



Bachelier en Informatique de Gestion

Projet de Développement Web

Enseignement supérieur économique de type court

Code FWB : 7534 30 U32 D1

Code ISFCE : 4IPDW



= BUROTIX ()

Table des matières

Généralités

- 01. Introduction au web
- 03. Outils
- 05. Frameworks

Côté Client

- 12. Structure HTML
- 13. Formulaire HTML
- 14. Mise en forme CSS
- 15. Adaptabilité
- 17. Javascript
- 18. Framework jQuery
- 19. AJAX

Côté Serveur

- 21. Middleware PHP
- 22. Traitement du formulaire
- 23. Architecture MVC
- 24. Base de données SQL
- 25. Données XML
- 26. Données JSON





24. Base de données SQL

MySql

Créer une DB

Procédure SELECT

PhpMyAdmin

Fonctions PHP/MySQL

Importer une DB

Procédure INSERT



= BUROTIX ()



Configuration

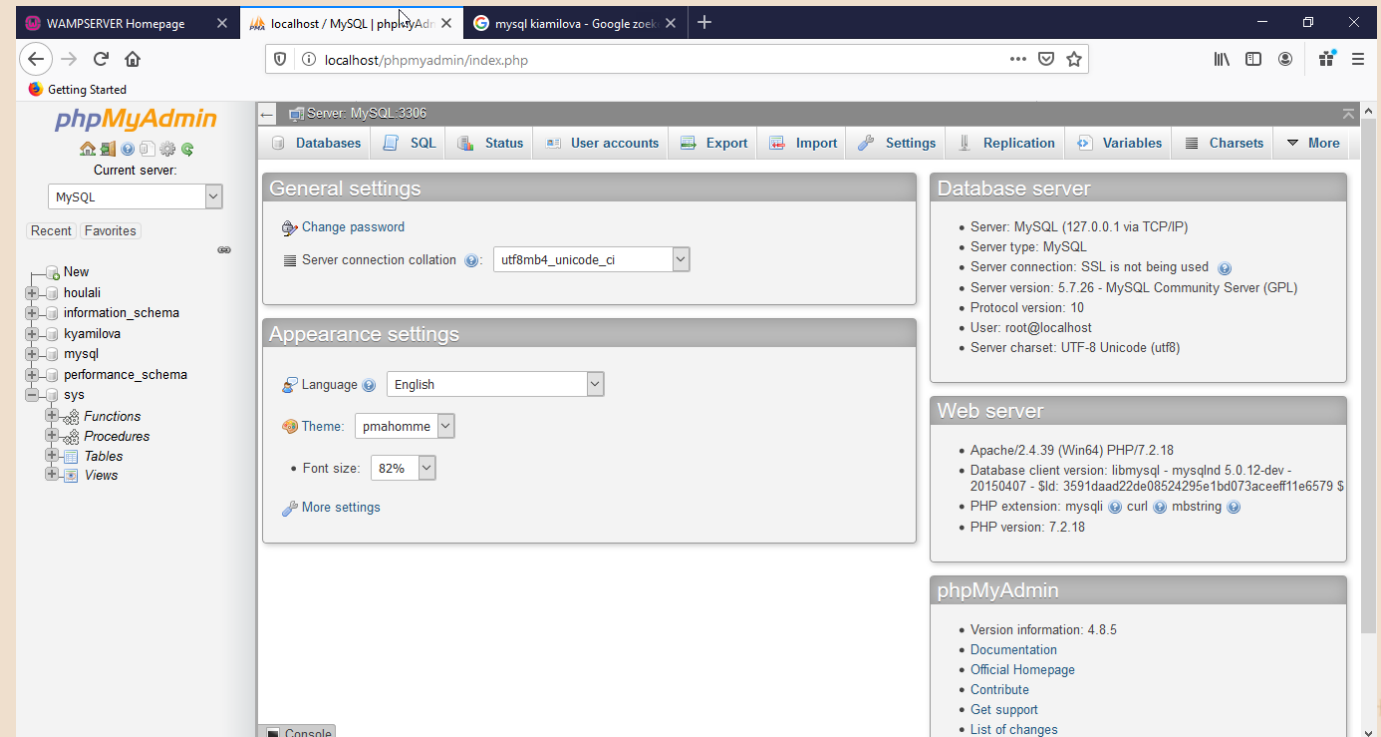
Configuration MySQL Engine

- Hypothèse
 - Le serveur MySQL est installé avec WAMP
- Configuration à modifier
 - *default storage engine* : **InnoDB** (et non MyISAM)
 - **C:\wamp64\bin\mysql\mysql5.7.26\my.ini**
 - Vers line 58 : **default-storage-engine=INNODB**
 - Explication technique (en Anglais)
 - InnoDB supporte les relations entre tables (*foreign key*)
- Redémarrer le service MySQL



