

Création d'une base de données SQL

Table des matières

I. Contexte	3
II. Notions de structure de données	3
III. Exercice : Appliquez la notion	5
IV. Création d'une structure simple	6
V. Exercice : Appliquez la notion	11
VI. Compléter la structure	11
VII. Exercice : Appliquez la notion	15
VIII. Optimisations	15
IX. Exercice : Appliquez la notion	19
X. Essentiel	19
XI. Auto-évaluation	19
A. Exercice final	19
B. Exercice : Défi.....	21
Solutions des exercices	21

I. Contexte

Durée : 1 h

Environnement de travail : XAMPP

Pré-requis :

- Avoir installé XAMPP
- Notions de bases de données

Contexte

Le but d'une base de données est de stocker, traiter et fournir des données. Afin de pouvoir remplir son rôle, la base de données ne conserve pas les données en vrac : elle dispose d'une structure dans laquelle elles sont stockées méticuleusement.

Un annuaire, par exemple, va gérer des données de type *nom de famille*, *prénom*, *numéro de téléphone* et *e-mail*, sous la forme de données texte. La base de données prendra bien soin de ranger les noms de famille dans la case « Nom de famille », les prénoms dans la case « Prénom », etc.

Il est important de dissocier les données de la structure qui va les accueillir : la structure est là pour fournir un cadre permettant de définir quelles données seront conservées et de quelle façon.

Il n'est pas possible de conserver des données dans un SGBD relationnel sans définir au préalable une structure : ce n'est qu'une fois que celle-ci a été déterminée qu'il est possible d'y stocker des données et de les manipuler.

Ce cours traite donc de structure de données, et non des données elles-mêmes.

II. Notions de structure de données

Objectif

- Acquérir les notions de structure de données

Mise en situation

Avant de manipuler des bases de données, il est important de savoir précisément ce que c'est, à quoi elles servent et comment elles fonctionnent.

Définition

Base de données et SGBD

Une base de données (BDD) est un ensemble de données structurées : elle contient à la fois une structure de données et des données.

Un système de gestion de base de données (SGBD) est un logiciel fournissant une base de données fonctionnelle.

Ce logiciel fournit non seulement une base de données, mais également les outils permettant de la gérer.

Structures de données

Il existe plusieurs types de SGBD. Ce cours s'appuie sur MariaDB, qui appartient à la famille des SGBDR (R pour *relationnel*).

Un SGBD fonctionne avec un moteur de base de données. Chaque SGBD dispose d'au moins un moteur, certains, comme MariaDB, en possèdent plusieurs, laissant à l'utilisateur le choix.

Remarque

Le sujet des moteurs de base de données est un sujet très complexe qui dépasse très largement le cadre de ce cours, c'est pourquoi le moteur par défaut de MariaDB sera utilisé. Ce moteur est appelé InnoDB.

Définition Tables, colonnes et lignes

Dans une base de données relationnelle comme MariaDB, les données sont organisées en tableaux à deux dimensions, appelés « tables ».

Les colonnes de ces tables représentent la structure des données. Par exemple, une table conservant des données *nom*, *prénom*, *téléphone* et *e-mail* contiendra une colonne pour les noms, une colonne pour les prénoms, et ainsi de suite.

Une ligne de cette table représente un enregistrement. Pour enregistrer la données « Jean Dupont », la base de données inscrira donc sur la même ligne « Dupont » dans la colonne Nom, et « Jean » dans la colonne Prénom.

Le croisement d'une ligne et d'une colonne représente donc une donnée unitaire, appelée « champ » ou « attribut ».

Exemple

Voici à quoi pourrait ressembler la table en question, avec sa structure et ses données :

Table des personnes				
Nom des colonnes	Nom	Prénom	Téléphone	Email
Type de données et longueur	Texte(50)	Texte(50)	Texte(10)	Texte(250)
Contenu de la table	Dupond	Jean	01234567	jean.dupond@email.com
	Dupont	Jeanne	06010101	jeanne.dupont@email.com
	Dupré	Jeanine	07010101	jdupre@email.com

Le rôle du SGBD

Les données sont écrites sur le stockage physique (disque dur, SSD...) de la machine hébergeant la base de données.

Le SGBD est en fait une abstraction du stockage physique : au lieu d'écrire directement les données sur le disque dur de la machine, on utilise un outil qui va s'en charger.

L'intérêt de cet outil est qu'il dispose de nombreuses fonctions : ce n'est pas qu'un simple outil de lecture et d'écriture. Par exemple, il permet de trouver rapidement un ensemble de données selon de nombreux critères.

Il est toutefois important de se rappeler qu'un ordinateur n'est capable de gérer que des données binaires, c'est-à-dire des suites de 0 et de 1. Un ordinateur ne stocke pas de dates, de nombres ou du texte, mais simplement des 0 et des 1. C'est le SGBD qui fait la traduction entre le binaire et la donnée.

Une même chaîne binaire peut avoir des sens totalement différents selon la traduction que l'on utilise. Le SGBD va donc à la fois retenir la chaîne binaire (la donnée) et son format (la structure).

Attention

Il ne s'agit pas seulement de savoir si une donnée est un texte ou un nombre : il existe plusieurs formats pour une même chose. Par exemple, la date 01/02/2020 peut correspondre à la fois au 1^{er} février 2020 ou au 2 janvier 2020. En effet, en Europe, les dates sont au format jour/mois/année, mais aux États-Unis, c'est le format mois/jour/année qui est plébiscité.

Une donnée sans format n'a aucune signification.

Les différents formats de données

Il existe de nombreuses normes :

- Pour les dates, les plus fréquentes sont JJ/MM/AAAA, MM/JJ/AAAA et AAAA/MM/JJ.
- Pour les nombres, on différencie les nombres entiers, les nombres décimaux (qui représentent les valeurs exactes des nombres à virgule) et les nombres à virgule flottante (qui représentent une approximation, moins coûteuse en place, mais peu adaptée pour le calcul).
- Le cas du texte est plus complexe. Non seulement il y a de nombreux alphabets, mais de nombreux caractères spéciaux existent au sein d'un même alphabet (tels que l'æ immortalisé par Serge Gainsbourg ou le ñ, présent dans de nombreux noms ibériques).
 - Les correspondances entre un caractère et sa représentation binaire sont rassemblées dans des tables d'encodage. La table la plus connue est la table ASCII, mais il s'agit d'une table très simple, qui n'est pas appropriée pour un usage moderne.
 - Il existe une norme universelle appelée Unicode. Plusieurs encodages différents sont issus de cette norme. Le plus utilisé est l'UTF-8, et c'est celui qui sera recommandé dans le cadre de ce cours. Dans MariaDB, l'encodage correspondant à l'UTF-8 se nomme **utf8mb4**.

