Principes d'utilisation des systèmes de gestion de bases de données

Le mode programme : PHP

M1, Informatique

Emmanuel Waller, LRI, Orsay

Le mode programme : PHP

- But
- Exemple
- Principe : intuition
- Devant les machines
- Architecture
- PHP et les pbs MP
- PHP et l'architecture en couches

but

- On sait écrire des pages web (PHP)
  - Appelables depuis le web
  - Qui, au cours de leur exécution :
    - Font des calculs
    - Font des effets de bord (accès fichiers, etc.)
    - Ne savent pas (encore) accéder à la base
  - Construisent dynamiquement une page web (HTML) renvoyée comme résultat de leur appel
- But : accès à la base pendant l'exécution d'un programme PHP = gestion des pbs MP en pHP

exemple

- Table produit
- Insérer nouveauproduit dont la quantité est passée en paramètre
- Fichier : td1_maj.php
- Démonstration : par url, par formulaire

  + view page source
- Rem : aucun echo
- Vérification : en interactif

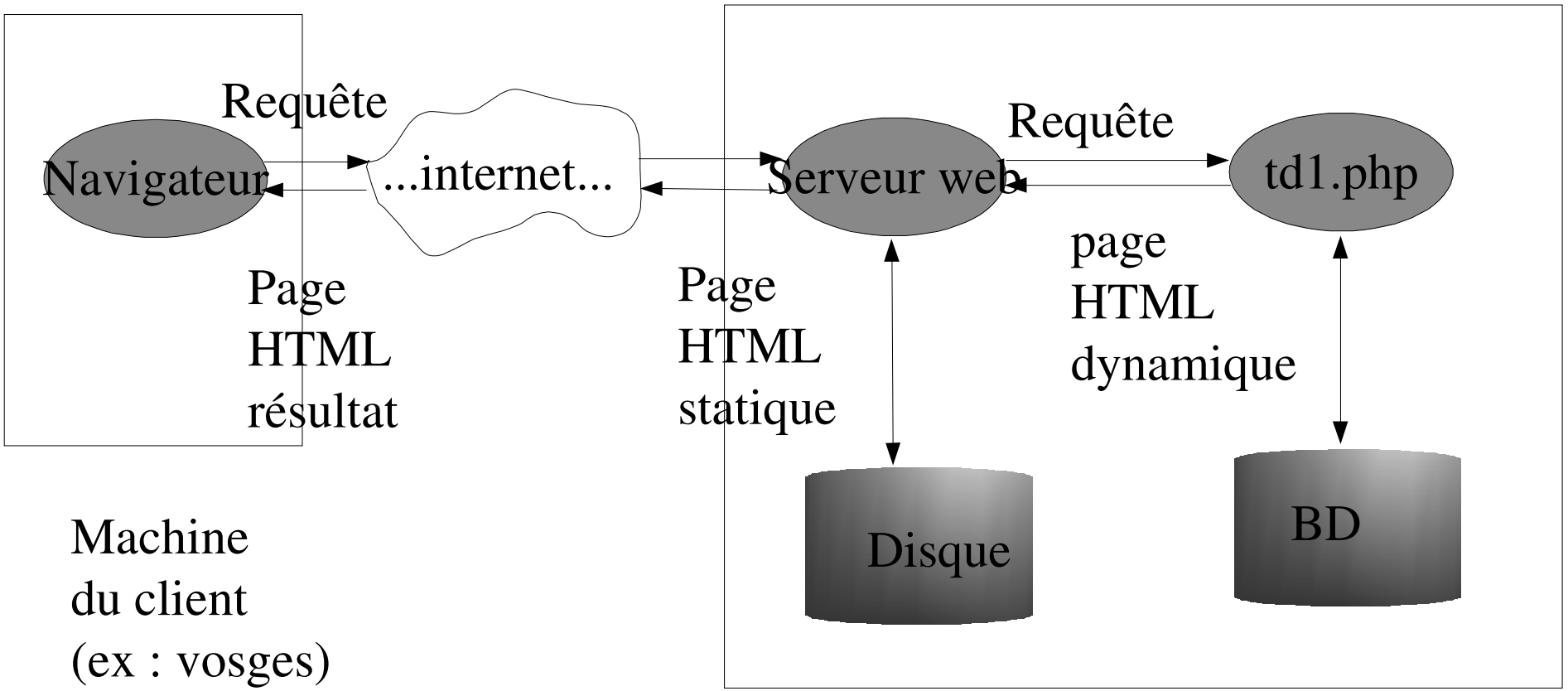
Principe : intuition

- connexion au serveur BD, construction de l'ordre
- analyse ordre renvoie objet intermédiaire : «curseur»
- exécution ordre via ce curseur : lignes résultat rangées dans curseur
- déconnexion

devant les machines

- Fichier environnement, page cours

Architecture



Machine
du client
(ex : vosges)

