ISCC – JOUR 11

SQL – Jour 2&3

Le MySQL

- Un système de base de données (BDD)
- Son but est de **stocker** des données et d'interagir avec
- Pour accéder à ces données, nous utiliserons les requêtes SQL

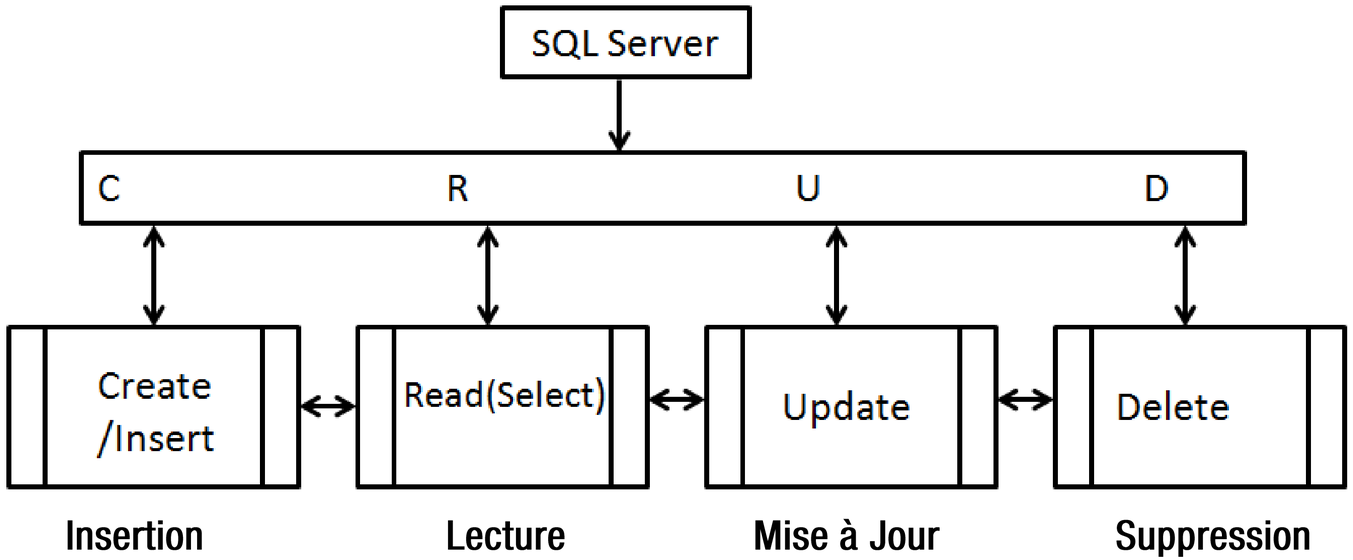
CRUD

En tant que développeur il vous faudra apprendre à :

- Insérer des enregistrements en base de données
- Sélectionner des enregistrements en base de données
- Mettre à jour des enregistrements en base de données
- Supprimer des enregistrements en base de données

Dans un projet web, ces quatre actions servent à réaliser le CRUD (create, read, update, delete) d'un modèle de donnée. Le CRUD user revient souvent dans les projets web par exemple. On parle alors d'un CRUD user.

CRUD


SQL Server
C
R
U
D
Create
/Insert
Read(Select)
Update
Delete
Insertion
Lecture
Mise à Jour
Suppression

SELECT - FROM

Permet de **sélectionner des données** dans une **table précise**

```
SELECT champ1, champ2,
FROM MyTable;
```

Syntaxe

```
SELECT username
FROM users;
```

Exemple

SELECT - WHERE

Permet de sélectionner des données avec une condition particulière

```
SELECT champ1, champ2,
FROM MyTable
WHERE condition;
```

Syntaxe

```
SELECT email
FROM users
WHERE username = "gecko";
```

Exemple

INSERT INTO

Permet d'ajouter des données dans une table définie

```
INSERT INTO table
VALUES('valeur1', 'valeur2');
```

Syntaxe

```
INSERT INTO users
VALUES('gecko',
'gecko@42.eu');
```

Exemple

UPDATE - WHERE

Permet de mettre à jour des données suivant une condition

```
UPDATE table
SET champ1 = 'valeur1',
champ2 = 'valeur2'
WHERE condition;
```