Attention

Le fait d'utiliser plusieurs encodages différents au sein d'un même système peut aboutir à des problèmes extrêmement complexes. Il est préférable de s'en tenir à l'UTF-8.

Syntaxe À retenir

- Une base de données relationnelle stocke les données sous la forme de tables.
- Chaque table contient une liste de colonnes ayant chacune un nom et un type de données.
- Une ligne de la table représente un enregistrement.
- Il faut faire très attention au type d'une donnée.
- Toujours privilégier l'UTF-8 pour le texte.

Complément

Base de données¹

III. Exercice : Appliquez la notion

En installant XAMPP, votre environnement local, vous avez également installé un moteur de base de données MySQL. Il dispose d'une interface en ligne permettant de visualiser et manipuler les tables de vos bases de données. Ce premier exercice va vous permettre de vous familiariser avec cet outil.

Question 1

[solution n°1 p.23]

Lancez XAMPP et connectez-vous à PHPMyAdmin via l'adresse <http://localhost/phpmyadmin/>. Dans l'interface, cliquez sur l'onglet **SQL** et exécutez les requêtes suivantes :

```
1 DROP SCHEMA IF EXISTS introduction;
2 CREATE SCHEMA introduction;
3 CREATE TABLE introduction.clients
4 (
5     Id int not null primary key auto_increment,
```

¹ https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es

```

6   Prenom varchar(250),
7   Nom varchar(250),
8   Email varchar(250),
9   DateNaissance date
10 );
11 INSERT INTO introduction.clients(Prenom, Nom, Email, DateNaissance) VALUES
12 ('John', 'Doe', 'john@doe.com', '1987-11-21'),
13 ('Sarah', 'Dupond', 'sarahdupond@monmail.com', '1985-01-16'),
14 ('Laure', 'Mondi', 'lmondi@mailiam.org', '1992-05-28');
```

Dans la colonne de gauche, la base de données « Introduction » a dû apparaître. Cliquez dessus pour afficher la liste des tables de cette base de données. La requête ci-dessus en a créé une, quel est son nom ?

Question 2

[solution n°2 p.23]

Cliquez sur la table pour voir son contenu. Combien de lignes de données possède-t-elle ?

Question 3

[solution n°3 p.23]

Quel est le format de la date de naissance ?

IV. Création d'une structure simple

Objectif

- Découvrir les instructions de création de structure

Mise en situation

Les données sont contenues dans des tables, elles-mêmes contenues dans des bases de données. La première chose à faire est donc de créer une base de données.

Remarque Le terme « base de données »

Le terme « base de données » peut désigner plusieurs choses. On l'utilise souvent pour désigner le SGBD, donc l'ensemble logiciel incluant la base de données proprement dite.

Ici, le terme de base de données détermine en fait un compartiment au sein de la base de données proprement dite, compartiment qui va héberger les tables.

Pour plus de clarté, le terme employé à présent sera celui de « schéma ».

Syntaxe Créer un schéma

Il existe deux instructions permettant de créer un schéma :

- CREATE DATABASE NomDuSchema;
- CREATE SCHEMA NomDuSchema;

Ces deux instructions sont équivalentes.

Remarque

Un schéma peut être comparé à un dossier dans un système de fichiers. Une base de données doit contenir au moins un schéma pour accueillir les tables.

Syntaxe Créer une table

L'instruction pour créer une table est :

```
1 CREATE TABLE NomDeTable
2 (
3 Colonne1 Type_de_donnée Modificateurs,
4 Colonne2 Type_de_donnée Modificateurs,
5 (...)
6 );
7
```

Exemple

```
1 CREATE TABLE Personnes
2 (
3     Nom varchar(50) not null,
4     Prenom varchar(50) not null,
5     Telephone varchar(10) null,
6     email varchar(250)
7 )
```

Ce code permet de créer une table nommée Personnes, contenant les colonnes suivantes :

- **Nom**, une colonne de type texte de maximum 50 caractères, ne pouvant être vide
- **Prenom**, une colonne de type texte de maximum 50 caractères, ne pouvant être vide
- **Telephone**, une colonne de type texte de maximum 10 caractères. Cette colonne est explicitement décrite comme pouvant être vide.
- **Email**, une colonne de type texte de maximum 250 caractères. Rien d'autre n'est spécifié, donc cette colonne peut être vide.

Définition

Dans une instruction `CREATE TABLE`, une colonne est décrite de la façon suivante : `Nom_de_colonne Type_de_donnée Modificateurs`.

- Le nom de colonne doit être unique au sein d'une même table.
- Le type de données est un type disponible pour le SGBD considéré.
- Le modificateur ne s'applique que pour la colonne en question.
- Toutes les définitions de colonnes doivent se terminer par une virgule, à l'exception de la dernière.

Il existe de nombreux types de données possibles pour nos champs. Voici une liste (non exhaustive) des types les plus courants.

Type de donnée : nombres entiers

Il existe plusieurs types de nombres entiers en fonction de la fourchette de valeurs permises.

Type	Taille (octets)	Minimum	Maximum
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INT	4	-2147483648	2147483647
BIGINT	8	-2^{63}	$2^{63}-1$

Conseil

À moins de gérer des milliards d'enregistrements, essayer d'optimiser en prenant la taille la plus adaptée n'est pas forcément utile. Le type `INT` répond à la grande majorité des besoins.

Type de donnée : nombres décimaux

Une base de données dispose d'un espace de stockage fini. Par conséquent, une base de données ne stocke que des nombres décimaux, pas des nombres réels.