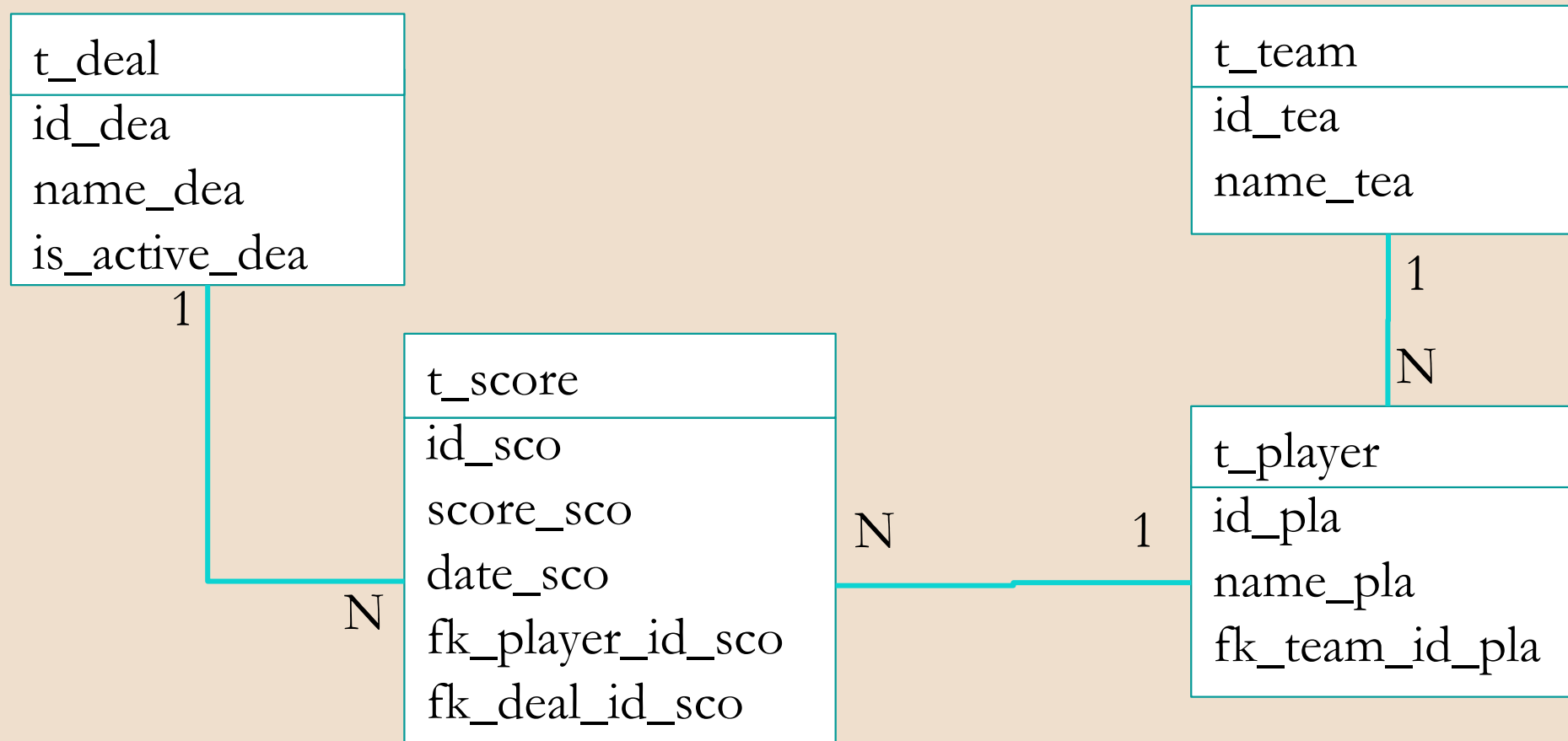
Accéder à l'interface d'administration MySQL

