1. Introduction

Objectifs du cours

L'objectif principal est de présenter les concepts fondamentaux des bases de données relationnelles. Pour ce faire, nous utiliserons une approche **éminemment pratique** .

A la fin de ce cours l'étudiant doit être capable de:

- **Comprendre les fondements théoriques** le plus basiques des bases de données relationnelles.
- **Manipuler les données** à l'insu du langage **SQL .**
- Interagir avec une base de données dans une **architecture client-serveur** et en utilisant le langage de programmation **PHP.**
- **Concevoir** une base de donnés.

1.1 Un monde plein de données

Nous enregistrons des données sur les formats et les supports les plus variés. Pour que vous puissiez acheter dans un supermarché, regarder une vidéo ou sauvegarder un très beau selfie dans votre portable, il faut que l'information soit stockée pour pouvoir la rendre accessible plus tard, quand nous avons besoin d'elle.

Le stockage des données sur des dispositifs numériques nous permet de maintenir l'information disponible à tout moment et nous accorde une capacité d'analyse qui n'est pas négligeable. Cette capacité d'analyse peut, par exemple, aider à simplifier de façon considérable la gestion de notre entreprise ou nous renseigner sûr les goûts et les tendances d'achat de nos clients.

Une base de données nous permet d'enregistrer des faits au sein d'un organisme tel qu'une banque, une entreprise, une administration, une université, un hôpital, etc. Une fois que nous avons accès aux données enregistrées nous pouvons les combiner pour en tirer des conclusions et prendre ensuite des décisions.

Aujourd'hui, "les grands géants du web" utilisent des techniques d'analyse très sophistiquées qui, entre autres, servent à créer des modèles prédictifs et anticiper nos actions. Par exemple, les suggestions de recherche que propose Google ou les propositions de Netflix quand nous ne savons pas quel film ou quelle série regarder.

1.2 Pourquoi utiliser une base de données?

Imaginons que nous avons une entreprise et que nous enregistrons l'information, par exemple, sur des feuilles Excel. A priori cela nous paraît largement suffisant pour gérer notre entreprise. Mais à mesure que notre entreprise grandit, il s'avère très difficile de maintenir à jour les données tout en les partageant et en les faisant évoluer. De plus, on constate très rapidement que pour en tirer des conclusions pertinentes à partir des données, il faut une maitrise importante du logiciel ce qui n'est pas toujours le cas pour l'ensemble de nos employés.

C'est ainsi qu'une base de donnés va nous aider à surmonter ces problèmes.

Avantages de l'utilisation d'une base de données

La manipulation et la gestion des bases de données constituent des opérations complexes.
Les **SGBD (Systèmes de Gestion de Bases de Données)** sont des logiciels qui permettent de gérer les Bases de Données.
Il prend en charge ces deux fonctionnalités:

- Accès aux fichiers et mises a jour.
- Interactions avec les applications et utilisateurs.

L'utilisation des SGBD permet de surmonter un tas de problèmes complexes. Voici une liste, non exhaustive:

- Problème **d'Intégrité**
- Problème de **Confidentialité**
- Problème de **Sécurité**
- Problème de **gestion d'opérations concurrentes**

1.3 Qu'est ce qu' une base de données?

Une base de données est un ensemble d'informations structurées et **mémorisées sur un support *persistant***. Pour simplifier, nous pouvons imaginer les bases des données comme un **ensemble d'informations qui sont organisées en forme d'une ou plusieurs tables.**

Nous pouvons définir une table comme **une collection de données de même nature**. Les colonnes représentent les champs et les lignes les enregistrements.

Par exemple, le contenu de l'ensemble des Régions de France organisée par code géographiques, pourrait être représenté par la table suivante :

Nom	**Code**
Ardèche	07
Manche	50
Paris	75

Une table est constituée par:

- **Colonnes:** C'est l'ensemble de valeurs de même type correspondant à une même propriété. Dans l'exemple précèdent on à deux colonnes, la colonne « Nom » et la colonne « Code ».
- **Lignes**. Une ligne est une suite de **valeurs**, **chacune d'un type déterminé**. De façon générale, nous pouvons dire qu'une ligne regroupe des informations concernant un objet, un individu, un événement, etc... C'est à dire un concept du monde réel que nous appellerons une **entité**.

1.4 Valeurs et types

Chaque valeur correspond à un type qu'on a déterminé pendant la conception et la création de notre table. Il est important d'utiliser le type correct pour les valeurs de chaque colonne.
Le conséquences d'un mauvais choix peut être :

- **Problèmes de performance** (ex. : il est plus rapide de faire une recherche sur un nombre que sur une chaîne de caractères)

- **Gaspillage de mémoire** (ex. : si vous stockez de toutes petites données dans une colonne faite pour stocker de grosses quantités de données)
- **Comportement inattendu** (ex. : trier sur un nombre stocké comme tel ou sur un nombre stocké comme une chaîne de caractères ne donnera pas le même résultat)
- **Impossibilité d'utiliser des fonctionnalités propres à un type de données** (ex. : stocker une date comme une chaîne de caractères vous prive des nombreuses fonctions temporelles disponibles).

Voici une liste de quelques types de données sur MySQL:

Type de données	Exemple
CHAR (20)	'Chaine de texte'
VARCHAR (20)	'Chaine de texte'
SMALLINT, BIGINT ou INTEGER	7500
NUMERIC ou DECIMAL	3425.432
REAL, FLOAT ou DOUBLE PRECISION	6.626E-34
BOOLEAN	TRUE
DATE	'1957-08-14'

Si vous voulez une liste plus exhaustive des types de données, vous pouvez suivre ce lien vers la documentation des types supportés par MySQL.

