Partie 4

TECH & CODE FACTORY

Data Control language

TECH & CODE FACTORY

Visualiser les utilisateurs

Cours

La première étape pour sécuriser une base de données est de savoir quels utilisateurs y ont déjà accès. Cette information est stockée, sans surprise, dans **une base de données MySQL** appelée **mysql**.

La base de données mysql contient une table appelée **user** qui contient à son tour un certain nombre de colonnes, dont le nom de connexion de l'utilisateur et les différents privilèges et droits de connexion de l'utilisateur.

Syntaxe

```
SELECT user FROM mysql.user;
```

Créer un nouvel utilisateur

Pour ajouter un nouveau compte utilisateur, on peut utiliser la commande **CREATE USER.**

La création d'un nouveau compte utilisateur nécessite le nom de connexion de l'utilisateur et un mot de passe facultatif. Même si le mot de passe est facultatif, il est peu judicieux d'ajouter un nouveau compte sans mot de passe.

Syntaxe

```
CREATE user_name IDENTIFIED BY 'password';
```

Créer un nouvel utilisateur

Exemples

```
CREATE USER 'johnB'@'localhost' IDENTIFIED BY 'password';
```

TECH & CODE FACTORY

Renommer un utilisateur

Le nom de compte d'un utilisateur MySQL peut être modifié à l'aide de la commande **RENAME USER.**

Syntaxe

```
RENAME USER user name TO new user name;
```

Renommer un utilisateur

Exemples

```
RENAME USER 'johnB'@'localhost' TO 'johnBrown'@'localhost';
```

TECH & CODE FACTORY

Changer le mot de passe d’un utilisateur

Cours

Le mot de passe attribué à un compte utilisateur peut être modifié à l'aide de la commande **SET PASSWORD**.

Syntaxe pour l'utilisateur courant

```
SET PASSWORD = 'newpassword';
```

Syntaxe pour un autre utilisateur

```
SET PASSWORD FOR user = 'newpassword';
```

Néanmoins on préfèrera utiliser la commande **ALTER USER** pour toutes modifications reliés aux mots de passe.

Changer le mot de passe d’un utilisateur

Exemples

```
SET PASSWORD FOR 'johnBrown'@'localhost' = 'newpassword';
```

TECH & CODE FACTORY

Supprimer un utilisateur

Un compte utilisateur existant peut être supprimé à l'aide de la commande **DROP USER**.

Syntaxe

```
DROP USER user name;
```

Supprimer un utilisateur

Exemples

```
DROP USER 'johnB'@'localhost';
```

TECH & CODE FACTORY

Gestion d’utilisateurs

Exercices globaux

Créez un utilisateur nommé ‘employe’

Ajouter un mot de passe à l’utilisateur

Connectez vous à la BDD avec l’utilisateur ‘employe’

Supprimer l’utilisateur

Changer le mot de passe de votre utilisateur ‘employe’

Reconnectez vous à la BDD avec l’utilisateur ‘employe’

Voir les droits utilisateurs

Cours

Avant de modifier les privilèges d'un utilisateur, il peut être utile de voir quels privilèges sont déjà attribués. Pour ce faire, vous pouvez utiliser la commande **SHOW GRANTS** avec le nom du compte de l'utilisateur.

Syntaxe

```
SHOW GRANTS FOR user;
```

Voir les droits utilisateurs

Exemples

```
SHOW GRANTS FOR 'johnB'@'localhost';
```

TECH & CODE FACTORY

Modifier les droits utilisateurs

Cours

Un nouvel utilisateur peut se connecter au serveur MySQL mais n'a **par défaut aucun privilège** pour faire quoi que ce soit une fois connecté. La tâche suivante, après la création d'un nouveau compte utilisateur, est donc d'ajouter des privilèges au compte. Pour ce faire, on utilise l'instruction **GRANT**.

L'instruction 'USAGE ON *.*' indique que l'utilisateur n'a aucun privilège sur une base de données ou une table. En d'autres termes, l'utilisateur ne peut rien faire une fois connecté au serveur de base de données.

