle logique de données – MLD.
5. Création du modèle physique de données – MPD.

# LES OUTILS

## DE QUOI AVONS NOUS BESOIN ?



### Google Docs

Convention de nommage



### Google Sheets

Dictionnaire de données



### Draw.IO

Modèle conceptuel de données



### MySQL Workbench

Modèle logique de données - Modèle physique de données

Ils existent bien d'autres logiciels permettant la création de base de données avec chacun leurs avantages et leurs inconvénients.

Je vous invite à réaliser une veille technologique afin de connaître les meilleurs outils.

---

# MODÈLE RELATIONNEL

## COMPRENDRE PAR L'EXEMPLE, LE CAS D'UNE AUTO-ÉCOLE

Une auto-école comprend des moniteurs et des véhicules. Elle prépare des élèves à passer un permis de type donné. Les moniteurs donnent des leçons aux élèves, une leçon possède une durée en heure.



### RÈGLES DE GESTION

1. Une leçon n'est donnée que par un seul moniteur.
2. Un élève ne passe qu'un seul type de permis.
3. Une leçon est donnée que pour un seul élève.
4. Un élève apprend à conduire sur un seul véhicule.

# CONVENTION DE NOMMAGE

## DATABASE STYLE GUIDE

La base de données est une partie indispensable à une application.

Définir une convention de nommage permet de respecter une certaine norme entre les noms des différents éléments d'une base de données et ainsi minimiser des erreurs potentielles.

### Database Style Guide

v1.0

#	Standards	Exemple
Schema	snake_case	my_blog
Entity / Table	snake_case / plural	users
Column	snake_case	created_at
Primary Key	id	id
Foreign Key	<plural_table_name>_id	users_id
Pivot Table	<plural_table_1>_<plural_table_2>	articles_tags





# DICTIONNAIRE DE DONNÉES

## LE DÉBUT DU PÉRIPLE

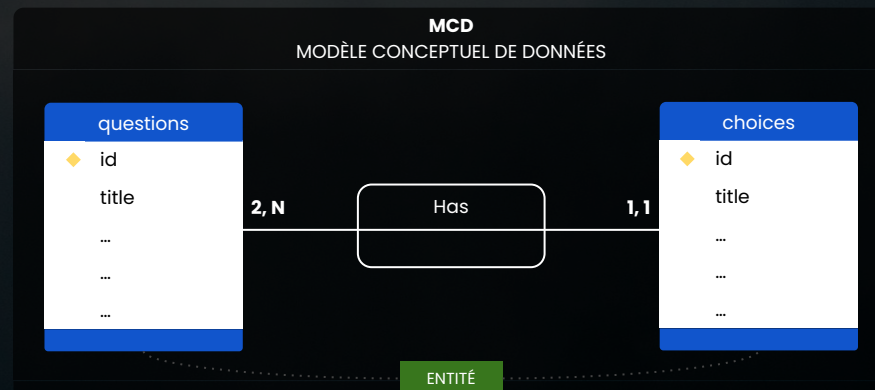
Le dictionnaire de données permet de recenser toutes les données nécessaires à la réalisation de l'application sans pour le moment se soucier du modèle relationnel.

	A	B	C	D	E
1	<b>students</b>				<b>lessons</b>
2	<b>columns</b>	<b>type</b>	<b>description</b>		<b>columns</b>
3	id	integer			id
4	first_name	text			duration
5	last_name	text			planned_for
6	email	text	unique		created_at
7	password	text	encrypted		updated_at
8	created_at	timestamp	data logging		deleted_at
9	updated_at	timestamp	data logging		
10	deleted_at	timestamp	soft deletion		
11					
12					
13	<b>teachers</b>				<b>lesson_statements</b>
14	<b>columns</b>	<b>type</b>	<b>description</b>		<b>columns</b>
15	id	integer			id
16	first_name	text			title
17	last_name	text			created_at
18	email	text	unique		updated_at
19	password	text	encrypted		deleted_at
20	created_at	timestamp	data logging		
21	updated_at	timestamp	data logging		
22	deleted_at	timestamp	soft deletion		
23					
24					
25	<b>driving_licenses</b>				
26	<b>columns</b>	<b>type</b>	<b>description</b>		
27					

# MODÈLE CONCEPTUEL DE DONNÉES

COMMENÇONS PAR UN CAS SIMPLE

Avant de réfléchir au schéma relationnel d'une application, il est bon de modéliser la problématique à traiter d'un point de vue conceptuel et indépendamment du logiciel utilisé.



dl\_types = driving\_license\_types, pour des raisons de lisibilité, les colonnes ne sont pas représentées.



# TABLE DE RÉFÉRENCE

## DONNÉES QUASI STATIQUES

Une table de référence, c'est par exemple la table des civilités, celles des villes et de leur code postal, une table de noms de mois, de jours de semaine... Ce sont des listes de données quasi statiques dont on identifie les valeurs par des clefs.

Ci-dessus un lien pointant vers un exemple de table de référence de marques de voiture.

