

Présentation faite au colloque IBM des 5 et 6 avril 2012
IBM Forum de Bois-Colombes

Session
DB2 et PHP - Bonnes pratiques sous IBM i

Grégory JARRIGE - [Linkedin](#)

DB2 et PHP - Bonnes pratiques sous IBM i



- Intervenant : Grégory JARRIGE
- Expert PHP 5 certifié par Zend en février 2010
- Début 2022, totalise plus de 30 ans d'expérience dans le développement d'applications de gestion sur plateforme IBM i (langages RPG et SQLRPGLE, AGL Adelia, DB2 SQL-PSM, PHP, Node.js).
- Intervient régulièrement sur des problématiques de performances SQL et d'urbanisation de systèmes d'informations en environnement IBM i.
- A été régulièrement formateur pour IBM et Zend, pour le cours PHP dédié aux développeurs IBM i, et pour le cours Zend Server
- A publié de nombreux article sur le site www.foothing.net (ex. XDocs400) autour de SQL DB2, RPG et PHP.

DB2 et PHP - Bonnes pratiques sous IBM i

“Un expert est une personne qui a commis toutes les erreurs qui peuvent être commises, dans un domaine bien précis.”

Niels Bohr (Prix Nobel de Physique 1922)

DB2 et PHP - Bonnes pratiques sous IBM i

Durant cette session, nous allons passer en revue un certain nombre de techniques permettant d'utiliser le couple PHP - DB2 for i, de manière optimale. Parmi les sujets que nous allons aborder :

- quel connecteur PHP faut-il utiliser pour accéder à DB2 for i ?
- quelles techniques utiliser pour gérer la pagination SQL ?
- quelles techniques utiliser pour travailler avec des dates d'effet en SQL ?
- comment appeler des programmes RPG à partir de PHP en s'appuyant sur DB2 ?
- comment gérer les problèmes de verrouillage ?
- quel bénéfice peut-on retirer d'un "wrapper" de bases de données ?
- quel bénéfice peut-on retirer de l'utilisation du design pattern « Active Record » ?
- quels outils utiliser pour développer des modules de type C.R.U.D. ?

Nous verrons des exemples concrets, développés avec Zend Studio et déployés sur différentes versions de Zend Server (pour Windows et IBM i).

Mon objectif dans cette présentation est de vous démontrer que le couple PHP-DB2 permet de développer rapidement et simplement des outils puissants adaptés à des usages très variés (administration système, administration base de données, applications "métiers", etc...).

Choisir son connecteur DB2

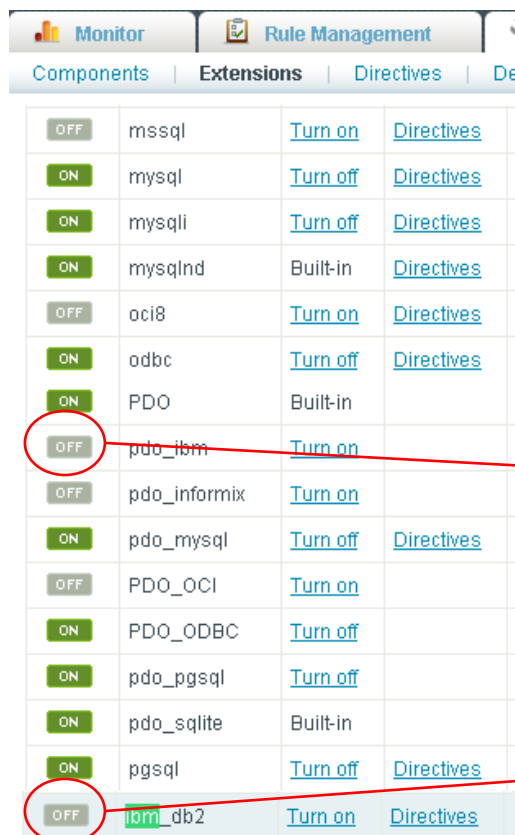
Le choix du connecteur DB2 dépend de critères qui ne sont pas uniquement techniques... tour d'horizon:

- Votre application PHP est-elle destinée à être utilisée uniquement sur serveur IBM i ?
- Votre application PHP est-elle destinée à être utilisée à la fois sur serveur IBM i et sur serveur Windows/Linux (par exemple chez des partenaires ou des succursales qui ne peuvent être connectées au serveur IBM i du siège) ?
- Votre application PHP est-elle destinée à servir de projet pilote, en étant développée d'abord sur serveur Windows/Linux, avec pour objectif de la déployer ultérieurement sur serveur IBM i ?