Syntaxe

```
GRANT SELECT on database.table TO user;
```

TECH & CODE FACTORY

Liste de droits utilisateurs

Voici une courte liste de permissions couramment utilisées :

- **ALL** - Autoriser l'accès complet à une base de données spécifique. Si une base de données n'est pas spécifiée, alors autorisez l'accès complet à l'intégralité de MySQL.
- **CREATE** - Permet à un utilisateur de créer des bases de données et des tables.
- **DELETE** - Permet à un utilisateur de supprimer des lignes d'une table.
- **DROP** - Permet à un utilisateur de supprimer des bases de données et des tables.
- **EXECUTE** - Permet à un utilisateur d'exécuter des routines stockées.
- **OPTION GRANT** - Permet à un utilisateur d'accorder ou de supprimer les privilèges d'un autre utilisateur.
- **INSERT** - Permet à un utilisateur d'insérer des lignes dans une table.
- **SELECT** - Permet à un utilisateur de sélectionner des données dans une base de données.
- **SHOW DATABASES** - Permet à un utilisateur d'afficher une liste de toutes les bases de données.
- **UPDATE** - Permet à un utilisateur de mettre à jour les lignes d'une table.

L'utilisation d'un astérisque (*) à la place de la base de données ou de la table est une option tout à fait valable, et implique que toutes les bases de données ou toutes les tables sont sélectionnés.

Source: https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html

Modifier les droits utilisateurs

Exemples

```
GRANT SELECT on MySampleDB.* TO 'johnB'@'localhost';
```

TECH & CODE FACTORY

Comment choisir les droits utilisateurs ?

Cours

Généralement le **principe du moindre privilège** prévaut. Ce principe stipule qu'un utilisateur ou un programme ne doit disposer que du minimum absolu de privilèges nécessaires pour accomplir ses tâches.

Ensuite d'autres bonnes pratiques peuvent également être cités:

- **Ne donnez jamais** à personne (sauf aux comptes root de MySQL) **l'accès à la table user** de la base de données système mysql !
- N'accordez jamais de privilèges à tous les hôtes.

Il y a également certaines choses que vous ne pouvez pas faire avec le système de privilèges MySQL :

- Vous ne pouvez pas spécifier explicitement que l'accès à un utilisateur donné doit être refusé.
- Vous ne pouvez pas spécifier qu'un utilisateur ait des privilèges pour créer ou supprimer des tables dans une base de données mais pas pour créer ou supprimer la base de données elle-même.
- Un mot de passe s'applique globalement à un compte. Vous ne pouvez pas associer un mot de passe à un objet spécifique tel qu'une base de données ou une table

Sécurités de mdp

Cours

Au cas ou quelqu'un arriverait à récupérer les informations de notre table users (au traver l'injection SQL par exemple), MySQL possède un filet de sécurité supplémentaire qui est de **hasher** nos mots de passe pour pas que ceux-ci soient écrits en texte brut.

Le hachage d'un mot de passe est un string dérivé du mot de passe en appliquant une fonction de hachage au mot de passe. L'idée essentielle ici est que ce processus ne peut pas être inversé, c'est-à-dire qu'il n'existe aucune méthode mathématique pour récupérer le mot de passe à partir du hachage.

L'algorithme de hachage utilisé sur mysql est **SHA2** (caching_sha2_password). Les fonctions de hachage MD5(), SHA1() ou Password() sont obsolètes. Nous pouvons observer les mots de passe hachés de la manière suivante:

```
SELECT user, authentication_string FROM mysql.user;

SELECT SHA2('abc', 256);

        -> '23097d223405d8228642a477bda255b32aadbce4bda0b3f7e36c9da7'
```

Dans les faits un mot de passe haché peut quand même être retrouvé en hachant tous les mots de passe possibles et en comparant les hashs entre eux. Plusieurs méthodes de protection de mdp peuvent donc être combinés pour plus de sécurité: https://www.vaadata.com/blog/how-to-securely-store-passwords-in-database/

TECH & CODE FACTORY

Gestion des permissions utilisateurs

Exercices globaux

Créez votre propre utilisateur

Donnez vous les permissions appropriés

Connectez vous à la BDD en tant que votre utilisateur

Testez le bon fonctionnement de vos permissions

Renommez l'utilisateur root

Changez le mot de passe root à un mot de passe sécurisé (mais gardez le qq part quand même vous en aurez encore besoin)

SQL Injection

Qu’est ce que l’injection SQL ?

Cours

Une attaque par injection SQL consiste en l'insertion ou "l'injection" d'une requête SQL via les données d'input du client vers l'application. Une d'injection réussie d'SQL peut lire des données sensibles de la base de données et/ou les modifier.

Regardons cette vidéo très complète qui nous explique les différents types d’injections SQL: https://www.youtube.com/watch?v=ciNHn38EyRc

Comment empêcher l'injection SQL ?

Cours

Une **instruction préparée (prepared statements)** est une requête SQL paramétrée et réutilisable qui oblige le développeur à écrire séparément la commande SQL et les données fournies par l'utilisateur.