Syntaxe

```
UPDATE users
SET email = 'marvin@toto.eu',
username = 'toto42'
WHERE id = 42;
```

Exemple

DELETE - WHERE

Permet de supprimer des données en fonction d’une condition

```
DELETE FROM table
WHERE condition;
```

Syntaxe

```
DELETE FROM users
WHERE username = 'marvin';
```

Exemple

Persistance des données

Une donnée est dite persistante si elle est sauvegardée après la fin de l'exécution d'un programme et peut être récupérées en cas d'arrêt soudain de ce dernier (à cause d'un crash par exemple).

Quand un internaute ferme sa page web ou que le serveur de votre site redémarre, les données persistantes seront celles que vous stockez dans le cache ou dans votre BDD. Il faut donc bien faire attention à enregistrer ces données !

PHP Data Object (PDO)

- La PDO permet aux développeurs de manipuler une base de données (BDD) comme s'il s'agissait d'un Objet PHP.

- Cela offre une couche supplémentaire de code entre l'utilisateur de votre site internet et votre BDD,
  il est ainsi plus facile de contrôler et d'utiliser ces données pour développer vos pages web.

MySQL dans PHP

Rappel - Connexion à la base de données

```php
function connect_to_database(){
    $servername = "localhost";
    $username = "root";
    $password = "root";
    $databasename = "madatabase";

    try {
        $pdo = new PDO("mysql:host=$servername;dbname=$databasename", $username, $password);
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        echo "Connected successfully";
        //on est connectee
        return $pdo;
    } catch (PDOException $e) {
        echo "Connection failed: ". $e->getMessage();
    }
}
```

MySQL dans PHP

Rappel- Récupérer des données

```php
$pdo = connect_to_database();
//apres la connection on peut effectuer des enregistrement
//requete qui recupere un seul enregistrement
$query = $pdo->query("SELECT * FROM masupertable");
$user = $query->fetch();
var_dump($user);


//requete qui recupere plusieurs enregistrements
$users = $pdo->query("SELECT * FROM masupertable")->fetchAll();
// Ensuite on peut afficher
var_dump($users);
foreach ($users as $user) {
    echo $user['Nom']."<br/>";
}
```

PHP Data Object (PDO)

- A la ligne 1, on prépare la requête: on l'écrit.
- A la ligne 2 et 4, on exécute la requête avec des paramètres spécifiques.
- A la ligne 3 et 5, on récupère tout les résultats correspondant à la requête SQL.
- On peut donc exécuter la même requête plusieurs fois avec des paramètres différents au moment de l'exécution.
- La ligne 2 et 3 ainsi que la ligne 4 et 5 donneront des résultats différents grâce aux paramètres qui sont passés

```php
<?php
/* Exécute une requête préparée en passant un table
au de valeurs */
1 $sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < ? AND couleur = ?');

2 $sth->execute(array(150, 'rouge'));
3 $red = $sth->fetchAll();

4 $sth->execute(array(175, 'jaune'));
5 $yellow = $sth->fetchAll();
```

Prepare

- On ne connait pas à l'avance les paramètres que l'utilisateur va entrer pour sa requête, on va donc la préparer.
- Pour les éléments que nous ne connaissons pas encore, comme les calories nous allons mettre des paramètres.
- Ces paramètres peuvent être soit:
  - Nommés (Exemple du haut)
  - Marqués (Exemple du bas)
- Ces paramètres seront remplacés au moment de l'exécution de la requête.

```php
<?php
$sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < :calories AND couleur = :couleur');
```

Nommés (:calories)

```php
<?php
$sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < ? AND couleur = ?');
```

Marqués (?)

Prepare permet d'augmenter la sécurité et donc d'éviter les intrusions en base de données

https://www.php.net/manual/fr/pdo.prepare.php

Execute

- Au moment du **execute** la requête préparée va s'exécuter.

- Si nous ne passons pas de paramètre au **execute** la requête SQL exécutée sera celle qui a été préparée.

- Si des paramètres sont passés au **execute**, Ils remplacerons ceux du **prepare**.

- Seul un array (tableau) peut être passé en paramètre d'un **execute**.