---

id	title
1	Lundi
2	Mardi
3	Mercredi
4	Jeudi
5	Vendredi
6	Samedi
7	Dimanche

TABLE : DAYS\_REF





# CARDINALITÉS

## AU COEUR DU MODÈLE RELATIONNEL

NOTATIONS	DESCRIPTIONS
-----------	--------------

0, 1	Aucune ou une instance
1	Strictement une instance
0, N	Aucune instance ou plusieurs
1, N	Une instance ou plusieurs

# LA CLEF PRIMAIRE

IDENTIFIANT UNIQUE ET AUTO-INCRÉMENTÉE

articles	
◆ id	
title	
content	
created_at	
updated_at	

	COLONNE 1
	id
ENREGISTREMENT 1	1
ENREGISTREMENT 2	2
ENREGISTREMENT 3	3

AUTO-INCRÉMENT

Chacune des tables de la base de données à nécessairement une colonne id.

Elle permet d'identifier un enregistrement, sa valeur est unique et associée à un auto-increment.

Par défaut, la valeur de départ de l'auto-increment est 1, et est incrémenté de 1 pour chaque nouvel enregistrement.

# CLEF UNIQUE

## LA CONTRAINTE D'UNICITÉ

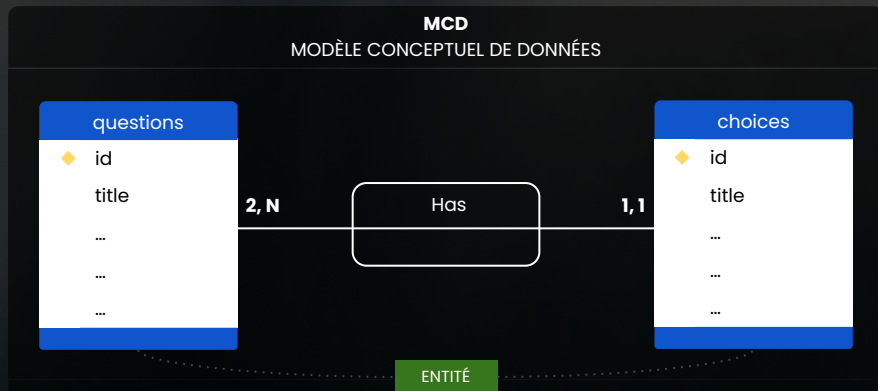
users	
◆	id
	username
u	email
	password
	...

Afin d'éviter une même valeur pour deux enregistrements sur la même colonne, MySQL offre la possibilité d'ajouter une contrainte d'unicité.

Pour ne pas avoir deux utilisateurs ayant exactement la même adresse email, nous pouvons spécifier à l'aide de MySQL, l'ajout d'une clef unique sur la colonne email.

# CLEF ÉTRANGÈRE

C'EST LÀ QUE LE MODÈLE RELATIONNEL PREND TOUT SON SENS



Les clefs étrangères apparaissent dans le modèle logique de données en fonction des cardinalités de la relation.

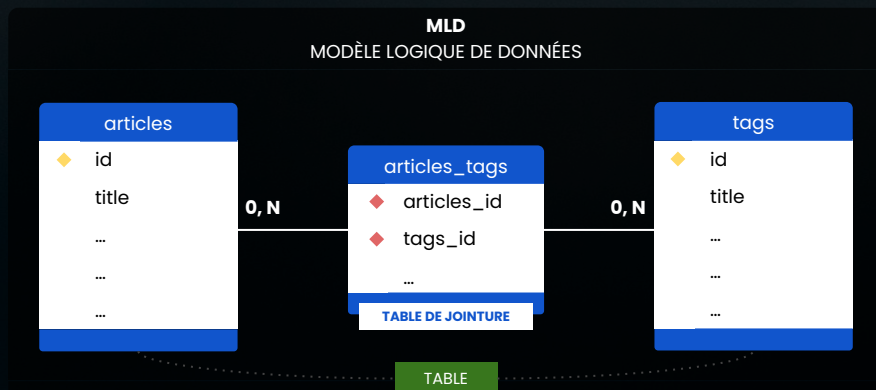
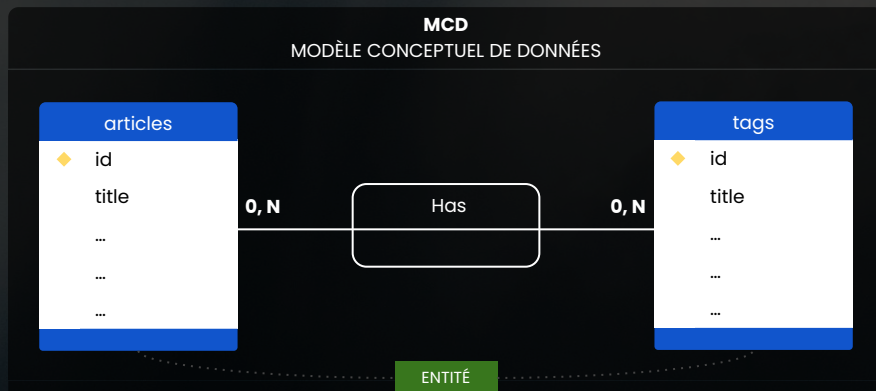
Grâce à la clef étrangère `questions_id`, nous pourrons à partir de la table `choices` retrouver la question à laquelle appartient le choix.