Il existe principalement deux formats pour les nombres non-entiers dans une base de données :

- Les formats dits décimaux, qui sont constitués d'un nombre entier et d'une puissance de 10 fixe
- Les formats dits à virgule flottante, qui sont constitués d'un nombre entier et d'une puissance de 10 variable

Définition Le type décimal

Un décimal en base de données est défini par un entier constitué de x chiffres et d'un second entier déterminant la position de la virgule.

La position de la virgule est commune à toutes les données de la colonne.

Le format s'écrit : `DECIMAL(x, y)` avec x étant le nombre de chiffres significatifs (de 1 à 65) et y étant le nombre de chiffres après la virgule (de 0 à 38).

Exemple

```
1 Prix DECIMAL(10,2),
2 Quantité DECIMAL(4,0),
3 Taille DECIMAL(10,5)
```

Dans ces exemples :

- Prix est un décimal contenant 10 chiffres avant la virgule et 2 après (1465789452.32, par exemple).
- Quantité est un décimal contenant 4 chiffres avant la virgule (1246). Il aurait sans doute été préférable de le définir comme un `INT`.
- Taille est un décimal contenant 10 chiffres avant la virgule et 5 après.

Définition Nombres à virgule flottante (base de données)

Un nombre à virgule flottante en base de données est défini par un entier constitué de x chiffres, appelé « mantisse », et d'un second entier déterminant la position de la virgule, appelé « exposant ».

De la sorte, le nombre stocké correspond à : $\text{Mantisse} * 10^{\text{exposant}}$.

Il existe deux tailles de format, sur 4 et 8 octets : `FLOAT` et `DOUBLE`.

Exemple

```
1 Temperature FLOAT,  
2 DistanceTerreLune DOUBLE
```

Attention

Pour des raisons complexes qui sortent du cadre de ce cours, les nombres à virgule flottante ne sont pas adaptés pour des valeurs décimales précises, telles que des prix, des quantités... Ce sont des valeurs approchées, et non pas des valeurs exactes.

Les nombres à virgule flottante sont adaptés pour des mesures scientifiques, comme les valeurs renvoyées par des capteurs.

Il est très fortement recommandé d'utiliser `DECIMAL` dans tous les types de colonnes n'étant pas destinés à conserver des données scientifiques.

Seul le format `DECIMAL` garantit des calculs exacts sur les valeurs du quotidien.

Définition **Texte de longueur fixe**

Dans une colonne de type texte de longueur fixe, un texte sera tronqué s'il est trop long, et complété par des espaces à la suite s'il est trop court. Le mot-clé correspondant est : `CHAR (Longueur)`.

La longueur doit être comprise entre 1 et 255.

Définition **Texte de longueur variable**

Dans une colonne de type texte de longueur variable, un texte sera tronqué s'il est trop long. Par contre, il ne sera pas complété par des espaces s'il est trop court, il sera conservé tel quel. Le mot-clé correspondant est : `VARCHAR (Longueur)`.

La longueur doit être comprise entre 1 et 255.

Conseil

Le type `VARCHAR` est de très loin le plus utilisé des deux. Il est préférable de l'employer afin de ne pas avoir à se préoccuper des espaces supplémentaires.

Définition **Texte long**

Lorsqu'un texte est plus long que 255 caractères, il existe un autre format, moins performant, appelé `TEXT` : `TEXT(Longueur)`. Longueur doit être compris entre 1 et 65535.

Notez toutefois qu'il n'est pas obligatoire de préciser la longueur du format `TEXT`. Si vous utilisez `TEXT` sans la spécifier, vous pourrez tout de même stocker un texte d'une longueur, comme indiquée, comprise entre 1 et 65535.

Exemple

```
1 Code CHAR(8),  
2 Nom VARCHAR(50),  
3 Description TEXT(5000)  
4 Commentaire TEXT
```

Définition Date et heure

MariaDB propose plusieurs formats pour conserver les dates et les heures. Les plus utilisés sont :

- DATETIME pour conserver à la fois la date et l'heure
- DATE pour ne conserver que la date
- TIME pour ne conserver que l'heure

Exemple

```
1 VenduLe DATETIME,
2 DateVente DATE,
3 HeureVente TIME
```

Dans ces exemples, si une vente a été conclue le 5 janvier 2020 à 16h30 :

- VenduLe peut contenir la totalité de l'information : 2020-01-05 16:30:00
- DateVente peut contenir la date : 2020-01-05
- HeureVente peut contenir l'heure : 16:30:00

Définition Modificateurs de colonne

Il existe de nombreux modificateurs en fonction des SGBD. Les plus courants sont :

- NOT NULL : indique qu'une colonne ne peut pas avoir une valeur vide de données.
- NULL : indique qu'une colonne peut avoir une valeur vide de données. Ce paramètre est implicite.
- DEFAULT valeur : indique que, si une ligne est créée dans la table sans qu'une valeur ne soit précisée pour cette colonne, la colonne prendra la valeur spécifiée.

Modifier la structure d'une table

Il est parfois nécessaire de devoir modifier ou supprimer une table.

Supprimer une table efface bien évidemment toutes les données de la table.

Supprimer une colonne efface les données de la colonne.

Ajouter une colonne n'affectera pas les données déjà existantes.

Définition Supprimer une table ou une colonne

Pour supprimer une table, l'instruction à employer est : DROP TABLE Nom_de_la_table.

Pour supprimer une colonne, l'instruction à employer est : ALTER TABLE Nom_de_la_table DROP COLUMN Nom_de_la_colonne.

Syntaxe Ajouter ou supprimer une colonne

La modification de la structure d'une table se fait grâce à l'instruction ALTER TABLE, suivie de l'action à réaliser.

