le cnam

MySQL

Janvier 2015

Introduction aux bases de données et à SQL

- **SQL: structured Query Language** :
  Créer, extraire, modifier, supprimer, fusionner …
  - Cad communiquer avec des bases de données relationnelles dotées d'un SGBDR
  - Apparu en 1979, standardisé ANSI (v3 en 1999)
- Bases de données:
  - Sert au stockage de données, ordonnées en unités hiérarchisées
  - Contenu :
    - Des tables (nom) qui contiennent des enregistrements (lignes)
    - Un enregistrement est composé de plusieurs champs (colonnes)
    - Chaque champ a un nom et un type de données (texte, nombre,date,heures, blob (Binary Large Object pour les images)
  - Les tables sont reliées entre elles par un **schéma de relation ou "vue conceptuelle" qui décrit la structure (relationnelle) de la base**

SQL

- **SQL : langage non procédural**
  - **On indique ce qu'on cherche, le SGBD décide comment récupérer l'information**
- **Langage de manipulation de données** : requêtes regroupant des instructions SQL
- Une requête est une question (avec conditions)
  - Si la BD comporte des données vérifiant la condition, SQL retourne les données.
- PHP permet d'incorporer des requêtes dans un programme et de transmettre ces requêtes au SGBD par des fonctions php.
- La plupart des SGBDR sont accessibles (Oracle, Sybase, Microsoft SQL Server, PostgreSQL , MySQL,…)

Détails PHP/SGBD (1)

La connexion s'obtient par une fonction php :

mysql_connect("nom_serveur","nom_utilisateur","mot_passe"**)**;

mssql_connect("nom_serveur,"nom_utilisateur","mot_passe"**)**;

oci_connect("nom_utilisateur","mot_passe", "nom_base"**)**;

pg_connect("dbname=nom_base host=nom_serveur port=num_port user=nom_utilisateur password=mot_passe"**)**;

sybase_connect("nom_serveur","nom_utilisateur","mot_passe"**)**;

qui retourne un $id_connexion

Il existe 2 façons de se connecter à une Base de Données.

- Les connexions non-persistantes (*base_**connect***).
- Les connexions persistantes (*base_**pconnect***) i.e. réutilisables

ATTENTION : s'assurer de la fermeture correcte des processus par base_close($id_connexion)

Détails PHP/SGBD (2)

- **Sélection** d'une base de données

  mysql_select_db($nom_base_donnee, $id_connexion);

  sybase_select_db($nom_base_donnee, $id_connexion);

  (faite dans le pg_connect et oci_connect)

- **Création** de base de données (ou par sql)

  mysql_create_db ($nom_base_donnee, $id_connexion);

  mssql_create_db($nom_base_donnee, $id_connexion);

- **Suppression** d' une base de données (ou par sql)

  mysql_drop_db ($nom_base_donnee, $id_connexion);

  mssql_drop_db($nom_base_donnee, $id_connexion);

- **Utilisation de SQL:**

$requete = "CREATE DATABASE nom_base_donnee";

$pg_exec($id_connexion,$requete);

Sql DDL - Data Definition Language (1)

```
CREATE TABLE  Fiche_Personne  (
 ID INT(8) UNIQUE,
 Sexe VARCHAR(50) NOT NULL,
 Nom VARCHAR(50) NOT NULL,
 Prenom VARCHAR(50) NOT NULL,
 Snd_Prenom VARCHAR(50) NULL,
 Adresse VARCHAR(100) NOT NULL,
 Code_Postal INT(5) NOT NULL,
 Ville VARCHAR(30) NOT NULL,
Telephone INT(10) NULL,
 eMail VARCHAR(30) NULL) ;
```

```
-- Création d'une table
-- Avec contraintes  sur
-- certains champs :
-- . ID Unique
-- . sexe, nom, … obligatoires
…
```

CONTRAINTES

NULL indique que la colonne peut NE PAS être renseignée.
NOT NULL indique que la colonne DOIT contenir une valeur.
PRIMARY KEY indique que la colonne constitue la clé primaire de la table.
UNIQUE impose que chaque valeur de la colonne doit être unique.