---



# CLEF PRIMAIRE COMPOSÉE

## COMPRENDRE LA TABLE DE JOINTURE / TABLE PIVOT

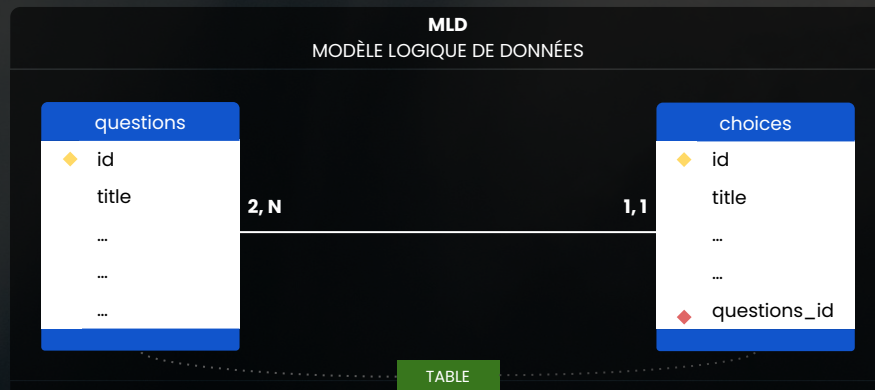


Lors d'une relation **n, m** entre deux entités d'un modèle conceptuel de données, nous aurons dans le modèle logique de données l'apparition d'une **table de jointure** ayant une **clef primaire composée**.

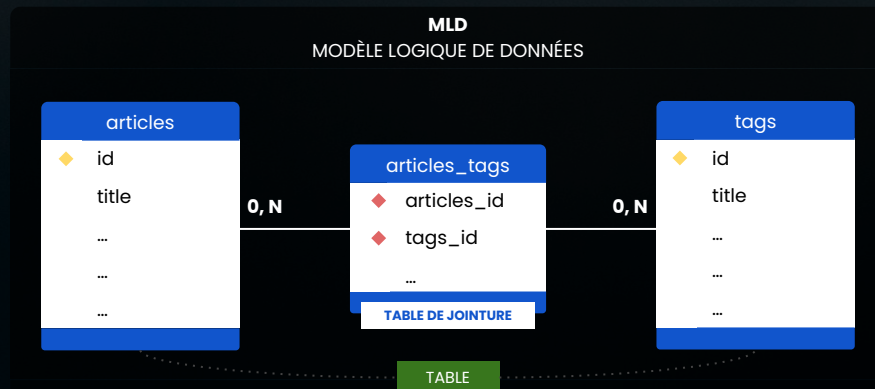
C'est via cette table que nous saurons quels sont les tags associés à un article et quels sont les articles associés à un tag.

# MODÈLE LOGIQUE DE DONNÉES

MLD



Le modèle logique de données est caractérisé par la disparition des associations et l'apparition des clefs étrangères ainsi que des tables de jointure en fonction des cardinalités.



# HISTORISATION DES DONNÉES

## DATA LOGGING, LES TIMESTAMPS

### articles

- ◆ id
- title
- content
- created\_at**
- updated\_at**

### CREATED\_AT

La colonne "created\_at" permet de stocker le moment de l'ajout d'un enregistrement. Pour la table articles, cette colonne nous permettra de savoir quand a été ajouté un article.

La colonne "created\_at" prend en valeur par défaut le moment de l'ajout défini soit par la constante `CURRENT_TIMESTAMP`, soit par la fonction `NOW()`.

### UPDATED\_AT

La colonne "updated\_at" permet de stocker le moment de la dernière modification d'un enregistrement. Pour la table articles, cette colonne nous permettra de savoir quand a eu lieu la dernière modification de l'article.

La colonne "updated\_at" prend en valeur par défaut le moment de l'ajout défini soit par la constante `CURRENT_TIMESTAMP`, soit par la fonction `NOW()`.



# LA SUPPRESSION DOUCE

## SOFT DELETION

### articles

- ◆ id
- title
- content
- created\_at
- updated\_at
- deleted\_at**

La colonne "deleted\_at" permet la gestion de la suppression douce.

La suppression douce est une bonne pratique qui permet de supprimer les données affichées côté applicatif tout en les conservant en base de données.

La colonne "deleted\_at" est de type `TIMESTAMP` et prend la valeur `NULL` par défaut. Pour appliquer la suppression douce, on changera la valeur `NULL` en timestamp correspondant au moment de la suppression.