- Se logger dans PhpMyAdmin
 - `http://localhost/phpmyadmin/`
 - Username : *root*
 - Password : *(empty)*



Exo 01 : importation de base de données

fichier: `exo01_import_sample.sql`



= BUROTIX ()

Importer un script SQL dans une base de données MySQL

- Create database
 - Name: **sample**
 - Encodage : **latin1_general_ci**
- MyPhpAdmin / menu : **Importer**
 - Fichier à importer : **exo01_import_sample.sql**
 - Source: *www.burotix.be/.../internet/24_mysql*
 - Format: **SQL**
 - **Exécuter**
- Vérifier
 - Default data engine : **InnoDB**
 - L'existence de quatre tables et de leurs champs
 - Le contenu de ces tables (toutes remplies)
 - L'existence de trois vues
 - Les relations entre les tables => menu : **Concepteur**

Importer un fichier CSV dans une base de données MySQL

- Dans une base nouvelle ou existante
- MyPhpAdmin / menu : **Importer**
 - Fichier à importer : **exo01_catalogue.csv**
 - Format: **CSV** ... vérifier les options : séparateur etc.
 - **Exécuter**
- Vérifier
 - Default data engine : **InnoDB**
 - L'existence de la table, de ses champs et son contenu
- Modifier
 - Modifier le nom de la table => menu : **Opérations**
 - Modifier le nom et les propriétés des champs => menu : **Structure**
 - Ajouter la clé primaire => menu : **SQL**
 - **ALTER TABLE tbl ADD `id_cat`
INT PRIMARY KEY AUTO_INCREMENT FIRST;**
 - C'est la manière la plus efficace

Exporter une base de données MySQL

- Dans une base existante
- MyPhpAdmin / menu : **Exporter**
 - Méthode d'exportation: *rapide*
 - Format : *SQL*
 - Option : *include create base*
 - **Exécuter**
 - Un fichier SQL est créé et téléchargé.





MySQL et PHP

Approche objet générique : PDO

Approche objet spécifique : class mysqli

Approche procédurale : mysqli_***



= BUROTIX ()

Lier PHP et une base de données

- PHP contient des extensions (bibliothèque de fonctions) permettant l'accès à une base de données.
 - MySQL Interface (MySQLi)
 - Spécifique à MySQL
 - Rapide
 - Implantation POO ou implantation procédurale
 - PHP Data Object (PDO)
 - Code générique et portable, "couche d'abstraction"
 - Connexion à un grand nombre de bases de données
 - Implantation POO



MySQLi : fonctions PHP

- DML only
- connexion
 - `mysqli_connect`
 - `mysqli_close`
- requête
 - `mysqli_query`
 - `mysqli_prepare`
 - et amis
- requête "SELECT"
 - `mysqli_fetch_array`
 - `mysqli_fetch_all`
 - `mysqli_free_result`
 - `mysqli_num_rows`
- requête "INSERT" etc.
 - `mysqli_affected_rows`
 - `mysqli_commit`



MySQLi : objets et méthodes PHP

- DML only
- connexion
 - `new mysqli(...)`
 - `mysqli::close`
- requête
 - `mysqli::query`
 - `mysqli::prepare`
 - `stmt::bind_param`
 - `stmt::execute`
 - `stmt::fetch`
 - et cousins
- requête "SELECT"
 - `result::fetch_array`
 - `result::fetch_all`
 - `result::free`
 - et cousins
 - `result->num_rows`
- requête "INSERT" etc.
 - `mysqli->affected_rows`
 - `mysqli::commit`

PDO : objets et méthodes PHP

- DML only
- connexion
 - `new PDO(...)`
 - `PDO::close` ~~⚡~~
- requête
 - `PDO::query`
 - `PDO::prepare`
 - `PDOStatement::bindParam`
 - `PDOStatement::bindValue`
 - `PDOStatement::execute`
- requête "SELECT"
 - `PDOStatement::fetchAll`
 - et cousins
 - `result->num_rows` ~~⚡~~
 - use `SELECT COUNT(*)`
 - `result::free` ~~⚡~~
- requête "INSERT" etc.
 - `PDOStatement::rowCount`
 - `PDO::commit`