Sql DDL - Data Definition Language (2)

- D'autres fonctions:
  - ALTER TABLE pour supprimer, changer la définition des colonnes ou les contraintes
  - DROP TABLE pour supprimer une table (avec variantes RESTRICT(erreur si référencement par une vue) ou CASCADE
  - CREATE INDEX : pour créer un index référençant une ou plusieurs colonnes (performance)
  - DROP INDEX
  - CREATE VIEW permet de créer une table virtuelle par une requête de sélection
    - Utilisable comme une table
    - Pas de stockage physique
    - Permet de ne donner accès qu'aux données définies dans la vue (sécurité)

CREATE VIEW fiche_femme **AS** SELECT nom, prenom, adresse, cp, ville FROM tbl_personnel WHERE sexe = 'F'

Sql DQL (Data Query Language)

- **SELECT * | { [ALL | DISTINCT] nom_colonne,...,nom_colonneN } FROM nom_table WHERE Condition**
  - Sélectionne un à plusieurs champs (distincts si DISTINCT)

  FROM l'extraction provient de une ou plusieurs tables

  WHERE la condition booléenne est vérifiée

Exemple :

SELECT **nom_table**.nom_colonne1,..., **nom_tableN**.nom_colonneP F**ROM nom_table,..., nom_tableN**

WHERE nom_tableN.nom_colonneJ='F'

ORDER by nom_table.nom_colonne1

L'**expression conditionnelle** peut être construite à l'aide de **plusieurs prédicats** constitués d'**opérateurs de comparaisons** (= | <> | != | > | > = | !> | < | <= | !<) et séparées par des **opérateurs booléens** *AND*, *OR* ou *NOT*.

Sql DQL (Restriction sur une comparaison de chaîne)

- Le prédicat *LIKE* permet de faire des comparaisons sur des chaines grâce à des caractères, appelés caractères *jokers* :
- Le caractère **%** permet de remplacer une séquence de caractères (éventuellement nulle)
- La caractère _ permet de remplacer un caractère (l'équivalent du "blanc" au scrabble...)
- Les caractères **[-]** permettent de définir un intervalle de caractères (par exemple [J-M])

- Exemple :
  La sélection des contacts dont le nom a un E en deuxième position se fait par l'instruction :

```
SELECT * FROM AGENDA WHERE nom LIKE "_E%"
```

Exemple

- Table personnes

table	personnes			
Champs	Id	Nom	Prénom	CP
type	INT(10)	varchar(20)	varchar(20)	INT(5)

Id: clé unique (clé primaire) pour accès rapide non équivoque à un enregistrement

- Table Code_postal

Code_postal	
nom_ville	CP
varchar(20)	INT(5)

Sql : select sur plusieurs tables

personnes			
Id	Nom	Prénom	CP
INT(10)	varchar(20)	varchar(20)	INT(5)

Code_postal	
nom_ville	CP
varchar(20)	INT(5)

- SELECT P.Id, P.nom,P.prenom,C.nom_ville FROM personnes P, code_postal C WHERE P.CP=C.cp

nous permet d'accèder au tableau

Id	Nom	Prénom	nom_ville

SQL DML (Data manipulation Language)

- UPDATE : La commande *UPDATE* permet de mettre à jour des données existantes au sein d'une table.

  UPDATE Nom_Table SET Col_1 = Nouv_Val_1[, Col_2 = Nouv_Val_2[, ..., Col_N = Nouv_Val_N]] [WHERE Condition]

- INSERT INTO: La commande *INSERT INTO* permet d'ajouter des données dans une table.