---

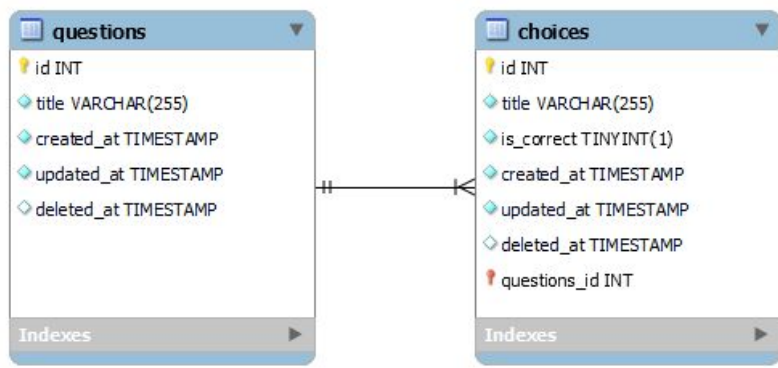
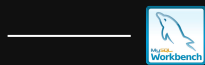


# DÉCOUVERTE DE MySQL WORKBENCH

SGBDR MySQL

À l'aide de MySQL Workbench, nous allons pouvoir créer notre modèle logique de données [MLD] à partir du MCD.

Ce logiciel est gratuit et permet beaucoup de choses pour la conception et la manipulation d'une base de données.



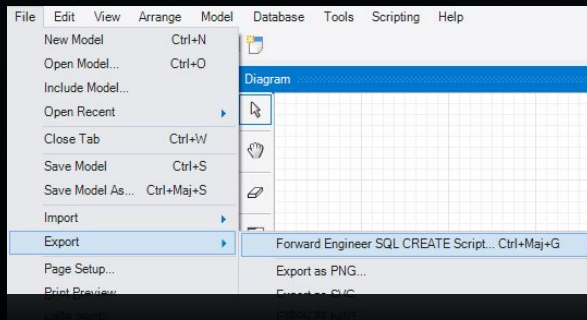
# LE MODÈLE PHYSIQUE DE DONNÉES

## GÉNÉRATION DU MPD À L'AIDE DE MySQL WORKBENCH

Le modèle physique de données n'est ni plus ni moins que le code sql permettant la création de la base de données.

MySQL Workbench s'occupera de générer pour nous le MPD [script SQL]. Pour ce faire, il faudra se rendre sur :

1. File
2. Export
3. Forward Engineer SQL Create script...



```
-- MySQL Script generated by MySQL Workbench
-- Sat Oct 15 19:27:21 2022
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering
```

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
```

```
-- -----
-- Schema quiz
-- -----
-- A database for quiz creation.
```

```
-- -----
-- Schema quiz
-- -----
-- A database for quiz creation.
```

```
CREATE SCHEMA IF NOT EXISTS `quiz` DEFAULT CHARACTER SET utf8 COLLATE utf8_bin ;
USE `quiz` ;
```

```
-- -----
-- Table `quiz`.`questions`
-- -----
DROP TABLE IF EXISTS `quiz`.`questions` ;
```

```
CREATE TABLE IF NOT EXISTS `quiz`.`questions` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(255) NOT NULL,
  `created_at` TIMESTAMP NOT NULL DEFAULT NOW(),
  `updated_at` TIMESTAMP NOT NULL DEFAULT NOW(),
  `deleted_at` TIMESTAMP NULL DEFAULT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

```
...
```





# LES MOTS RÉSERVÉS

NE PAS UTILISER CES MOTS AFIN D'ÉVITER LES CONFLITS AVEC LE LANGAGE SQL

## EXEMPLES DE MOTS RÉSERVÉS

<b>NAME</b>	<b>MASTER</b>
<b>AFTER</b>	<b>NEXT</b>
<b>SELECT</b>	<b>ALTER</b>
<b>ANALYSE</b>	<b>CONNECTION</b>
<b>NONE</b>	<b>...</b>

Le langage SQL utilise de nombreux mots dans sa syntaxe, ils sont dits réservés, afin d'éviter les conflits. Il vaut mieux éviter de les utiliser lors de la création d'une structure de données.



# LES TYPES DE DONNÉES

ILS SONT NOMBREUX

## LES CHAÎNES DE C.

**CHAR**

**VARCHAR**

**TEXT**

**LONGTEXT**

## LES NOMBRES

**TINYINT**

**INT**

**BIGINT**

**FLOAT**

**DOUBLE**

## LES DATES

**DATE**

**TIME**

**DATETIME**

**TIMESTAMP**

# COMMANDE SHOW

## INFORMATION SUR LA STRUCTURE

-- Affiche toutes les base de données

```
SHOW DATABASES;
```

-- Affiche des infos. sur les tables

```
SHOW TABLE STATUS;
```

-- Affiche des infos. sur les procédure

```
SHOW PROCEDURE STATUS;
```

-- Affiche le nombre d'erreurs d'une base de données

```
SHOW COUNT(*) ERRORS;
```

-- Affiche le nombre d'avertissements d'une base de données

```
SHOW COUNT(*) WARNINGS;
```

show.sql

La commande SHOW permet d'afficher des métadonnées d'une base de données.