- Pour ajouter une colonne, il faut utiliser la requête SQL suivante :

```
ALTER TABLE Nom_de_la_table
```

```
ADD Nom_de_colonne Type_de_colonne Modificateurs
```

- Et pour supprimer une colonne :

```
ALTER TABLE Nom_de_la_table
```

```
DROP Nom_de_colonne
```

Syntaxe **Modifier une colonne existante**

- Pour modifier la définition d'une colonne :

```
ALTER TABLE Nom_de_la_table
```

```
MODIFY Nom_de_colonne Type_de_colonne Modificateurs
```

- Et pour renommer une colonne :

```
ALTER TABLE Nom_de_la_table
```

```
CHANGE Ancien_nom Nouveau_nom Type_de_colonne Modificateurs
```

Syntaxe **À retenir**

- CREATE DATABASE permet de créer un schéma.
- CREATE TABLE, suivie de la définition des colonnes, permet de créer une table et ses colonnes.
- ALTER TABLE permet de modifier les colonnes.
- DROP TABLE permet de supprimer une table.
- **Attention aux données en modifiant une table déjà existante !**

Complément

Base de données¹

V. Exercice : Appliquez la notion

Question

[solution n°4 p.23]

Une entreprise souhaite disposer d'un registre contenant la liste des employés. Cette table pourra contenir les valeurs suivantes :

- Nom
- Prénom
- Ville
- Code postal
- Numéro de téléphone
- Code de badge : un code de 10 caractères

Écrivez une instruction CREATE TABLE pour créer cette table, en choisissant les formats appropriés.

Indice :

Le code postal peut commencer par un 0 : le type INTEGER n'est pas adapté.

VI. Compléter la structure

¹ https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es

Objectif

- Créer des relations entre plusieurs tables

Mise en situation

Un schéma de base de données contiendra parfois de très nombreuses tables. Habituellement, chaque table contient une structure de données.

Il serait possible de tout mettre dans une seule table, mais tout l'intérêt d'une base de données est de pouvoir structurer les données sous la forme de schémas, puis de tables, puis de colonnes.

Par exemple, un magasin voudra gérer une liste de produits en vente, une liste des ventes, une liste des vendeurs, et une liste des clients. Cela fera donc au moins 4 tables.

Identifiants

Une table peut contenir de nombreuses colonnes, et encore plus de lignes. Pour simplifier les traitements, il est d'usage de disposer d'une ou plusieurs colonnes permettant d'identifier chaque ligne.

La valeur de cette colonne (ou la combinaison des valeurs de ces colonnes, s'il y en a plusieurs) sera unique pour chaque ligne, de sorte que cette valeur (ou combinaison de valeurs) puisse identifier la ligne en question.

Il peut s'agir d'une adresse e-mail, d'un code produit, ou tout simplement d'un numéro de ligne.

Définition

Une donnée capable d'identifier un enregistrement unique au sein d'une table est appelé « clé primaire ».

Remarque

Une clé primaire est habituellement constituée d'une seule colonne, mais elle peut être composée de plusieurs colonnes : ce type de clé est nommée « clé primaire composite ». Par définition, la ou les colonnes de la clé primaire ne peuvent pas recevoir de valeur nulle.

Syntaxe

Pour une colonne seule, il est possible de la définir comme clé primaire au moyen d'un modificateur dans sa définition : `Colonne Type_de_donnée NOT NULL PRIMARY KEY`.

Si la clé primaire est composée de plusieurs colonnes, il faudra alors rajouter l'instruction suivante après la définition des colonnes (n'oubliez pas les virgules à la fin de toutes les lignes, sauf la dernière) : `PRIMARY KEY (Liste des colonnes)`.

Le cas où notre clé primaire est composée de plusieurs colonnes est appelée « clé primaire composite ». On se servira de la « clé primaire composite » dans le cas où l'on souhaite stocker plusieurs données sur un seul ID. Par exemple, si on veut stocker un même article traduit dans plusieurs langues, on sera amené à utiliser le système de « clé primaire composite » afin d'éviter de créer une table pour chaque langue, ce qui surchargerait très vite le nombre de table dans notre base de données. Dans ce cas, on va éviter l'AUTO_INCREMENT et uniquement définir quelles colonnes seront utilisées.

Pour des raisons pratiques et utiles, il est vivement conseillé de respecter la convention « id_nomDeLaTable » pour le nom de votre ID. Si je crée une table qui s'appelle « employes », j'utiliserai alors « id_employes » comme nom.

Exemple

```
1 CREATE TABLE Personnes
2 (
3     NumeroUnique INT NOT NULL PRIMARY KEY,
4     Nom VARCHAR(50) NOT NULL,
5     Prenom VARCHAR(50) NOT NULL
6 )

1 CREATE TABLE Personnes
2 (
3     NumeroNonUnique INT NOT NULL,
4     Nom VARCHAR(50) NOT NULL,
5     Prenom VARCHAR(50) NOT NULL,
6     PRIMARY KEY (NumeroNonUnique , Nom, Prenom)
7 )
```

Notez la virgule en fin de ligne dans la définition de `Prenom` dans le second exemple.

Autre exemple possible :

```
1 CREATE TABLE article (
2     id_article INT NOT NULL,
3     lang CHAR(2) NOT NULL,
4     title VARCHAR(255) NOT NULL,
5     author VARCHAR(255) NOT NULL,
6     PRIMARY KEY (id_article, lang)
```

Pour y ajouter des données, vous pouvez alors utiliser :

```
1 INSERT INTO article (id_article, lang, title, author) VALUES ("1", "fr", "Initialiser une Base
De Données", "");
2 INSERT INTO article (id_article, lang, title, author) VALUES ("1", "en", "Report", "Initialize
a Database");
3 INSERT INTO article (id_article, lang, title, author) VALUES ("2", "fr", "Tutoriel", "SQL pour
les nuls");
4 INSERT INTO article (id_article, lang, title, author) VALUES ("2", "en", "Tutorial", "SQL for
Dummies");
```

Remarque

Les colonnes intervenant dans une `PRIMARY KEY` doivent recevoir le mot-clé `NOT NULL`.

Une clé primaire est souvent définie comme un compteur. La base de donnée se chargera de renseigner elle-même le numéro de la ligne.

Un compteur est défini dans MariaDB par le mot-clé `AUTO_INCREMENT`.

Exemple

```
1 CREATE TABLE Personnes
2 (
3     Compteur INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
4     Nom VARCHAR(50) NOT NULL,
5     Prenom VARCHAR(50) NOT NULL
6 )
```

Les contraintes

Il est fréquent que deux tables différentes contiennent des données ayant un rapport entre elles. Par exemple, une table de ventes va contenir le code de l'article vendu, article qui est décrit dans une table des articles.