La requête SQL est précompilée avec des espaces réservés, et les données de l'utilisateur sont ajoutées ultérieurement. Si l'utilisateur insère **`admin`** and **`a' or '1'='1,`** la logique de la requête SQL initiale ne sera pas modifiée. Au lieu de cela, la base de données recherchera un utilisateur **`admin`** dont le mot de passe est littéralement **`a' ou '1'='1`**.

```
Voici un exemple d'une approche non sécurisée en PHP :

$query = "SELECT * FROM users WHERE user = '$username' and password = '$password'" ;

$result = mysql_query($query) ;
```

```
Voici un exemple d'approche d'une instruction préparée en PHP :

$stmt = $mysqli->prepare("SELECT * FROM users WHERE user = ? AND password = ?") ;

$stmt->bind_param("ss", $username, $password) ;

$stmt->execute() ;
```

Pour aller plus loin: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

TECH & CODE FACTORY

Injection SQL

Exercices

Suivez les instructions du site suivant pour vous entraîner au queries SQL dans un environnement protégé:
https://rmauro.dev/practice-of-sql-injection/

TECH & CODE FACTORY

Bonnes pratiques générales

Bonnes pratiques générales

Cours

Nous avons pu aborder certains points de sécurité et les bonnes pratiques les concernant (la gestion des utilisateurs et leur mots de passe, se protéger des injections SQL) mais nous n'avons pas pu couvrir tous les sujets de sécurité. Si cela vous intéresse je vous invite à vous référer à la documentation de MySQL pour une liste très exhaustive des points de liées à la sécurité: https://dev.mysql.com/doc/mysql-security-excerpt/8.0/en/security.html ou bien vers des articles de blog plus succints: https://www.upguard.com/blog/top-11-ways-to-improve-mysql-security

TECH & CODE FACTORY

Les bases de données en tant que système distribué

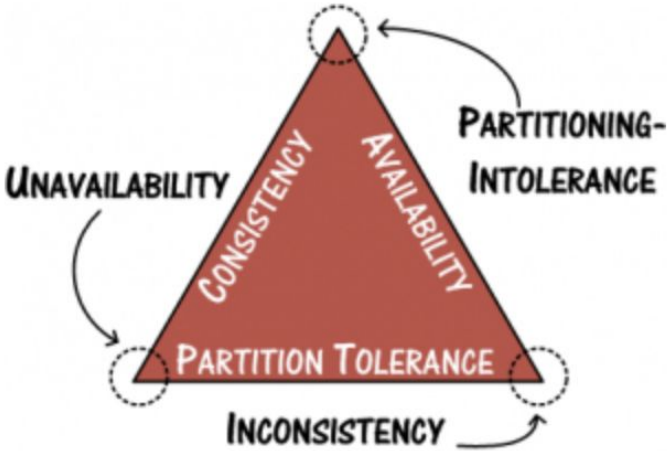
TECH & CODE FACTORY

Le théorème CAP

Cours

Jusqu'ici nous avons étudié les BDD comme étant plutôt un seul système centralisé, or il est intéressant de se demander qu'est ce qu'impliquerait une BDD en tant que système distribué.

Pour cela comprendre le théorème CAP peut être utile. Celui ci nous dit que nous pouvons avoir que 2 des 3 caractéristiques suivantes: cohérence, disponibilité, tolérance aux partitions (*en anglais: Consistency, Availability, Partition tolerance*).



Je vous invite à regarder la vidéo suivante qui vous explique bien le théorème CAP :
https://www.youtube.com/watch?v=k-Yaq8AHlFA

De manière général nous avons donc:

- Les BDD relationnelles qui se concentrent généralement sur la cohérence et la disponibilité. Elles répondent au propriétés ACID
- Les BDD non relationnelles qui se concentrent généralement sur la tolérance aux partitions et la disponibilité et ne se focalisent pas sur la cohérence. Elles répondent aux propriétés BASE

Les propriétés ACID

Cours

Les propriétés ACID correspondent aux caractéristiques suivantes:

Atomique - Chaque transaction est soit correctement exécutée, soit le processus s'arrête et la base de données revient à l'état antérieur au début de la transaction. Cela garantit que toutes les données de la base de données sont valides.

Cohérent - Une transaction traitée ne mettra jamais en danger l'intégrité structurelle de la base de données.

Isolation - Les transactions ne peuvent pas compromettre l'intégrité des autres transactions en interagissant avec elles alors qu'elles sont toujours en cours.