Références

- <https://phpdelusions.net/pdo>
 - *(the only proper) PDO tutorial*
- <https://www.php.net/manual/en/book.pdo.php>
 - *site officiel*





Ouvrir la connexion

new mysqli

mysqli_connect

new PDO



= BUROTIX ()

MySQLi : Connexion à la base de données

- Établir la connexion entre l'application PHP et la base de données SQL.

```
$conn = mysqli_connect(  
    "localhost", "root", "", "sample_4ipdw" );
```

- Paramètres

- Serveur : "localhost"
- Utilisateur : "root"
- Mot de passe : ""
- Base de données : "sample_4ipdw"

- Retour :

- **\$conn** : objet de connexion à la base de données
 - utilisé comme paramètre des fonctions **mysqli_XXX**
- **false** si erreur



PDO : Connexion à la base de données

- Établir la connexion entre l'application PHP et la base de données SQL.

```
$pdo = new PDO($dsn, "root", "", $options);
```

- Paramètres

- Serveur : "localhost"
- Utilisateur : "root"
- Mot de passe : ""
- Base de données : "4ipdw_sample"
- Charset : 'utf8mb4'
- DSN

- string, paires `param=valeur`, délimitées par ";", ni espace, ni apostrophe

```
$dsn = "mysql:host=localhost;dbname=4ipdw_sample;port=3306;charset=utf8mb4";
```

- Retour :

- `$pdo` : objet gérant la connexion à la base de données
 - ses méthodes seront utilisées pour les requêtes, etc.



Connexion à la base de données

- Caractères spéciaux
 - Pour assurer leur affichage correct, il faut imposer à la DB le format UTF-8 :
 - requête SQL : "SET NAMES UTF8"
 - Nécessaire avec PDO ?
- MySQL ou MariaDB ?
 - Dans l'environnement WAMP !
 - Pour établir la connexion avec la base de données MySQL, port 3306 (défaut)

```
$conn = mysqli_connect(
    "localhost", "root", "", "sample_4ipdw", 3306 );
```
 - Pour établir la connexion avec la base de données MariaDB, port 3307

```
$conn = mysqli_connect(
    "localhost", "root", "", "sample_4ipdw", 3307 );
```





Consulter des données SELECT

Requêtes avec attente d'un
résultat

`mysqli_query`

`mysqli_num_rows`

`mysqli_fetch_all`

`PDO::query`

`PDO::prepare`

`PDOStatement::fetchAll`



= BUROTIX ()

MySQLi : SELECT : requête simple

- Pour envoyer la requête SQL à la base de données

```
$result = mysqli_query(  
    $conn, "SELECT * FROM City");
```

- Paramètres

- `$conn`

- String SQL : `"SELECT Name FROM City"`

- Retour

- MySQL result set : `$result`



MySQLi : SELECT : le résultat, ligne par ligne

- Pour capturer le résultat de la requête ligne par ligne

```
while ($row = mysqli_fetch_array($result))  
{  
    echo $row['joueur'].':'. $row['partie'];  
}
```

- Paramètres

- `$result`

- Retour

- Une seule ligne du résultat : `$row`
(tableau associatif)
- Donc boucle `while` sur
`mysqli_fetch_array`
nécessaire pour lire tout le résultat

une ligne, affichée avec `print_r()` :

```
$row = Array  
(  
    [joueur] => Allen,  
    [partie] => ISFCE,  
)
```

MySQLi : SELECT : tout le résultat d'un coup

- Pour capturer le résultat de la requête en une seule instruction

```
$data = mysqli_fetch_all($result, MYSQLI_ASSOC);
```

- Paramètres

- `$result`

- Option générant un tableau associatif : `MYSQLI_ASSOC`

- Retour

- Tout le résultat : `$data`
(tableau indexé de tableau associatif)
 - Donc boucle `foreach` sur `$data`
nécessaire pour lire tout le résultat

tout le résultat, affiché avec `print_r()` :

```
$data = Array
(
    [0] => Array (
        [joueur] => Alex,
        [partie] => entre amis
    ),
    [1] => Array (
        [joueur] => Allen,
        [partie] => entre copains
    )
);
```


Requêtes préparées : motivation

- Éviter l'injection SQL
 - requête avec variable
- Optimiser les ressources en cas de requêtes multiples