Si un seul connecteur DB2 pouvait couvrir tous les usages, ce chapitre n'aurait pas de justification, mais puisque ce n'est pas le cas, il est judicieux de faire le tour du propriétaire. La diapo suivante présente quelques unes des extensions bases de données disponibles pour 2 versions de Zend Server (Windows et IBM i).

Choisir son connecteur DB2

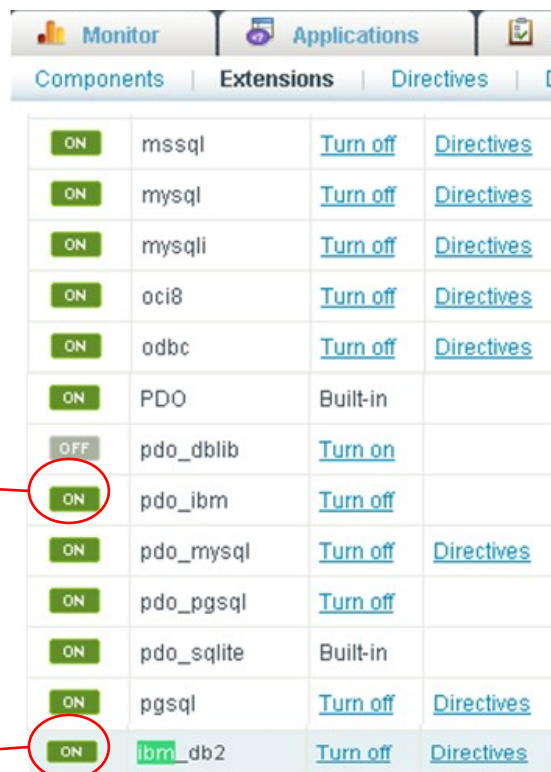
Zend Server 5.6.0 pour Windows



The screenshot shows the 'Extensions' tab in the Zend Server 5.6.0 for Windows interface. It displays a list of database extensions with their status (ON/OFF) and actions (Turn on/off, Directives). The extensions pdo_ibm and ibm_db2 are circled in red, indicating they are not active by default.

Status	Extension	Action	Directives
OFF	mssql	Turn on	Directives
ON	mysql	Turn off	Directives
ON	mysqli	Turn off	Directives
ON	mysqlnd	Built-in	Directives
OFF	oci8	Turn on	Directives
ON	odbc	Turn off	Directives
ON	PDO	Built-in	
OFF	pdo_ibm	Turn on	
OFF	pdo_informix	Turn on	
ON	pdo_mysql	Turn off	Directives
OFF	PDO_OCI	Turn on	
ON	PDO_ODBC	Turn off	
ON	pdo_pgsql	Turn off	
ON	pdo_sqlite	Built-in	
ON	pgsql	Turn off	Directives
OFF	ibm_db2	Turn on	Directives

Zend Server 5.6.0 for i



The screenshot shows the 'Applications' tab in the Zend Server 5.6.0 for i interface. It displays a list of database extensions with their status (ON/OFF) and actions (Turn on/off, Directives). The extensions pdo_ibm and ibm_db2 are circled in red, indicating they are active by default.

Status	Extension	Action	Directives
ON	mssql	Turn off	Directives
ON	mysql	Turn off	Directives
ON	mysqli	Turn off	Directives
ON	oci8	Turn off	Directives
ON	odbc	Turn off	Directives
ON	PDO	Built-in	
OFF	pdo_dblib	Turn on	
ON	pdo_ibm	Turn off	
ON	pdo_mysql	Turn off	Directives
ON	pdo_pgsql	Turn off	
ON	pdo_sqlite	Built-in	
ON	pgsql	Turn off	Directives
ON	ibm_db2	Turn off	Directives

Attention : Les extensions pdo_ibm et ibm_db2 ne sont pas activées par défaut sur Zend Server pour Windows/Linux. Elles ne peuvent être activées que si une base DB2 pour Windows/Linux a été détectée par le logiciel d'installation du Zend Server. Si ces extensions sont activées, elles sont dédiées à la consultation de bases DB2 pour Windows/Linux (et non pas IBM i), contrairement aux extensions de même nom utilisées sur Zend Server for i.