  INSERT INTO Nom_Table (Champ_1, Champ_2, ..., Champ_N)
  VALUES (Valeur_1, Valeur_2, ..., Valeur_N)

- DELETE FROM :La commande *DELETE FROM* permet de supprimer des enregistrements au sein d'une table.

  DELETE FROM Nom_Table [WHERE Condition]

  PS : si pas de condition, tous les enregistrements sont supprimés…

On peut utiliser une sous-requête pour exprimer la condition :

DELETE FROM fiche_personnes WHERE ville IN (SELECT ville FROM table_ville)

SQL DCL - Data Control Language (1)

- Pour gérer les droits d'accès aux objets de la base
  - **Un utilisateur de base de données doit posséder un compte de sécurité** afin d'accéder aux tables ou à tout autre élément, ou d'obtenir des droits d'exécution de requêtes SQL.
  - Pour cela, il faut non seulement que l'utilisateur soit créé mais également que des privilèges lui soient accordés.
- ***L'administration des utilisateurs est liée à l'implémentation***
- ***Exemple : sous postgresql:***
  - **L'administrateur crée des utilisateurs par CREATE USER**
  - **ALTER USER permet de modifier ceux-ci**
  - **GRANT et REVOKE permettent de gérer leurs droits**

SQL DCL - Data Control Language (2)

- ***Sous mysql,***
  - **on dispose d'une base utilisateur mysql, installée à l'utilisation qui gère les utilisateurs et leurs privilèges d'accès sur l'ensemble des bases de données, y compris mysql.**
    C'est l'utilisateur privilégié "root" qui est en charge d'administration de cette base. C'est donc le seul qui pourra ajouter des utilisateurs, en retirer, et modifier leurs droits.
  - D'où l'urgence à définir un mot de passe pour cet utilisateur "root"
- tables : user,db,host
  - La table **user** recense tous les utilisateurs qui ont accès au serveur des bases de données MySQL.
    Elle contient les privilèges de chacun d'entre eux dans tout mysql**.**
  - La table **db** permet de lier un utilisateur à une ou plusieurs base(s).
  - La table **host** permet de lier un hôte à une base de données.

SQL DCL - Data Control Language (3)

Toujours sous MySQL, pour créer un utilisateur et lui affecter des droits on peut donc directement travailler sur ces tables.

- Il est cependant conseillé de travailler avec les requêtes **grant** et **revoke** : Les commandes GRANT et REVOKE permettent à l'administrateur système de créer et supprimer des comptes utilisateur et de leur donner ou retirer des droits.

Les droits sont donnés à 4 niveaux :

- **Niveau global** :Les droits globaux s'appliquent à toutes les bases de données d'un serveur. Ces droits sont stockés dans la table mysql.user.

  *Exemple :REVOKE ALL ON *.* retirera seulement les privilèges globaux.*

- **Niveau base de données** **Les droits de niveau de base de données s'appliquent à toutes les tables d'une base de données. Ces droits sont stockés dans les tables mysql.db et mysql.host**.

  Exemple REVOKE ALL ON db. retirera seulement les privilèges de base de données*

- **Niveau table :** Les droits de table s'appliquent à toutes les colonnes d'une table. Ces droits sont stockés dans la table mysql.tables_priv.

  REVOKE ALL ON db.table retirera seulement les privilèges de table.

- **Niveau colonne** : Les droits de niveau de colonnes s'appliquent à des colonnes dans une table. Ces droits sont stockés dans la table mysql.columns_priv.

SQL et PHP

- **Le langage PHP dispose de nombreuses fonctions permettant de travailler sur des bases de données MySQL.**
- L'utilisation de ces fonctions nécessite une **compilation de PHP avec le support MySQL en activant l'option**

--with-mysql=[répertoire].

Le répertoire indiqué correspond au chemin des librairies du SGBDR MySQL.