# SQL : CHEAT SHEET

ACRONYMES & ÉLÉMENTS DE VOCABULAIRE

## #

- DB - DATABASE
- RDBMS - RELATIONAL DATABASE MANAGEMENT SYSTEM
- SQL - STRUCTURED QUERY LANGUAGE
- NoSQL - NOT ONLY SQL
- CRUD - CREATE / READ / UPDATE / DELETE



- BDD - BASE DE DONNÉES
- SGBDR - SYSTÈME DE GESTION DE BASE DE DONNÉES RELATIONNELLE

## CONCEPTION

- UML - UNIFIED MODELING LANGUAGE
- MCD - MODÈLE CONCEPTUEL DE DONNÉES
- MLD - MODÈLE LOGIQUE DE DONNÉES
- MPD - MODÈLE PHYSIQUE DE DONNÉES

## LANGAGE SQL

- DDL - DATA DEFINITION LANGUAGE - CREATE / DROP / ALTER / TRUNCATE
- DQL - DATA QUERY LANGUAGE - SELECT
- DML - DATA MANIPULATION LANGUAGE - INSERT / UPDATE / DELETE
- DCL - DATA CONTROL LANGUAGE - GRANT / REVOKE
- TCL - TRANSACTION CONTROL LANGUAGE - COMMIT / ROLLBACK





# TRAVAUX PRATIQUES

---



# QUIZ

Appliquer la méthode Merise pour créer une base de données permettant de stocker les données d'un questionnaire.

Créer un dossier "quiz" où vous stockerez tous les documents de la base de données "quiz".

---

1. Définir une convention de nommage
2. Créer le dictionnaire de données
3. Créer le MCD
4. Créer le MLD
5. Créer le MPD

Appliquer la méthode Merise pour créer une base de données permettant de stocker les données d'une auto-école.

Créer un dossier "driving\_school" où vous stockerez tous les documents de la base de données "driving\_school".

---

1. Définir une convention de nommage

2. Créer le dictionnaire de données

3. Créer le MCD

4. Créer le MLD

5. Créer le MPD



# AUTO-ÉCOLE

**MAIS POURQUOI AVOIR DEUX  
TABLES POUR STUDENTS ET  
TEACHERS ?**

**CES DEUX ENTITÉS ONT LES MÊMES  
INFORMATIONS...**





# BLOG

Appliquer la méthode Merise pour créer une base de données permettant de stocker les données d'un blog, des spécifications techniques vous seront fournies.

Créer un dossier "blog" où vous stockerez tous les documents de la base de données "blog".

1. Définir une convention de nommage
2. Créer le dictionnaire de données
3. Créer le MCD
4. Créer le MLD
5. Créer le MPD



## **BONUS : FIRESHIP SQL**



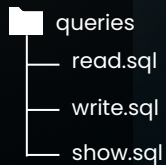
# LIRE DES DONNÉES

---

# CRÉATION DU PROJET

OÙ ALLONS-NOUS ÉCRIRE ET CONSERVER NOS REQUÊTES ?

**mysql-course**



1. Créer et ouvrir le dossier mysql-course à l'aide de VS. Code.
2. Créer un dossier queries.
3. À l'intérieur du dossier queries, créer deux fichiers, read.sql et write.sql.



# CRÉATION DU JEU DE DONNÉES

## CRÉER DES ENREGISTREMENTS

Il est important de ne pas confondre structure de données et jeu de données.

Une fois la structure définie, le script sql ci-contre ajoutera des données. Nous avons trois questions avec trois choix par question.

```
-- QUESTIONS: COUNT 3
INSERT INTO questions (`title`) VALUES ('Combien mesure la Tour Eiffel en mètres ?');
INSERT INTO questions (`title`) VALUES ('Quelle est la capitale de la Hongrie ?');
INSERT INTO questions (`title`) VALUES ('Qui a animé le Bigdil ?');

-- CHOICES: COUNT 9 (3 PER QUESTION)
INSERT INTO choices (`title`, `is_correct`, `questions_id`) VALUES ('200', 0, 1);
INSERT INTO choices (`title`, `is_correct`, `questions_id`) VALUES ('300', 1, 1);
INSERT INTO choices (`title`, `is_correct`, `questions_id`) VALUES ('400', 0, 1);

INSERT INTO choices (`title`, `is_correct`, `questions_id`) VALUES ('Helsinki', 0, 2);
INSERT INTO choices (`title`, `is_correct`, `questions_id`) VALUES ('Ljubljana', 0, 2);
INSERT INTO choices (`title`, `is_correct`, `questions_id`) VALUES ('Budapest', 1, 2);

INSERT INTO choices (`title`, `is_correct`, `questions_id`) VALUES ('Vincent Lagaf', 1, 3);
INSERT INTO choices (`title`, `is_correct`, `questions_id`) VALUES ('Cyril Hanouna', 0, 3);
INSERT INTO choices (`title`, `is_correct`, `questions_id`) VALUES ('Thierry Ardisson', 0, 3);
```

# COMMANDE SELECT

## SÉLECTIONNER DES DONNÉES

La commande "SELECT" en SQL permet de lire les données d'une base de données relationnelle.