La valeur NULL

Parfois, au moment d'enregistrer des données nous pouvons nous retrouver dans une situation où nous **ne savons pas la valeur d'une colonne** dans une ligne déterminée. On peut, donc , considérer la valeur NULL comme l'absence de valeur. **Il ne s'agit pas d'un type à proprement parler**, mais d'une valeur possible dans tous les types.
Par contre, **il ne faut pas confondre la valeur NULL avec d'autres valeurs**. comme 0, FALSE ou avec la chaîne espace de caractère.

Disposer de la valeur NULL est utile, mais il faut aussi savoir que si on l'utilise d'une manière inadéquate **elle peut être source de problèmes importants**.

Si nous interdisons ou non l'assignation de la valeur NULL à une colonne, celle-ci devient une **colonne obligatoire** ou une **colonne facultative** respectivement.

1.5 Identifiants ou clés

A quoi sert une clé ou un identifiant?

Une clé permet de:

- **Distinguer une ligne d'une autre ligne quelconque** (Clé primaire) dans la même table .
- **Etablir des références** (Clé étrangère) vers d'autres tables.

Nous pouvons distinguer deux types de clés:

- **Clé primaire :** « une colonne dont les valeurs identifient chaque ligne de la table **d'une manière unique** ». C'est important de noter qu'une clé primaire ne peut avoir la valeur NULL.

- **Clé étrangère :** « une colonne, ou un groupe de colonnes, dans une table qui correspond à (ou **référence**) une clé primaire **dans une autre table**. Une clé étrangère n'a pas à être unique elle-même, mais elle doit identifier de manière unique la ou les colonnes qu'elle référence dans l'autre table. »

Pour illustrer ces deux concepts, nous allons utiliser une base de données qu'on appellera BIBLIOTHEQUE
qui possède deux tables: Auteur et Livre.

Table Auteur :

Id	Nom	Prénom
4	Proust	Marcel
9	Hugo	Victor

Dans la table Auteur la **clé primaire (*Pk ou Primary Key en anglais*)** correspond à la colonne Id. A chaque fois qu'une nouvelle ligne est insérée dans la base de données, la valeur du nouveau Id appartenant à cette ligne sera unique.

Table Livre :

Id	Titre	Auteur_id
1	Du côté de chez Swann	4
2	Les Misérables	9

Ici nous retrouvons une clé primaire dans la colonne Id mais aussi **une clé étrangère (*Fk ou Foreign Key en anglais*)** dans la colonne Auteur_id. Cela nous permet de créer des relations entre les deux tables comme le montre l'image suivante :

1.6 Schéma et contenu

Nous distinguons deux parties dans une base de données: **contenu** et **schéma** .

Le **contenu** est l'ensemble de lignes dans la table.

Le **schéma** dans une base de données définit la structure d'une table. Il nous informe des:

- **Les colonnes** que possède la table.
- **Les types** de valeurs el le caractère facultatif ou obligatoire de chaque valeur.
- **Les clés** que possède la table.

1.7 Les contraintes d'intégrité

Les contraintes d'intégrité assurent que pendant une manipulation des données (**ajouter** une ligne, la **supprimer** ou tout simplement **modifier** une valeur) certaines règles soient respectées pour que les données ne perdent pas leur intégrité.

A) Contraintes d'unicité :
Une clé ou identifiant impose que à tout instant les lignes d'une table possèdent des **valeurs distinctes pour une ou plusieurs colonnes**. Par exemple, si nous reprenons l'exemple de notre table livre, nous ne pouvons pas avoir deux lignes avec la même valeur de Id.

Cette contrainte s'applique lors de:

- **La création d'une ligne**. Action autorisée uniquement s'il n'existe pas une autre ligne avec la même valeur de l'identifiant ou clé.
- **La modification de l'identifiant d"une ligne**. Action autorisée seulement s'il n'y a pas une autre ligne avec la nouvelle valeur de l'identifiant ou clé que nous voulons modifier.

La contrainte ne s'applique pas lors de la suppression d'une ligne.

B) Contraintes référentielles (clés étrangères) :
Une clé étrangère déclaré comme obligatoire (c'est à-dire qu'elle ne puisse pas avoir la valeur NULL) doit
contenir une valeur qu'on retrouve comme clé ou identifiant primaire d'une ligne dans une autre table.
Selon les actions qu'on réalise nous appliquerons certaines contraintes:

- **Création d'une ligne de la table Livre :**

  Si nous créons une ligne et la colonne Auteur_id est obligatoire, la valeur doit être présente dans la colonne Id de la table Auteur.

- **Suppression d'une ligne de la table Livre :**

  Nous pouvons effectuer cette action sans restriction.

- **Modification de la valeur de la colonne Auteur_id d'une ligne de la table Livre :**

  Si nous modifions la valeur de la colonne Auteur_id et la colonne Auteur_id est obligatoire, cette nouvelle valeur doit être présente dans la colonne Id d'une ligne de la table Auteur.

- **Création d'une ligne de table Auteur :**

  Nous pouvons effectuer cette action sans restriction.

- **Suppression d'une ligne de la table Auteur :**

  Quand on supprime une ligne de la table Auteur il y a plusieurs actions qui laissent la base de données dans une état correct. Les plus importantes sont :

  - **Blocage ou *no action* :**

    Nous empêchons la suppression de la ligne Auteur pour ne pas laisser des lignes orphelines de la table Livre.

  - **Propagation ou *cascade :***

    Nous supprimons la ligne de la table Auteur et les lignes de Livre qui la référencent.

  - I**ndépendance ou *set null* :**

Si la clé étrangère est facultative, nous assignons la valeur NULL de la colonne Auteur_id aux lignes des la table Livre qui référencent la ligne de la table Auteur à supprimer.