2 machines : serveur web (ssh1)
et serveur BD (servora)

Détailler le déroulement

- Requête 1 : http://.../td1.html
- Requête 2 : via action du formulaire : http://.../td1.php
- Rem : td1.php : fichier (rectangle) provenant du disque (corriger dessin)

Le mode programme : PHP

- But
- Exemple
- Principe : intuition
- Devant les machines
- Architecture
- PHP et les pbs MP
- PHP et l'architecture en couches

Rappel : les ps du mode programme

- Interface, portabilité
- Connexion
- Envoi serveur d'ordres BD (SQL, PL/SQL)
  - Curseurs
  - Paramètres
  - SQL dynamique
- Erreurs
  - côté serveur
  - côté programme lors échange données
- Transactions

PHP et les pbs MP

- un programme PHP souhaitant accéder à BD doit gérer ces problèmes
- les outils nécessaires pour gérer les problèmes du mode programme sont fournis par des bibliothèques de fonctions (assez proche JDBC)

pbs MP : interface (½)

- comment exécuter un ordre BD depuis PHP ?
- Comme Java/JDBC (presque)
- PHP appelle des fonctions d'une bibliothèque
- ces fonctions encapsulent les accès au serveur BD
- les ordres BD sont :
  - certaines fonctions de la bibliothèque (ex : commit)
  - chaînes de caractères paramètres de fonctions de la bibliothèque (ex : select)

pbs MP : interface (2/2)

- Java : JDBC :
  - Fournit interface unique à tous SGBD
  - La bibliothèque fournie par chaque SGBD est l'implantation de l'interface JDBC
- PHP :
  - Il n'y a pas d'interface unique
  - Chaque SGBD fournit une bibliothèque (donc noms de fonctions différents)
- Oracle : bibliothèque OCI
  - (autre obsolète : ORA)

pbs MP : portabilité

- Non
- Conséquence de l'interface par bibliothèques avec fonctions de noms différents
- Mais chaque SGBD fournit des fonctions assez proches (pour gérer pbs MP), donc portage raisonnable

pbs MP : connexion

- $c = ocilogon(waller, waller, orcl)
  - établit connexion avec le serveur
  - renvoie « objet » correspondant, utilisé pour interactions avec le serveur
- Déconnexion : ocilogoff($c) (sinon implicite)
- Rem : possible (hors programme) :
  - Connexion unique pour plusieurs pages (subtilités)
  - Une page PHP : plusieurs connexions avec plusieurs serveurs

déroulement d'un programme : comme Java/JDBC

- début de l'exécution d'un programme PHP : inconnu du serveur
- connexion : devient client
- envoi ordres SQL au serveur
- Déconnexion : termine en tant que client
- inconnu du serveur : continue son exécution, puis termine comme programme

Exemple précédent

- illustrer

pbs MP : gestion des erreurs BD (intuition) (½)

- rappel : un ordre BD envoyé au serveur peut générer une erreur
- fonction PHP (ex : ocilogon) ayant demandé cet ordre :
  - échoue
  - Affiche un message d'erreur (code et message Oracle, numéro de ligne du fichier)
- programme ne s'interrompt pas (différent JDBC) : éventuels ordres suivants sont exécutés (tentative)

pbs MP : gestion des erreurs BD (intuition) (2/2)

- Comment le programmeur est-il informé ?
  - L' « objet associé à l'ordre » a une valeur particulière
  - Ex :

```
$c = ocilogion(...);
if ($c == 0)
    echo « erreur »;
```

Démonstration (½)

- Exemple précédent
- + Ocilogon faux
- Suivi de Ocilogon correct
- Voir code suivant

```
$connection = ocilogon(iety, waller, orcl);
if ($connection == 0) {
  echo "il y a eu une erreur, ";
  echo "mais l'exécution du fichier PHP ne s'arrête pas !<br>";
}
echo "je recommence la connexion<br>";
$connection = ocilogon(waller, waller, orcl);
if ($connection == 0) {
  echo "il y a eu une erreur, ";
  echo "mais l'exécution du fichier PHP ne s'arrête pas !<br>";
 }
$query = "select nom from produit where prix < 300";
echo "(affichage pour vérifier : ".$query.")<br>";
... ociparse... ociexecute... while(ocifetchinto...) ...  // curseur
```

pbs MP : transactions (intuition)