Si rien n'est précisé, rien n'empêche d'avoir une vente qui contient un code article qui n'existe pas. Il est possible de forcer la vérification, de sorte qu'il ne soit pas possible d'enregistrer une vente sur un article non existant. Cela s'appelle une **contrainte**.

Définition Clé étrangère

Une contrainte de clé étrangère de A vers B est un mécanisme par lequel le SGBD vérifiera systématiquement qu'une donnée existe dans la table B avant d'autoriser une manipulation dans la table A.

L'instruction pour créer une contrainte est :

```
FOREIGN KEY (liste des colonnes à vérifier de la tableA)
```

```
REFERENCES TableB(liste des colonnes correspondantes de la table B)
```

Cette instruction doit être ajoutée dans la définition de tableA lors de son instruction `CREATE TABLE`, ou peut être ajoutée par la suite via un `ALTER TABLE`.

Exemple

```
1 CREATE TABLE Article
2 (
3     Code varchar(8) NOT NULL PRIMARY KEY,
4     Prix DECIMAL(10,2)
5 );
6
7 CREATE TABLE Ventes
8 (
9     NumeroVente INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
10    CodeArticle varchar(8) NOT NULL,
11    Quantite INT,
12    FOREIGN KEY (CodeArticle)
13        REFERENCES Article (Code)
14 );
```

Ce code permet de créer une table Article, dont chaque article est défini par un code unique, et une table Ventes, dont le code article doit se trouver dans la colonne Code de la table Article.

Remarque Différentes clés

Il existe plusieurs façons de définir une clé primaire :

- Par une valeur purement technique, comme un compteur géré par la base de données,
- Par une valeur métier définie comme unique, comme un code ou un numéro de série,
- Par une combinaison de plusieurs valeurs, où chaque valeur n'est pas forcément unique, mais la combinaison l'est.

Il est tentant de vouloir utiliser une valeur métier que l'on pense être unique. Cependant, cela peut poser plusieurs problèmes : tout d'abord, il faut que cette valeur soit réellement unique. Cela n'est pas forcément garanti et, si c'est le cas dans le présent, cela ne le sera plus forcément à l'avenir. Ensuite, il faut que cette valeur ne change pas avec le temps. On ne peut pas garantir que le code d'un produit, par exemple, ne changera jamais.

Dans bon nombre de cas, il est préférable de s'appuyer sur une valeur purement technique qui sera garantie unique. Cette valeur n'aura pas de signification réelle et ne devra jamais en avoir.

Syntaxe **À retenir**

- Une clé primaire permet d'identifier une ligne d'une table de façon unique.
- Une clé étrangère permet de contrôler si une valeur existe dans une table de référence avant toute écriture dans une table.

ComplémentBase de données¹

VII. Exercice : Appliquez la notion

Question 1

[solution n°5 p.23]

Une entreprise dispose d'un registre contenant la liste des employés, définie par l'instruction suivante :

```
1 CREATE TABLE Employes
2 (
3     Nom varchar(250),
4     Prenom varchar(250),
5     Ville varchar(250),
6     CodePostal varchar(5),
7     Telephone varchar(20)
8 )
```

Modifiez ce code afin de rajouter une nouvelle colonne, qui servira de clé primaire auto-incrémentée.

Indice :

Le type de données `INT` est très adapté pour un identifiant auto-incrémenté.

Indice :

Attention aux virgules en fin de ligne, sauf pour la dernière.

Question 2

[solution n°6 p.23]

Chaque employé possède un badge ayant un identifiant auto-incrémenté, un code de 20 caractères maximum, une date d'expiration, une heure de début d'accès et une heure de fin d'accès. Créez une table `Badges` permettant de gérer ces données, et modifiez la table `Employes` de manière à lier chaque employé à son badge.

VIII. Optimisations

Objectif

- Comprendre le fonctionnement d'une base de données

Mise en situation

Comment un SGBD fait-il pour gérer les données qu'il contient ?

La réponse est assez complexe, mais, pour simplifier, bien qu'un SGBD soit très sophistiqué, il fonctionne de façon similaire à un être humain, en plus performant.

Par exemple, s'il doit rechercher une donnée précise, il va parcourir la liste des données dont il dispose. Sur une base de données importante, cela peut prendre du temps. Il est cependant possible d'aider le SGBD.

1 https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es

Méthode Fonctionnement d'une clé primaire

Un bon exemple d'optimisation est celui d'un dictionnaire par rapport à un livre.

Pour trouver un mot précis dans un livre ordinaire, il est nécessaire de parcourir le livre du début à la fin, et donc de lire toutes les pages.

Ce n'est pas le cas dans un dictionnaire. Pour y trouver un mot, il n'y a pas besoin de parcourir toutes les pages : comme les pages sont rangées dans un certain ordre, il est possible de trouver rapidement la page souhaitée en se basant sur cet ordre.

Dans le chapitre précédent, une clé primaire a été définie comme une référence unique de ligne pour une table. Mais une clé primaire est plus que cela : elle détermine également l'ordre dans lequel les données sont conservées dans une base de données.

La clé primaire va donc aider le SGBD à manipuler les données d'une table.

```
1 CREATE TABLE Ventes
2 (
3     -- La table Ventes sera triée par leur numéro
4     NumeroVente INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
5     CodeArticles varchar(8) NOT NULL,
6     Quantite INT,
7     FOREIGN KEY (CodeArticles)
8         REFERENCES Articles (Code)
9 );
```

Conseil

Définir des clés primaires dans les tables d'une base de données est une bonne pratique. Il est habituel d'utiliser la colonne « id » pour choisir sa clé primaire. Vous pouvez suivre par exemple la convention de nommage « id_nomtable » pour nommer vos clés primaires.

Méthode Fonctionnement d'une clé étrangère

Lorsqu'une contrainte de clé étrangère a été définie entre deux tables, cela va imposer au SGBD d'effectuer une recherche dans la seconde table à chaque fois que la première sera modifiée.

Puisque le SGBD doit effectuer une opération supplémentaire, cela va donc le ralentir, d'autant plus si la seconde table contient un nombre important de données.