Pour posgresql : **option**

--with-pgsql[=répertoire]

- **Wamp ou easyPHP** fournissent par défaut **MySQL et phpmyadmin** (outil web d'administration de BD MySQL)

Requêtes SQL : les bases

- **Voici les étapes nécessaires pour passer une requête SELECT, (détails dans la suite) :**
  - **Ouverture d'une Connexion au serveur MySQL** (authentification) : nous récupérons une *ressource de connexion* sur laquelle nous pouvons travailler.
  - **Sélection d'une base de données**
  - **Envoi de la requête** : nous récupérons ici une *ressource*, qui contient (mais pas de manière immédiatement accessible) la réponse à notre requête.
  - **Vérification de la validité de la ressource récupérée** : pour s'assurer que la réponse du serveur est valide.
  - **Extraction des données** à partir de la ressource : pour récupérer les informations dans un format exploitable par PHP.
  - **Fermeture de la connexion**.

MySql : Connexion et sélection

```
<?php
// Connexion à MySQL
    $link = mysql_connect("mysql_hote", "mysql_user",
    "mysql_pw") or die("Impossible de se connecter");

    echo "Connexion à MySql réussie";

// Sélection d'une BD
    mysql_select_db("ma_BD") or die("sélection ma_BD
    impossible");

     echo "Connexion à la Base de Donnée 'ma_BD' réussie";
?>
```

MySql : Création d'une BD

```
<?php
//Création d'une BD
$sql = 'CREATE DATABASE ma_nouvelle_BD';
if (mysql_query($sql, $link))    {
     echo "Base de données créée correctement<p />";
}else{
      echo 'Erreur lors de la création de la base de données : ‘;
      echo  mysql_error() . "<p />";
}
?>
```

On peut aussi utiliser la fonction php :

bool **mysql_create_db** ('ma_nouvelle_BD',$link); *qui retourne TRUE or FALSE*

SQL query (select) (1)

- **Passage d'une requête SELECT et récupération d'une ressource**

```
$result=mysql_query($requete,$link)
        or die($requete. " - " . mysql_error());
```

Retourne une ressource *pour une requête SELECT*.

Retourne true ou false pour les autres requêtes (*UPDATE*, *DELETE*, *DROP*, etc…)

le "or die" affiche l'erreur MySQL en cas d'échec de la requête,

SQL query(2) Exploitation de la ressource

- **Nombre d'enregistrements retournés :**

  $nb_result=mysql_num_rows ($result)

- **Récupération des résultats (1)**
  - **Tableau : mysql_fetch_array ($result)**
  - extrait un enregistrement (et un seul), une ligne de la table et déplace le pointeur.
  - Retourne false si plus de ligne

Exemple :

```
$result = mysql_query("SELECT id, nom FROM fiche_personne");
while ($row = mysql_fetch_array($result)) {
    printf("ID : %s  Nom : %s", $row[0], $row[1]);
  }
```

SQL query (3) *exemple 2 de base*

- **Exemple 2**

```
$link = mysql_connect("localhost", "cours", "passw");
$query = "SELECT * FROM table_test";
$result = mysql_query($query, $link) or die($query . " - " .
    mysql_error());
$nbResults = mysql_num_rows($result);
echo " on a $nbResults enregistrements   <br/> "  ;
//affichage des champs id et comment de la table table_test
    (tableau)
while ($tab = mysql_fetch_array($result)){
    echo $tab['id'] . " : " . $tab['comment']; echo "<br />";
}
```

SQL query (4)

- **Récupération des résultats (2)**
  - **mysql_fetch_assoc ($result)**
    - Retourne un tableau associatif correspondant à la ligne récupérée et déplace le pointeur.
    - Retourne false si plus de ligne

Exemple :

```
$result = mysql_query("SELECT id, nom FROM fiche_personne");
while ($row = mysql_fetch_assoc($result)) {
        echo $row["id"];
    echo "  ";
        echo $row["nom"];
}
```

SQL query(5) : Fermeture de la connexion

mysql_close($link);

Vision Objet (à revoir avec PHP5) :