Durabilité - Les données liées à la transaction terminée persisteront même en cas de panne de réseau ou d'électricité. Si une transaction échoue, cela n'aura pas d'impact sur les données manipulées.

Je vous invite à regarder l'exemple suivant qui décrit bien le concept de transaction et les propriétés ACID sur une BDD SQL: https://www.youtube.com/watch?v=AcqtAEzuoj0

Les propriétés BASE

Les propriétés BASE correspondent aux caractéristiques suivantes:

Basiquement disponible - Plutôt que d'imposer une cohérence immédiate, les BDD. assureront la disponibilité des données en les répartissant et en les répliquant sur les nœuds du cluster de base de données.

Soft State - En raison de l'absence de cohérence immédiate, les valeurs des données peuvent changer au fil du temps. Le modèle BASE rompt avec le concept d'une base de données qui assure sa propre cohérence, déléguant cette responsabilité aux développeurs.

Cohérence éventuelle - Le fait que BASE n'impose pas une cohérence immédiate ne signifie pas qu'elle ne l'atteindra jamais. Cependant, jusqu'à ce qu'elle y parvienne, la lecture des données est toujours possible (même si elle ne reflète pas la réalité).

Conclusion

TECH & CODE FACTORY

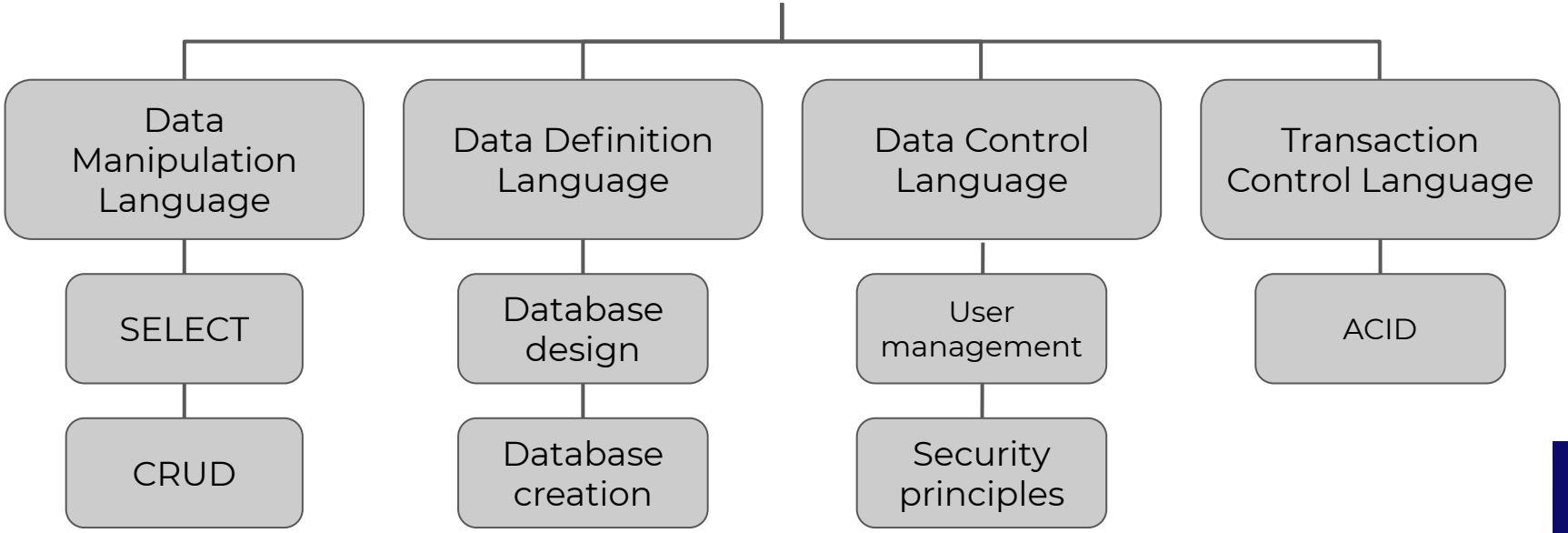
Sujets vus en classe

Cours

NoSQL

vs

SQL



TECH & CODE FACTORY

Pour aller plus loin

Cours

Mooc open classrooms:
https://openclassrooms.com/fr/courses/1959476-administrez-vos-bases-de-donnees-avec-mysql

Livres:

- “SQL in 10 Minutes, Sams Teach Yourself” - Ben Forta, ISBN: 978-0672336072
- “Beginning Database Design Solutions” - Rod Stephens, ISBN: 978-0470385494

TECH & CODE FACTORY