Cependant, si cette recherche se fait sur une clé primaire (ce qui est l'usage), la vérification en sera considérablement facilitée.

```
1 CREATE TABLE Articles
2 (
3     Code varchar(8) NOT NULL PRIMARY KEY,
4     Prix DECIMAL(10,2)
5 );
6
7 CREATE TABLE Ventes
8 (
9     NumeroVente INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
10    CodeArticles varchar(8) NOT NULL,
11    Quantite INT,
12    -- A chaque vente, le SGBD va vérifier l'article. Comme le Code est la clé primaire, cette
13    recherche sera rapide.
14    FOREIGN KEY (CodeArticles) REFERENCES Articles (Code)
15 );
```


Conseil

Il est préférable de définir une contrainte de clé étrangère vers la clé primaire d'une autre table.

Les index

Il n'est pas toujours possible, ni souhaitable, de trier les données dans un certain sens : on pourrait vouloir trier sur d'autres colonnes que la clé primaire.

Un exemple est celui d'une encyclopédie : contrairement à un dictionnaire, les articles d'une encyclopédie ne vont pas nécessairement être rangés par ordre alphabétique du titre, mais plutôt par chapitres, en fonction du sujet abordé.

Afin de permettre au lecteur de trouver rapidement un article précis, une encyclopédie fournit généralement deux supports : un sommaire, qui donne la liste des chapitres et des articles dans l'ordre dans lequel ils sont écrits, et un index, qui liste des mots-clés par ordre alphabétique et donne les pages correspondantes.

Un lecteur pourra donc utiliser un index pour trouver rapidement l'article de son choix, sans avoir à parcourir toute l'encyclopédie.

Ce mécanisme existe également dans une base de données : il est appelé « index secondaire », l'index primaire étant un autre nom pour la clé primaire.

Définition **Index secondaire**

Un index secondaire de table dans une base de données est une liste ordonnée de valeurs d'une ou plusieurs colonnes, associée à des références de lignes, permettant à la base de données de trouver rapidement les lignes recherchées.

Remarque

Contrairement à une clé primaire, un index secondaire n'est pas un identifiant de ligne. Il n'a pas à être unique. Il faut préciser si l'index doit avoir des combinaisons uniques ou non.

De même que l'index d'une encyclopédie occupe quelques pages supplémentaires, un index secondaire de base de données occupe un espace supplémentaire.

De plus, lorsque la base de données effectue des opérations sur la table contenant l'index, elle doit donc également le mettre à jour, ce qui prend du temps.

Il est donc bon de se poser la question de la pertinence d'un index secondaire. Apporte-t-il suffisamment pour justifier l'espace et le temps nécessaires ?

Il faut garder à l'esprit qu'un index facilite la lecture d'une donnée, mais en complique la modification. Or, bien que cela dépende des circonstances, une donnée est très souvent plus lue que mise à jour. Par conséquent, un index apporte généralement plus qu'il ne demande.

Conseil

Il est souhaitable de créer un index lorsque les conditions suivantes sont remplies :

- La table comporte un grand nombre d'enregistrements
- Les données des colonnes de l'index font l'objet de recherches fréquentes
- La clé primaire de la table ne permet pas de faciliter ces recherches, soit car les colonnes n'y figurent pas, soit car leur ordre est différent

Il est possible de créer plusieurs index sur une même table.

Exemple

Dans un *listing* de personnes, il n'est pas rare de rechercher par nom ou par e-mail. Or, ces données ne font pas de bonnes clés primaires. Par conséquent, cela peut être une bonne idée d'avoir deux index, un sur les noms (ou noms et prénoms), l'autre sur les e-mails.

Lorsqu'un index est créé sur plusieurs colonnes, l'ordre de ces colonnes est important.

Par exemple, si un index référence une liste d'adresses par les colonnes Pays, Région et Ville, l'index est conçu pour trouver facilement toutes les adresses d'un pays, d'une région et d'une ville. Il pourra également être utilisé pour rechercher toutes les adresses d'un pays, ou d'un pays et d'une région, mais ne pourra pas être utilisé pour rechercher les adresses d'une ville seule, sans que le pays et la région aient été spécifiés.

Remarque

Si, pour une raison quelconque, il n'est pas possible de faire pointer une contrainte de clé étrangère vers la clé primaire d'une autre table, il est malgré tout possible de créer un index sur cette autre table. Cela aidera grandement le SGBD.

Syntaxe

Un index est créé par l'instruction : `CREATE INDEX Nom_de_l_index ON Nom_de_table (Liste des colonnes).`

Exemple

Sur une table `Employes` définie comme suit :

```
1 CREATE TABLE Employes
2 (
3     Id int not null primary key auto_increment,
4     Nom varchar(250),
5     Prenom varchar(250),
6     Ville varchar(250),
7     CodePostal varchar(5),
8     Telephone varchar(20),
9     CodeBadge char(10)
10 )
```

Il est possible de créer un index `Nom, Prenom` grâce à l'instruction suivante :

```
1 CREATE INDEX Index_Nom_Prenom
2 ON Employes
3 (
4     Nom,
5     Prenom
6 )
```

Syntaxe À retenir

- Une clé primaire permet au SGBD de manipuler facilement les données définies par la clé en question.
- Un index secondaire peut être rajouté à une table pour aider le SGBD sur d'autres colonnes que celles de la clé primaire.
- Une contrainte de clé étrangère sera plus performante si elle pointe vers une clé primaire ou un index secondaire.

ComplémentBase de données¹**IX. Exercice : Appliquez la notion****Question**

[solution n°7 p.24]

Une entreprise dispose d'une liste de magasins, définie par l'instruction suivante :

```
1 CREATE TABLE Magasins
2 (
3     Id int not null primary key auto_increment,
4     CodeMagasin varchar(20),
5     Nom varchar(250),
6     Adresse varchar(250),
7     Ville varchar(250),
8     CodePostal varchar(5),
9     Pays varchar(250),
10    Telephone varchar(20)
11 )
```

Écrivez une requête permettant d'ajouter un index secondaire portant sur le pays, la ville et le code postal, afin de simplifier les recherches.

Indice :

L'ordre des colonnes est important.

Indice :

N'oubliez pas de donner un nom à l'index.

X. Essentiel**XI. Auto-évaluation****A. Exercice final****Exercice 1**

[solution n°8 p.24]

Exercice

Un schéma est...