Requêtes préparées : injection SQL

- Regardez ce code PHP :

```
$myname = $_POST['myname'];  
$sql_s = "SELECT * FROM myTable WHERE name='$myname' ";  
mysqli_query($sql_s);
```
- Un hacker peut écrire dans le formulaire web, dans l'input de nom **myname** :
' OR '1'='1'
- La requête devient
SELECT * FROM myTable WHERE name=' ' OR '1'='1'
- Résultat ?
 - Toute la table est retournée ! Sans commentaire ... ☹
- Autres exemple similaires et solution
 - <https://websitebeaver.com/prepared-statements-in-php-mysqli-to-prevent-sql-injection>
 - https://phpdelusions.net/sql_injection



Requêtes préparées : principe

1. La requête SQL est **préparée** avec des valeurs vides
 - ces valeurs vides correspondent aux variables
2. Les valeurs vides sont liées à une valeur et à un type.
3. La requête est exécutée.
4. Les résultats sont traités.
5. Il est possible de recommencer 2., 3. et 4. avec d'autres valeurs.



MySQLi : Requêtes préparées

1. La requête SQL est préparée avec des valeurs vides

```
$stmt = mysqli_prepare(  
    "SELECT * FROM myTable  
    WHERE name = ? AND age = ? ");
```

2. Ces valeurs vides sont liées à une valeur et à un type.

```
$stmt->bind_param( "si",  
    $_POST['name'], $_POST['age'] );
```

3. Cette requête est exécutée.

```
$stmt->execute();
```

4. Les résultats sont traités.

```
// ... traitement du résultat ...  
$stmt->close();
```



PDO : Requêtes préparées

1. La requête SQL est préparée avec des valeurs vides

- Syntaxe : `":email"` ou `":status"`

```
$sql = '    SELECT * FROM users
        WHERE email = :email
        AND status = :status    ';
```

```
$stmt = $pdo->prepare($sql);
```

2. Ces valeurs vides sont liées à une valeur [et à un type] (assoc array).

```
$param = ['email' => $email, 'status' => $status];
```

3. Cette requête est exécutée.

```
$stmt->execute($param);
```

4. Les résultats sont traités.

```
$user = $stmt->fetch();
```



SELECT : nombre de lignes retournées

MySQLi

- Pour déterminer le nombre de lignes du résultat
`$n = mysqli_num_rows($result);`
- Paramètres
 - `$result`
- Retour
 - Nombre de lignes du résultat : `$n`

PDO

- Pas de fonction équivalente
- Utiliser une requête spécifique
 - `SELECT COUNT(*) FROM ...`



SELECT : libérer la mémoire

MySQL

- Pour libérer la mémoire associée à un résultat
`mysqli_free_result($result);`
- Paramètres
 - `$result`
- Retour:
 - -

PDO

- Pas de fonction équivalente
- Effacer l'objet libère la mémoire
`$result = null;`
- A vérifier dans les faits ?



Modifier des données

INSERT

UPDATE

DELETE

Requêtes sans attente d'un résultat

`mysqli_query`

`mysqli_affected_rows`

`mysqli_commit`

`new PDO`

`new PDO Statement`



= BUROTIX ()

MySQLi : INSERT : requête simple

- Requêtes "non préparées"
- Pour envoyer la requête SQL à la base de données

```
$b = mysqli_query($conn,  
    "INSERT INTO t_deal (name_dea)  
    VALUES ('partie avec Valérie');"
```
- Paramètres
 - `$conn`
 - String SQL : `"INSERT INTO t_deal (name_dea)
VALUES ('partie avec Valérie')"`
- Retour
 - Booléen, selon réussite ou échec : `$b`



MySQLi : INSERT : requête préparée

1. La requête SQL est préparée avec des valeurs vides

```
$stmt = mysqli_prepare( $conn,  
    "INSERT INTO myTable (name, age)  
    VALUES (?, ?) ");
```
2. Ces valeurs vides sont liées à une valeur et à un type.

```
$stmt->bind_param( "si",  
    $_POST['name'], $_POST['age'] );
```
3. Cette requête est exécutée.

```
$stmt->execute();  
$stmt->close();
```



PDO : INSERT : requête préparée

1. La requête SQL est préparée avec des valeurs vides

```
$sql = "  
    INSERT INTO myTable (name, age)  
    VALUES (:name, :age)           ";  
$stmt = $pdo->prepare($sql);
```