Afin de mieux comprendre sa syntaxe, n'hésitez pas à tester les requêtes ci-contre.

---

-- VOTRE COMMENTAIRE

-- Requête toutes les questions (\* équivaut à 'All')

```
SELECT * FROM questions;
```

-- La clause WHERE (filtre)

-- Requête tous les choix de la question dont l'id est égale à 1

```
SELECT * FROM choices WHERE questions_id = 1;
```

-- La clause WHERE avec LIKE

-- Requête toutes les questions dont le titre comprend le mot "eiffel"

```
SELECT * FROM questions WHERE title LIKE '%Eiffel%';
```

-- DISTINCT

-- Requête toutes les question dont le titre est différent

```
SELECT DISTINCT title FROM questions;
```

-- ORDER BY ASC/DESC - Ordonnancement numérique ou alphabétique

-- Requête tous les choix de question ordonnés alphabétiquement

```
SELECT * FROM choices ORDER BY title ASC;    -- Croissant
```

```
SELECT * FROM choices ORDER BY title DESC;    -- Décroissant
```

# CLAUSE WHERE

FILTRE UNE SÉLECTION

OPÉRATEURS DE COMPARAISON	TESTE SI...
$A = B$	A est égal à B
$A \neq B$	A est différent de B
$A > B / A < B$	A est sup. à B / A est inf. à B
$A \geq B / A \leq B$	A est sup. ou égal à B / A est inf. ou égal à B
A BETWEEN B AND C	A est compris entre B et C
A LIKE "string"	La chaîne de caractères est contenu dans A selon un modèle spécifié
A IN (B1, B2, B3)	A est présent dans la liste (B1, B2, B3, ...)
A IS NULL / A = NULL	A n'a pas de valeur

# OPÉRATEUR LOGIQUE

## ALGÈBRE DE BOOLE

```
-- Opérateur logique : AND
```

```
SELECT * FROM choices
      WHERE title LIKE '%eiffel'
      AND id = 1;
```

```
-- Opérateur logique : OR
```

```
SELECT * FROM choices
      WHERE title LIKE '%eiffel'
      OR id = 1;
```

```
-- Opérateur logique : NOT
```

```
SELECT * FROM questions WHERE deleted_at IS NOT NULL;
```

Ci-contre des exemples d'utilisation d'opérateurs logiques qui permettent d'affiner des expressions conditionnelles.

---

read.sql



# JOINTURES

## UTILISER LA TRANSITIVITÉ POUR REQUÊTER PLUSIEURS TABLES

Grâce aux jointures (INNER JOIN), nous pouvons requêter plusieurs tables en s'aidant de la clef étrangère permettant la transitivité de l'id. L'exemple ci-dessous récupère les questions et les choix qui leurs sont associés.

```
-- INNER JOIN
```

```
SELECT * FROM questions INNER JOIN choices ON choices.questions_id = questions.id; ✓
```

```
SELECT * FROM questions, choices WHERE choices.questions_id = questions.id;
```

read.sql

	id	title	crea	upd	del	id	title	is_correct	cre	up	del	questions_id
▶	1	Combien mesure la Tour Eiffel en mètres ?	2...	2...	NULL	1	200	0	2...	2...	NULL	1
	1	Combien mesure la Tour Eiffel en mètres ?	2...	2...	NULL	2	300	1	2...	2...	NULL	1
	1	Combien mesure la Tour Eiffel en mètres ?	2...	2...	NULL	3	400	0	2...	2...	NULL	1
	2	Quelle est la capitale de la Hongrie ?	2...	2...	NULL	4	Helsinki	0	2...	2...	NULL	2
	2	Quelle est la capitale de la Hongrie ?	2...	2...	NULL	5	Ljubljana	0	2...	2...	NULL	2
	2	Quelle est la capitale de la Hongrie ?	2...	2...	NULL	6	Budapest	1	2...	2...	NULL	2
	3	Qui a animé le Bigdil ?	2...	2...	NULL	7	Vincent Lagaf	1	2...	2...	NULL	3
	3	Qui a animé le Bigdil ?	2...	2...	NULL	8	Cyril Hanouna	0	2...	2...	NULL	3
	3	Qui a animé le Bigdil ?	2...	2...	NULL	9	Thierry Ardisson	0	2...	2...	NULL	3



# FONCTIONS NATIVES

## EXEMPLES

NOMS	DESCRIPTIONS
NOW( )	Fonction qui retourne le timestamp actuel.
AVG( )	Fonction d'agrégation calculant la moyenne d'une colonne.
CONCAT(str_1, str_2, str_3, ...)	Fonction qui concatène les chaînes de caractères passées en paramètres.
LENGTH(str)	Fonction qui retourne la taille d'une chaîne de caractères.
UPPER(str)	Fonction qui retourne une chaîne de caractères en majuscules.
LOWER(str)	Fonction qui retourne une chaîne de caractères en minuscules.