- ☐ Un dossier d'une base de données
- ☐ Un plan dans une base de données
- ☐ Un tableau de relations entre les tables

Exercice

Quelle est l'instruction permettant de créer une table ?

- ☐ ALTER TABLE
- ☐ ALTER DATABASE ADD TABLE
- ☐ CREATE TABLE

¹ https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es

Exercice

Comment modifier la colonne d'une table ?

- ☐ En supprimant la table et en la recréant
- ☐ En supprimant la colonne et en la recréant
- ☐ En utilisant `ALTER TABLE MODIFY COLUMN`

Exercice

Que deviennent les données d'une table après un `ALTER` ?

- ☐ Elles sont conservées
- ☐ Ça dépend de la modification
- ☐ Elles sont supprimées

Exercice

Quel format de données utiliser pour une colonne de prix ?

- ☐ DECIMAL
- ☐ INT
- ☐ FLOAT

Exercice

Quel est le rôle premier d'une clé primaire ?

- ☐ Améliorer les performances d'une contrainte de clé étrangère
- ☐ Identifier de manière unique une ligne et trier les données d'une table
- ☐ Interdire l'accès à une table à un utilisateur non autorisé

Exercice

Une contrainte de clé étrangère permet...

- ☐ De s'assurer que les valeurs d'une colonne sont des valeurs existant dans une autre table
- ☐ De s'assurer qu'il n'est pas possible d'accéder à une table sans inclure la table référencée
- ☐ De garantir que les attributs de la colonne ne peuvent être vides

Exercice

Un index secondaire est...

- ☐ Une contrainte demandant au SGBD de vérifier les données d'une colonne depuis une autre table
- ☐ Une seconde « clé primaire », lorsqu'une seule clé ne suffit pas
- ☐ Un ensemble valeurs/référence de ligne trié selon les valeurs, permettant au SGBD de retrouver facilement les lignes correspondantes

Exercice

Quelle affirmation est correcte ?

- ☐ Un index secondaire a de lourdes contraintes en termes de performances
- ☐ Un index secondaire améliore les performances lorsqu'il est créé à bon escient
- ☐ Un index secondaire peut améliorer les performances, mais ce n'est pas son but premier

Exercice

Lorsqu'une table dispose d'un index secondaire, le SGBD va pouvoir utiliser cet index...

- ☐ Si ses critères de sélection sont compatibles avec la définition de l'index
- ☐ Tout le temps, l'index est là pour l'aider
- ☐ Seulement si les critères de sélection correspondent exactement à la définition de l'index

B. Exercice : Défi

Une chaîne de magasins veut créer une base de données pour gérer son stock de produits.

Il y a plusieurs magasins, qui sont identifiés par leur code postal (il n'y a qu'un seul magasin par secteur). Ils ont également un nom et une adresse.

Un produit est défini par son code (un numéro à 10 chiffres), son libellé et sa description.

Les responsables veulent pouvoir connaître le stock d'un magasin pour chaque produit, donc la quantité de produits disponibles dans chaque magasin.

Question

[solution n°9 p.26]

Écrivez les requêtes de création des tables pour répondre au besoin exprimé par les responsables de l'enseigne de vente.

Il y a trois tables à créer : Magasins, Produits et Inventaires.

N'hésitez pas à incorporer quelques optimisations possibles dans la structure proposée par la chaîne de magasins.

Indice :

Bien que ce ne soit pas précisé dans la demande initiale, car les responsables ne sont pas des informaticiens, il est préférable de rajouter des identifiants uniques auto-incrémentés sur chaque table.

Indice :

Un code postal ne fait pas un bon identifiant de clé primaire ! Par contre, c'est un bon candidat pour un index secondaire.

Indice :

Un code est mieux défini comme texte, même s'il est décrit comme étant « un numéro à 10 chiffres ». En effet, le texte permettra de conserver les 0 à gauche, car ils sont significatifs.

Indice :

Prévoyez toujours large pour un nom ou une adresse.

Indice :

Au sens strict, seuls les identifiants doivent être marqués non nuls. Les autres colonnes n'ont pas nécessairement à l'être, puisque ce n'est pas spécifié dans la demande.

Toutefois, pour les « identifiants métiers », comme le code postal et le code produit, il est quand même préférable de les marquer non nuls.

Solutions des exercices

p. 5 Solution n°1

La table créée par la requête est la table « Clients ».

p. 6 Solution n°2

La table Client possède trois lignes de données, aux noms de *John Doe*, *Sarah Dupond* et *Laure Mondy*.

p. 6 Solution n°3

Les dates en MySQL sont au format AAAA-MM-JJ. On peut remarquer qu'au moment de l'insertion, la date est une chaîne de caractères.

p. 11 Solution n°4

```
1 CREATE TABLE Employes
2 (
3     Nom varchar(250),
4     Prenom varchar(250),
5     Ville varchar(250),
6     CodePostal varchar(5),
7     Telephone varchar(20),
8     CodeBadge char(10)
9 )
```

- Les noms et prénoms peuvent avoir des longueurs très variables dans le monde. Il est courant de trouver des limites à 20 ou 30 dans des bases de données, mais cela peut poser problème pour un petit nombre de personnes.
- Le code postal aurait pu être défini comme un `INT`. L'avantage du `VARCHAR`, ici, est qu'il est possible d'y conserver des valeurs commençant par un 0.
- Le numéro de téléphone est prévu un peu large, il peut ainsi stocker des numéros étrangers.
- Le code de badge peut être un `char(10)` ou un `varchar(10)`, au choix. Étant donné que `char(10)` n'apporte pas un bénéfice substantiel, il est parfaitement raisonnable de lui préférer `varchar(10)`.

p. 15 Solution n°5

```
1 CREATE TABLE Employes
2 (
3     Id int not null primary key auto_increment,
4     Nom varchar(250),
5     Prenom varchar(250),
6     Ville varchar(250),
7     CodePostal varchar(5),
8     Telephone varchar(20)
9 )
```

p. 15 Solution n°6

```

1 CREATE TABLE Badges
2 (
3     Id int not null primary key auto_increment,
4     Code varchar(20),
5     DateExpiration date,
6     HeureDebutAcces time,
7     HeureFinAcces time
8 );
9
10 CREATE TABLE Employes
11 (
12     Id int not null primary key auto_increment,
13     Nom varchar(250),
14     Prenom varchar(250),
15     Ville varchar(250),
16     CodePostal varchar(5),
17     Telephone varchar(20),
18     Badge int,
19     FOREIGN KEY (Badge) REFERENCES Badges (id)
20 )

```

p. 19 Solution n°7

L'énoncé précise bien le pays, la ville et le code postal, dans cet ordre. Il faut donc respecter cet ordre lors de la création de l'index :

```

1 CREATE INDEX Index_Magasins ON Magasins
2 (
3     Pays,
4     Ville,
5     CodePostal
6 )

```

On peut imaginer que l'on cherchera souvent en magasin, de manière progressive : pays, puis ville, puis code postal, mais qu'on ne cherchera jamais un magasin par son code postal seul.