- Dans le cas de paramètres:
  - **Nommés**, ce sera un **tableau associatif**
  - **Marqués**, ce sera un **tableau** contenant uniquement les valeurs

```php
<?php
$sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < :calories AND couleur = :couleur');
$sth->execute(
array(':calories' => 150, ':couleur' => 'rouge')
);
```

Nommés (:calories)

```php
<?php
$sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < ? AND couleur = ?');
$sth->execute(array(150, 'rouge'));
```

Marqués (?)

https://www.php.net/manual/fr/pdostatement.execute.php

Data Mapping

Le Data Mapping permet de faire communiquer deux modèles de données entre eux, ainsi chaque modèle traite des même données mais d'une manière différente.

Grâce au Data Mapping, il est possible lier les données d'un tableau PHP avec avec une table SQL afin de les insérer en Base de Données.

Le saviez-vous ?
En Québécois, on traduit Data Mapping par mise en correspondance des données.

BindParam

- Au lieu d'ajouter un tableau de paramètre dans la fonction **execute(),** ici on décide d'utiliser **bindParam().**

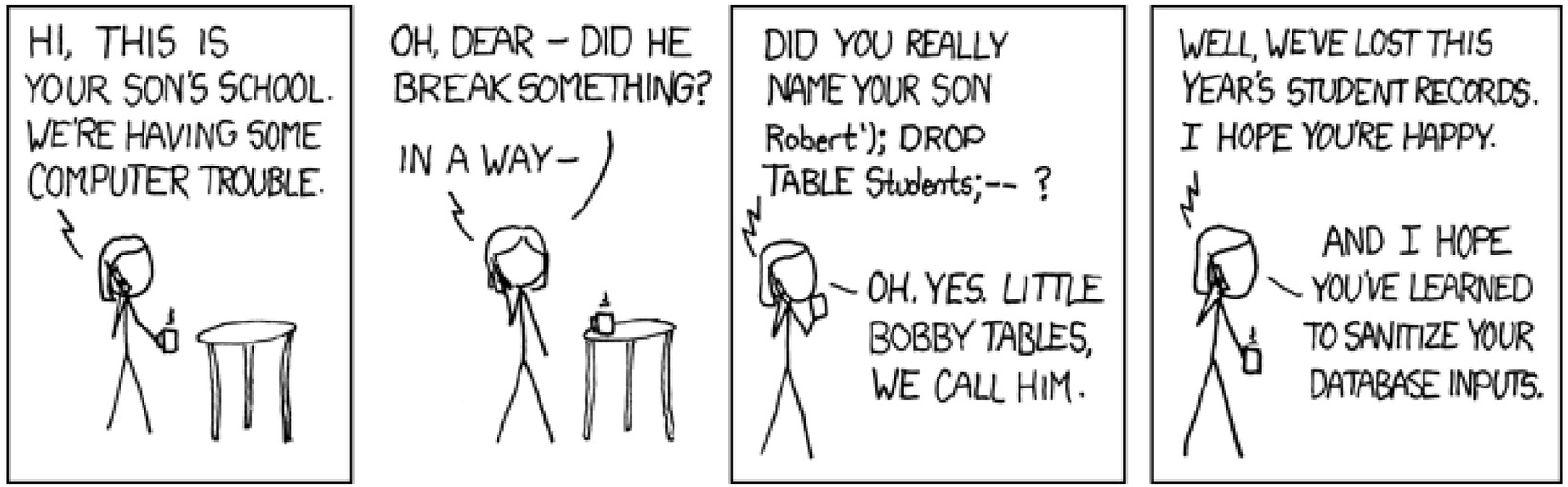
- Il s'agit du même procédé :
  Les paramètres SQL **:couleur** et **:calories** vont être remplacés par leur valeur indiquée dans **bindParam()**.

```php
<?php
$sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < :calories
    AND couleur = :couleur');

$sth->bindParam(':calories', 150);
$sth->bindParam(':couleur', 'rouge');

$sth->execute();
```

https://www.php.net/manual/fr/pdostatement.bindparam.php

« Un bon programmeur c’est quelqu’un qui regarde des deux côtés avant de traverser une route à sens unique.»

Doug Linder


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.