2. Ces valeurs vides sont liées à une valeur [et à un type] (assoc array).

```
$param = [ 'name' => $_POST['name'],  
           'age'   => $_POST['age'] ];
```

3. Cette requête est exécutée.

```
$stmt->execute($param);
```



MySQLi : INSERT : nombre lignes affectées

- Pour déterminer le nombre de lignes affectées par la dernière opération
 - pour vérifier si la requête s'est bien déroulée
`$n = mysqli_affected_rows($conn);`
- Paramètres
 - `$conn`
- Retour
 - Nombre de lignes affectées : `$n` (entier positif)
 - Si erreur : `-1`



PDO : INSERT : nombre lignes affectées

- Pour déterminer le nombre de lignes affectées par la dernière opération
 - pour vérifier si la requête s'est bien déroulée
`$n = $stmt->rowCount();`
- Paramètres
 - -
- Retour
 - Nombre de lignes affectées : `$n` (entier positif)
 - Si erreur : `E_WARNING` ou `PDOException`



MySQLi : INSERT : exécuter la requête

- Pour faire exécuter définitivement la requête SQL par la base de données

```
$b = mysqli_commit($conn);
```

- Paramètres

- `$conn`

- Retour

- Booléen, selon réussite ou échec : `$b`



PDO : INSERT : exécuter la requête

- Pour faire exécuter définitivement la requête SQL par la base de données

```
$b = $pdo->commit();
```

- Paramètres

- _

- Retour

- Booléen, selon réussite ou échec : `$b`

- Amis

- `$pdo->rollBack();`





Fermer la connexion

`mysqli_close`



= BUROTIX ()

Déconnexion de la base de données

MySQL

- Pour libérer les ressources, en fermant la connexion avec la base de données
`$b = mysqli_close($conn);`
- Paramètres
 - `$conn`
- Retour
 - Booléen, selon réussite ou échec : `$b`

PDO

- Pas de fonction équivalente
- Effacer l'objet libère la mémoire
`$pdo = null;`
- A vérifier dans les faits ?





Exos

exo02 : sur base de mysqli

exo03 : sur base du PDO

exo 12-14 : sur base de MVC (obsolète, TBC)



= BUROTIX ()

Exo 02 : MySQLi en pratique

- Fichier d'exemple
 - `exo02_mysql.php`
 - Téléchargez, installez, faites tourner,
 - Disséquez le code
- Cf extraits du code sur les slides suivants

The screenshot shows a web browser window with the address bar displaying `4ipdw/11_exemple_mysql/example_dml`. The page title is "Getting Started". The main content area is titled "Insert and Select Data In Database Using PHP." and is enclosed in a dashed border. It contains two sections: "Insérer une nouvelle partie (form)" and "Parties en cours".

Insérer une nouvelle partie (form)

*INSERT INTO t_deal (name_dea)
VALUES (...)*

Deal Name:

Insert New Deal

Parties en cours

*SELECT * FROM t_deal*

entre amis
entre collègues
ISFCE

Exo 02 : connexion

- Toujours en début de code
 - `$conn = mysqli_connect("localhost", "root", "", "4ipdw_sample");`
- Toujours en fin de code
 - `mysqli_close($conn);`



Exo 02 : INSERT

- Composer la requête en langage SQL

```
$q = "INSERT INTO t_deal (name_dea)  
VALUES ('$deal');";
```
- Exécuter la requête d'insertion

```
$result = mysqli_query($conn, $q );
```
- Compter le nombre de lignes affectées

```
$n = mysqli_affected_rows($conn);
```
- Confirmer l'insertion

```
$b = mysqli_commit($conn);
```



Exo 02 : SELECT (1/3)

- Composer la requête en langage SQL

```
$q = "SELECT name_dea AS partie  
      FROM t_deal  
      WHERE is_active_dea IS TRUE;";
```

- Exécuter la requête de sélection

```
$result = mysqli_query($conn, $q );
```

./..



Exo 02 : SELECT (2/3)

- Soit : Traiter le résultat ligne par ligne

```
while ($row = mysqli_fetch_array($result))  
{  
    echo "<div>{$row['partie']}</div>";  
}
```

./..