Exercice p. 19 Solution n°8


Exercice

Un schéma est...

- ☒ Un dossier d'une base de données
- ☐ Un plan dans une base de données
- ☐ Un tableau de relations entre les tables
- ☐ Bien qu'un schéma soit synonyme de base de données (au sens logique), il s'agit en réalité d'un dossier dans une base de données (au sens physique).


Exercice

Quelle est l'instruction permettant de créer une table ?

- ☐ ALTER TABLE
- ☐ ALTER DATABASE ADD TABLE
- ☒ CREATE TABLE
-  L'instruction de création est `CREATE TABLE`, suivie du nom et de la définition de la table.


Exercice

Comment modifier la colonne d'une table ?

- ☐ En supprimant la table et en la recréant
- ☐ En supprimant la colonne et en la recréant
- ☒ En utilisant `ALTER TABLE MODIFY COLUMN`
-  Bien qu'il soit toujours possible de supprimer puis de recréer une table ou une colonne, il est également possible d'utiliser l'instruction `ALTER TABLE MODIFY COLUMN`.


Exercice

Que deviennent les données d'une table après un `ALTER` ?

- ☐ Elles sont conservées
- ☒ Ça dépend de la modification
- ☐ Elles sont supprimées
-  Cela dépend de la modification effectuée. Un ajout de colonne n'a aucun impact sur les données déjà existantes, par contre une suppression de colonne va nécessairement supprimer les données de celle-ci.


Exercice

Quel format de données utiliser pour une colonne de prix ?

- ☒ DECIMAL
- ☐ INT
- ☐ FLOAT
-  `DECIMAL` permet des calculs en valeur exacte pour les nombres décimaux. `INT` est un entier, tandis que `FLOAT` est conçu pour gérer des valeurs approchées.


Exercice

Quel est le rôle premier d'une clé primaire ?

- ☐ Améliorer les performances d'une contrainte de clé étrangère
Une clé primaire peut améliorer les performances d'une contrainte, mais ce n'est pas son rôle premier.
- ☒ Identifier de manière unique une ligne et trier les données d'une table
- ☐ Interdire l'accès à une table à un utilisateur non autorisé
-  La clé primaire est un identifiant unique des enregistrements d'une table. De plus, les données de la table seront triées selon cette clé, ce qui facilitera la recherche pour le SGBD.


Exercice

Une contrainte de clé étrangère permet...

- ☒ De s'assurer que les valeurs d'une colonne sont des valeurs existant dans une autre table
- ☐ De s'assurer qu'il n'est pas possible d'accéder à une table sans inclure la table référencée
- ☐ De garantir que les attributs de la colonne ne peuvent être vides
-  En présence d'une contrainte de clé étrangère, le SGBD va vérifier que les données que l'on souhaite ajouter ou modifier dans une colonne existent bien dans une autre table.


Exercice

Un index secondaire est...

- ☐ Une contrainte demandant au SGBD de vérifier les données d'une colonne depuis une autre table
- ☐ Une seconde « clé primaire », lorsqu'une seule clé ne suffit pas
- ☒ Un ensemble valeurs/référence de ligne trié selon les valeurs, permettant au SGBD de retrouver facilement les lignes correspondantes
-  Un index secondaire est un ensemble valeurs/référence de ligne, créé en plus des données de la table, au contraire de la clé primaire qui fait partie de la table elle-même.


Exercice

Quelle affirmation est correcte ?

- ☐ Un index secondaire a de lourdes contraintes en termes de performances
- ☒ Un index secondaire améliore les performances lorsqu'il est créé à bon escient
- ☐ Un index secondaire peut améliorer les performances, mais ce n'est pas son but premier
-  L'unique intérêt de créer un index secondaire est d'améliorer les performances, encore faut-il qu'il soit pertinent.

Exercice

Lorsqu'une table dispose d'un index secondaire, le SGBD va pouvoir utiliser cet index...

- ☒ Si ses critères de sélection sont compatibles avec la définition de l'index
- ☐ Tout le temps, l'index est là pour l'aider
- ☐ Seulement si les critères de sélection correspondent exactement à la définition de l'index
-  Un index secondaire peut inclure plusieurs colonnes, mais il n'est pas nécessaire qu'une opération utilise toutes ces colonnes pour utiliser l'index, à condition qu'elle utilise au moins une partie des premières colonnes.

p. 21 Solution n°9

```
1 CREATE TABLE Magasins
2 (
3     Id int NOT NULL PRIMARY KEY AUTO_INCREMENT,
4     CodePostal varchar(5) NOT NULL,
5     Nom varchar(250),
6     Adresse varchar(250)
```

```
7 )
8 ;
9 CREATE TABLE Produits
10 (
11     Id int NOT NULL PRIMARY KEY AUTO_INCREMENT,
12     Code varchar(10) NOT NULL,
13     Libelle varchar(250),
14     Description text
15 )
16 ;
17 CREATE INDEX Index_Produits ON Produits (Code)
18 ;
19 CREATE TABLE Inventaires
20 (
21     Id int NOT NULL PRIMARY KEY AUTO_INCREMENT,
22     IdMagasin int NOT NULL,
23     IdProduit int NOT NULL,
24     Quantite int,
25     FOREIGN KEY (IdMagasin)
26         REFERENCES Magasins (Id),
27     FOREIGN KEY (IdProduit)
28         REFERENCES Produits (Id)
29 )
30 ;
```