- Confirmation : ocicommit
  - Rem : inutile pour nous : confirmation implicite automatique en fin de page

pbs MP : exécution d'ordres BD

- Rappel : pbs MP : non requête, paramètres, SQL dynamique, curseurs
- principe et déroulement
- ordres avec paramètres
- SQL dynamique
- Curseurs

principe et déroulement

- principe :
  - « analyse »
  - exécution
  - si curseur : récupération des lignes
- $ordre = « insert into t values('toto', 7) »
- $p = ociparse($connexion, $ordre)
  - « analyse » l'ordre $ordre sur la connexion $connexion
  - renvoie (comme connexion et toutes fonctions) :
    - « objet » représentant l'ordre analysé
    - 0 en cas d'erreur

- ociexecute($p)
  - exécute l'ordre analysé associé à $p
  - renvoie (toujours) :
    - « objet » représentant le « résultat » de l'ordre exécuté :
      - valeur (détails hors programme)
      - curseur
    - 0 en cas d'erreur
- Appel procédure PL/SQL : même chose :

  $ordre = « begin p(5); end; »

  ociparse, ociexecute

Exemple précédent

- illustrer

pbs MP : SQL dynamique

- immédiat car ordre est une chaîne de caractères : concaténation par « . »
- Impossible faire autrement : PHP ne fait que SQL dynamique
- contrepartie :
  - ordres non connus lors compilation
  - => vérification statique impossible (ex : table n'existe pas)
  - programmes moins robustes
- vérification statique : JSP/JSTL (hors programme)

Ordres avec paramètres

- Exemple ci-joint
- Comme SQL dynamique en PL/SQL

Ordres avec paramètres : à la main

- Par substitution de variable :
  - variable remplacée par sa valeur
  - attention : valeur est toujours chaîne
- Ex :

  $x = « toto »; $y = 7;

  insert into t values('$x', $y);
- Attention : y compris dans une chaîne « ... $y ... »
- Voir code suivant + démo http://...insertion.php

```
$connection = ocilogon(waller, waller, orcl);
$x = "cours peche a la ligne";
$y = 350;
$ordre = "insert into produit values ('$x', $y, 4)";
echo "(verification : $ordre)";
$cursor = ociparse($connection, $ordre);
ociexecute($cursor);
ocilogoff($connection); // implicite sinon
```

pbs MP : curseurs

- $curseur = ociexecute($p)
- ocifetch($curseur)
  - déplace pointeur sur ligne suivante
  - renvoie 0 quand pas de nouvelle ligne
- ociresult($curseur, $i)
  - renvoie valeur colonne $i ligne courante après fetch
- méta-données :
  - ocinumcols($curseur) : nombre colonnes curseur
  - ocicolumnname($curseur, $i) : nom colonne $i

```
<?php
$connexion = ocilogon(toto, toto, orcl);
$ordre = "select client form billet where km > " . $km;
$cursor = ociparse($connexion, $ordre);
ociexecute($curseur);
while (ocifetchinto($curseur, $ligne))
    echo $ligne[0] . " <br> ";
ocilogoff($connexion);
```

- si plusieurs champs dans ligne : $ligne[0], $ligne[1]
- rem : les echos de PHP forment une page HTML...

Le mode programme : PHP

- But
- Exemple
- Principe : intuition
- Devant les machines
- Architecture
- PHP et les pbs MP
- PHP et l'architecture en couches

PHP et l'architecture en couches

- Comme JDBC
- même principe qu'en objet :
  - les données sont encapsulées
  - accès par procédures stockées (sauf rares cas)
- PHP :
  - interagit peu avec la base :
    - appelle les procédures stockées
    - gère commit et rollback
    - si nécessaire : curseur pour affichage
  - est plutôt dédié à la partie non BD de l'application

le mode programme / web : vision naïve

- application en mode programme = ordres BD + code normal autour (+ connexion)
- appli web = page web normale (écrite en HTML) :
  - saisit données
  - appelle une appli BD en mode programme
  - affiche résultats de l'appli BD
- appli BD appelée par page web (écrite en PHP) :
  - fait ce qu'elle veut
  - accède BD (connexion, ordres SQL)
  - affiche des résultats

Exemple ci-joint : outils nécessaires : vision naïve

- ex : afficher les clients allant à plus de x km
- écrire page en HTML qui :
  - saisit x
  - déclenche le programme PHP
  - a une présentation personnalisée
- écrire un programme en PHP qui :
  - se connecte au serveur BD
  - exécute la requête
  - récupère chaque ligne résultat et l'affiche

Erreurs possibles

- Démonstration : les mêmes + toutes occurrences gestion pbs MP

Le mode programme : PHP

- But
- Exemple
- Principe : intuition
- Devant les machines
- Architecture
- PHP et les pbs MP
- PHP et l'architecture en couches

(voir « ouverture »)