$obj = mysql_fetch_object($result)

les attributs de l'objet portent le nom des champs de la table MySQL

```
//affichage des champs id et comment de la table table_test
while ($obj = mysql_fetch_object($result)){
    echo $obj->id . " : " . $obj->comment; echo "<br />";
}
mysql_close($link);
```

SQL query (6)

- Récupération des résultats pour d'autres requêtes que select

Exemple : Pour récupérer le nombre de lignes affectées par une requête : update, insert, delete

mysql_affected_rows() : Retourne un entier

le cnam

Injection et BD

Janvier 2015

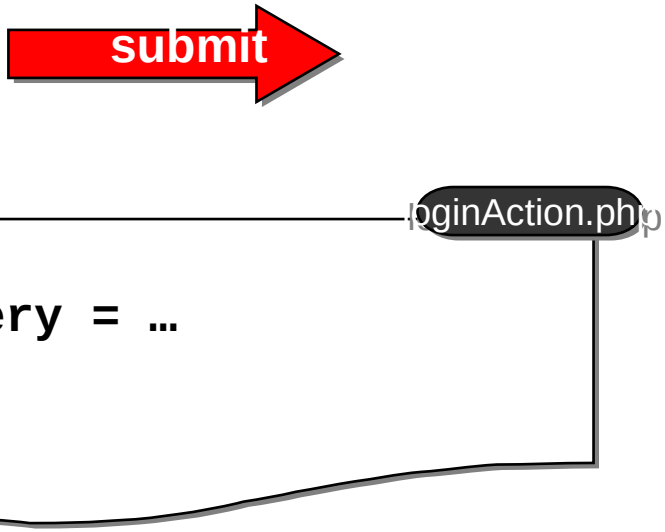
Injection SQL

- consiste à changer le but premier d'une requête SQL, grâce à des variables modifiables par l'internaute.
- se prémunir de ce type d'attaque en ne faisant jamais confiance aux données saisies et en les filtrant

Injection SQL : exemple (1)

Exemple : Imaginons une page php permettant à un utilisateur (enregistré en BD) de se connecter par un formulaire où il indique son login et son mot de passe:

```
<form method="POST" action= "loginAction.php" >
    <input type="text" name= "login"><br>
    <input type="password" name= "passwd"><br>
    <input type="submit" value="Valider">
</form>
```



Injection SQL : exemple (2)

Supposons disposer d'une table de **membres** ainsi définie :

```
CREATE TABLE users (
    id int(10) NOT NULL auto_increment,
    login varchar(25),
    password varchar(25),
    nom varchar(30),
    email varchar(30),
    PRIMARY KEY (id)
    )
```

Injection sql : exemple (3)

Sur le serveur, le code de loginAction.php ci dessous va s'exécuter :

```
$query = " SELECT id FROM users WHERE login= '" . $_POST['login']
        . "' AND password='"  .  $_POST['passwd'] . "'";
```

Si l'utilisateur a saisi dans le champ passwd: " OR 1=1

La requête exécutée est alors :

SELECT id FROM users WHERE login= 'dupont' AND password='' OR 1=1

Si l'utilisateur a saisi dans le champ login: dupont' # (ou dupont' --)

La requête exécutée est alors

SELECT id WHERE login = 'dupont' # ' AND password = '4e596e'

Soit SELECT id WHERE login = 'dupont'

(car ce qui suit # est considéré comme un commentaire)

Dans les 2 cas le pirate peut se connecter sur le compte de login dupont !

Des remèdes contre l’injection sql

Toujours vérifier de manière précise et exhaustive les données venant de l'utilisateur :

- Utiliser addslashes($chaine) ou **mysqli_real_escape_string($chaine)**
  (Protège $chaine pour la passer à mysqli_query)

- Utiliser des procédures stockées, à la place du SQL dynamique. Les données entrées par l'utilisateur sont alors transmises comme paramètres, sans risque d'injection.