# FONCTIONS NATIVES

## EXEMPLES D'UTILISATION

```
-- Exemples d'utilisation de fonctions
-- Retourne le nombre d'enregistrements d'une table
SELECT COUNT(*) FROM choices;

-- Retourne l'enregistrement avec le plus grand id
SELECT * FROM choices
WHERE id = (SELECT MAX(id) FROM choices);

-- Retourne l'enregistrement avec le plus petit id
SELECT * FROM choices
WHERE id = (SELECT MIN(id) FROM choices);
```

read.sql

Ci-contre, quelques exemples d'utilisation de fonctions, nous verrons plus tard que nous pourrons créer nos propres fonctions.

# GROUP BY

## AGRÉGATION DES DONNÉES

```
-- Retourne le nombre de choix par question
SELECT questions.title, COUNT(choices.id) FROM questions
INNER JOIN choices ON choices.questions_id = questions.id
GROUP BY questions.title;
```

read.sql

	title	COUNT(choices.id)
▶	Combien mesure la Tour Eiffel en mètres ?	3
	Quelle est la capitale de la Hongrie ?	3
	Qui a animé le Bigdil ?	3

La commande "GROUP BY" permet d'agréger des données, nous pouvons par exemple savoir combien de choix comporte chacune des questions.



# HAVING

## CLAUSE POST-AGRÉGATION

```
/**
  Retourne le nombre de choix par question
  pour les questions ayant 3 choix ou plus
  */
SELECT questions.title, COUNT(choices.id)
FROM questions
INNER JOIN choices ON choices.questions_id = questions.id
GROUP BY questions.title
HAVING COUNT(choices.id) >= 3;
```

read.sql

	title	COUNT(choices.id)
▶	Combien mesure la Tour Eiffel en mètres ?	3
	Quelle est la capitale de la Hongrie ?	3
	Qui a animé le Bigdil ?	3

La commande "HAVING" permet de filtrer le set de résultats après que l'agrégation soit effectuée.

Nous pouvons par exemple savoir le nombre de choix que comportent chacune des questions, et ne retourner que les questions qui n'ont que 3 choix ou plus.

# SQL : COMMANDE SELECT

<https://dev.mysql.com/doc/refman/8.0/en/select.html>

**SELECT**

**FROM**

**WHERE**

**GROUP BY**

**HAVING**

**ORDER BY      [ASC / DESC]**

**LIMIT**



# TRAVAUX PRATIQUES

---



# LIRE DES DONNÉES

QUIZ

Dans le dossier "quiz", créer un dossier "queries" avec à l'intérieur le fichier "read.sql". Dans ce fichier, écrire les requêtes suivantes.

---

1. Écrire une requête récupérant toutes les bonnes réponses.
2. Écrire une requête récupérant les réponses de la question 3.
3. Écrire une requête récupérant les questions commençant par "Q".
4. Écrire une requête récupérant le titre de chacune des questions.
5. Écrire une requête donnant la moyenne des réponses de la question 1.



Dans le dossier "blog", créer un dossier "queries" avec à l'intérieur le fichier "read.sql". Dans ce fichier, écrire les requêtes énoncées dans le PDF qui vous sera fourni.

---



# LIRE DES DONNÉES

BLOG



# ÉCRIRE DES DONNÉES

---

# INSERT INTO... VALUES...

AJOUTER UN OU PLUSIEURS ENREGISTREMENTS

```
-- INSERT INTO... VALUES...  
-- Ajout d'une nouvelle question  
INSERT INTO questions (`title`)  
VALUES ('De quelle couleur est le logo de CSS3 ?');  
  
-- Ajout de plusieurs questions  
INSERT INTO questions (`title`)  
VALUES  
    ('Que signifie NoSQL ?'),  
    ('Qu'est-ce qu'un logiciel libre ?');
```

write.sql

L'insertion de données dans une table s'effectue à l'aide de la commande INSERT INTO. Cette commande permet au choix d'ajouter une seule ligne à la base existante ou plusieurs lignes d'un coup.

—

# UPDATE... SET...

## MODIFIER DES ENREGISTREMENTS

```
-- UPDATE... SET... : Mettre à jour / Modifier des données
```

```
-- NOW() retourne le timestamp actuel
```

```
UPDATE choices
```

```
    SET title = '450',
```

```
        updated_at = NOW()
```

```
WHERE id = 3;
```

write.sql

La commande UPDATE permet d'effectuer des modifications sur des lignes existantes. Cette commande est souvent utilisée avec la clause "WHERE" pour spécifier sur quels enregistrements doivent porter les modifications.



Attention s'il n'y a pas de condition WHERE alors tous les enregistrements seront modifiés.



# DELETE

## SUPPRIMER DES ENREGISTREMENTS

```
-- DELETE : Suppression définitive
DELETE FROM questions WHERE id = 4;

-- Suppression douce
-- L'affichage sera géré du côté applicatif
UPDATE questions SET deleted_at = NOW() WHERE id = 4;
```

write.sql

La commande "DELETE" permet de supprimer les enregistrements d'une table. Si elle est suivie de la clause "WHERE" permet de sélectionner les enregistrements qui seront supprimés.