Exo 02 : SELECT (3/3)

- Soit : Traiter le résultat en un bloc

```
$deal_a = mysqli_fetch_all($result, MYSQLI_ASSOC);  
foreach( $deal_a as $row )  
{  
    echo "<div>{$row['partie']}</div>";  
}
```

- Libérer la mémoire

```
mysqli_free_result($result);
```



Exo 03 : PDO en pratique

- Fichier d'exemple
 - `exo03_pdo.php`
 - Téléchargez, installez, faites tourner,
 - Disséquez le code
- Cf extraits du code sur les slides suivants

The screenshot shows a web browser window with the address bar displaying "4ipdw/11_exemple_mysql/example_dml...". The page title is "Getting Started". The main content is titled "Insert and Select Data In Database Using PHP." and is enclosed in a dashed border. It contains two sections: "Insérer une nouvelle partie (form)" and "Parties en cours".

Insérer une nouvelle partie (form)

*INSERT INTO t_deal (name_dea)
VALUES (...)*

Deal Name:

Insert New Deal

Parties en cours

*SELECT * FROM t_deal*

entre amis
entre collègues
ISFCE

Exo 03 : connexion

- Toujours en début de code

```
$dsn = "mysql:host=localhost;dbname=sample;";  
$pdo = new PDO( $dsn, "root", "" );
```

- Toujours en fin de code

-



= BUROTIX ()

Exo 03 : INSERT

- Composer la requête en langage SQL

```
$q = "INSERT INTO t_deal (name_dea) VALUES (?);";
```

- Exécuter la requête d'insertion

```
$pdo->beginTransaction();  
$stmt = $pdo->prepare($q);  
$stmt->execute([$deal]);
```

- Compter le nombre de lignes affectées

```
$n = $stmt->rowCount();
```

- Confirmer l'insertion

```
$pdo->commit();
```



Exo 03 : SELECT (1/3)

- Composer la requête en langage SQL

```
$q = "SELECT name_dea AS partie  
      FROM t_deal  
      WHERE is_active_dea IS TRUE;";
```

- Exécuter la requête de sélection

```
$stmt = $pdo->query($q);
```

./..



Exo 03 : SELECT (2/3)

- Soit : Traiter le résultat ligne par ligne

```
while ($row = $stmt->fetch())  
{  
    echo "<div>{$row['partie']}</div>";  
}
```

./..



Exo 03 : SELECT (3/3)

- Soit : Traiter le résultat en un bloc

```
$deal_a = $stmt->fetchAll(FETCH_ASSOC);  
foreach( $deal_a as $row )  
{  
    echo "<div>{$row['partie']}</div>";  
}
```

- Libérer la mémoire

-

- *Remarque : non pleinement testé*



Exo 04 : variante

- Réalisez une application web permettant
 - d'introduire un nouveau **score** dans la base, à partir d'un formulaire.
 - d'afficher les scores courants
 - architecture similaire à l'exo 24-02.



Exo 12 : MVC

- Convertissez cette application en MVC.
- Conseils
 - Ne réinventez pas la roue ! Récupérez vos développements précédents (html_helper, db_helper, etc.). Enrichissez vos bibliothèques par de nouvelles fonctions.
 - INSERT : **Controller** envoie les données du form à **Model** qui gère la connexion avec la base.
 - SELECT : **Controller** demande à **View** d'afficher la liste des parties. **View** demande alors à **Model** les données (associative array) et les affiche.



Exo 14 : search engine

- Sur base de l'exo 12, ajoutez un **outil de recherche**.
 - Input field
 - Requête SQL adaptée
- Pas question d'écrire un code nouveau ou un algorithme comme du temps du CSV. Il suffit de modifier quelques lignes de l'application.
- L'architecture MVC est un atout pour ce genre de travail.



Exo 14 : search engine

Entrez vos mots-clés:

Chercher

Afficher l'ensemble des parties

```
if( ! empty($search_word))
{
    // il y a des mots-clés de recherche
    $q = <<< SQL
    SELECT name_dea AS partie
    FROM t_deal
    WHERE name_dea LIKE "%$search_word%"

    SQL;
}
else
{
    // il n'y a pas de mot-clé de recherche
    $q = <<< SQL
    SELECT name_dea AS partie
    FROM t_deal

    SQL;
}
```



= BUROTIX ()