Attention s'il n'y a pas de condition WHERE alors tous les enregistrements seront supprimés.

# VIDER UNE TABLE

## SUPPRIMER TOUS LES ENREGISTREMENTS D'UNE TABLE

```
-- DELETE
-- Vider une table sans réinitialisation de l'auto-increment
DELETE FROM questions;

-- TRUNCATE TABLE
-- Vider une table avec réinitialisation l'auto-increment
TRUNCATE TABLE questions;
```

write.sql

Il y a deux manières de vider une table en SQL, avec la commande "DELETE" ou avec la commande "TRUNCATE".

Ce qu'il faut retenir, c'est que la commande "TRUNCATE" est plus performante et réinitialise l'auto-incrément.



# TRAVAUX PRATIQUES

---



# ÉCRIRE DES DONNÉES

QUIZ

Dans le dossier "quiz", créer un dossier "queries" avec à l'intérieur le fichier "write.sql". Dans ce fichier, écrire les requêtes suivantes.

---

1. Écrire une requête ajoutant la question suivante :  
"Quelles sont les couleurs du drapeau d'Espagne ?"
2. Écrire une requête ajoutant les choix suivants à la question précédente :  
"Blanc & Bleu", "Vert & Rouge", "Rouge & Jaune"
3. Écrire une requête supprimant de manière douce la première question et ses choix respectifs.



Dans le dossier "blog", créer un dossier "queries" avec à l'intérieur le fichier "write.sql". Dans ce fichier, écrire les requêtes énoncées dans le PDF qui vous sera fourni.

---



# ÉCRIRE DES DONNÉES

BLOG



# **ALTÉRER UNE BASE DE DONNÉES**

---

# QUE SIGNIFIE ALTÉRER ?

MODIFIER LA STRUCTURE DE LA BASE DE DONNÉES

Il faut bien faire la différence entre la structure de données et les données elles-mêmes.

---

## DONNÉES

DQL - DATA QUERY LANGUAGE - SELECT  
DML - DATA MANIPULATION LANGUAGE - INSERT / UPDATE / DELETE

Lorsqu'on parle des données, on fait référence aux enregistrements. Dans les travaux pratiques précédents, nous les avons créés à l'aide du fichier "dataset.sql". [Jeu de données]

## STRUCTURE DE DONNÉES

DDL - DATA DEFINITION LANGUAGE - CREATE / DROP / ALTER / TRUNCATE

Lorsqu'on parle de la structure de données, on fait référence aux tables et aux colonnes. C'est ce qui est créé par le MPD.

# CRÉER UNE TABLE

## CREATE TABLE

```
-- Table `quiz`.`questions`  
-- -----
```

```
CREATE TABLE IF NOT EXISTS `quiz`.`questions` (  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `title` VARCHAR(255) NOT NULL,  
  `created_at` TIMESTAMP NOT NULL DEFAULT NOW(),  
  `updated_at` TIMESTAMP NOT NULL DEFAULT NOW(),  
  `deleted_at` TIMESTAMP NULL,  
  PRIMARY KEY (`id`))  
ENGINE = InnoDB;
```

alter.sql

L'analyse du code "MPD" généré par l'assistant de MySQL Workbench permet de comprendre comment fonctionne la requête "CREATE TABLE".

Il est nécessaire de définir toutes les colonnes, leurs spécificités ainsi que leurs types comme dans l'exemple ci-contre.



# SUPPRIMER UNE TABLE

## DROP TABLE

```
-- -----  
-- Table `quiz`.`questions`  
-- -----
```

```
DROP TABLE `quiz`.`questions`;
```

alter.sql

La commande "DROP TABLE" permet de supprimer définitivement une table d'une base de données. Cela supprime en même temps les éventuels indices, triggers, contraintes et permissions associées à cette table.



Attention, il faut utiliser cette commande avec prudence car une fois la table supprimée, les données sont définitivement perdues.

# MODIFIER UNE TABLE

## ALTER TABLE

```
-- Ajouter la colonne points à la table questions après la colonne title
```

```
ALTER TABLE questions  
ADD points INT AFTER `title`;
```

```
-- Modifier le type de la colonne points
```

```
ALTER TABLE questions  
MODIFY points BIGINT;
```

```
-- Renommer la colonne points en score
```

```
ALTER TABLE questions  
CHANGE `points`, `score` INT;
```

```
-- Supprimer la colonne points
```

```
ALTER TABLE questions  
DROP `points`;
```

La commande ALTER TABLE permet de modifier une table existante. Elle permet d'ajouter une colonne, supprimer une colonne ou modifier une colonne existante, par exemple pour changer le type.

alter.sql



**ALLER PLUS LOIN**

---

# ALLER PLUS LOIN

## CONTINUER D'APPRENDRE



Le format de données XML



NoSQL avec MongoDB



Le langage JavaScript



La bibliothèque TinyDB en Python





**MERCI DE VOTRE ATTENTION**