Choisir son connecteur DB2

■ ODBC et son extension pour PHP

- Une extension PHP basée sur le standard Microsoft ODBC
- Elle est trop limitée pour une utilisation avec DB2 for i, car elle ne gère pas certaines spécificités de la base de données (liste de bibliothèques...), et son support des procédures stockées DB2 serait incomplet (*je n'ai pas vérifié, faute de temps, si cette allégation émanant de confrères américains est fondée*).
- Cette extension est implémentée sur Zend Server pour IBMi. Vous pouvez donc envisager de porter sur IBM i des applications PHP utilisant cette extension, mais prévoyez une solide campagne de tests, et surveillez de près les performances.
- URL : <http://fr.php.net/manual/fr/book.uodbc.php>

– Exemple :

```
$conn = odbc_connect($database, $user, $password);  
if (!$conn) die("Connexion incorrecte: $database, $user");  
$sql = "select * from bibdta.tabdta";  
$r = odbc_exec($conn, $sql);  
if (!$r) die("<br />sélection incorrecte ".odbc_errormsg());  
echo "<br />$sql". PHP_EOL;  
echo "<table border='1'>";  
while ($row = odbc_fetch_array($r)) {  
    echo ("<tr><td>".implode("</td><td>",$row)."</td></tr>". PHP_EOL);  
}  
echo "</table>". PHP_EOL;
```

Choisir son connecteur DB2

■ PDO (PHP Data Objects)

- PDO est une librairie PHP qui fournit une « *abstraction d'interface* », terme barbare qui signifie que vous utilisez le même jeu de fonctions PHP pour exécuter des requêtes SQL, quelle que soit le SGBD utilisé. PDO est inclus dans le noyau de PHP depuis la version 5.1, et est en mesure de s'interfacer avec la grande majorité des bases de données, DB2 compris.
- L'implémentation de l'extension pdo_ibm sur IBM i fonctionne, mais elle est en l'état incomplète, et ne permet pas de gérer certaines spécificités de la base de données DB2 for i. Il est préférable d'utiliser l'extension "ibm_db2" dans ce contexte d'exécution.
- Sur un Zend Server pour Windows ou Linux, on peut attaquer une base DB2 for i en couplant l'extension PDO_ODBC avec le "Iseries Access ODBC Driver", driver fourni en standard avec le logiciel "System I navigator" d'IBM (nous y reviendrons dans une des diapos suivantes). Dans cette configuration, PDO offre un support complet des spécificités de DB2 for i (procédure stockées, liste de bibliothèques, etc...).
- URL : <http://fr.php.net/manual/fr/book.pdo.php>

Choisir son connecteur DB2

■ PDO (...suite) :

— Exemple 1 : connexion à MySQL

```
$user = "à définir"; $password = "à définir"; $database = "à définir";  
$dsn = 'mysql:host=localhost;dbname=' . $database; // définition de la « data source name » (DSN)  
$db = new PDO($dsn, $user, $password);  
$st = $db->query('SELECT id, libelle FROM ma_table');  
foreach ($st->fetchAll() as $row) {  
    print "id = {$row['id']} , nom = {$row['libelle']} <br />" . PHP_EOL;  
}
```

■ Exemple 2 : connexion à DB2 Express C (avec Zend Server pour Windows)

```
$user = "à définir"; $password = "à définir"; $database = "à définir";  
$dsn = "odbc:DRIVER={IBM DB2 ODBC DRIVER};HOSTNAME=localhost;PORT=50000;PROTOCOL=TCPIP;  
    DATABASE=$database";  
$db = new PDO($dsn, $user, $password);  
$st = $db->query('SELECT id, libelle FROM ma_table');  
foreach ($st->fetchAll() as $row) {  
    print "id = {$row['id']} , nom = {$row['libelle']} <br />" . PHP_EOL;  
}
```

Choisir son connecteur DB2

■ PDO (...suite) :

— Exemple 3 : connexion à DB2 for i avec Zend Server for i

```
$user = 'à définir'; $password = 'à définir'; $system = '*LOCAL';  
$dsn = "ibm:$system";  
  
/* implémentation de PDO for i très limitée :  
 * - impossibilité de choisir entre nommage SQL et nommage système,  
 * - impossibilité de définir une bibliothèque par défaut, ou une liste de bibliothèques  
 */  
  
$db = new PDO ( $dsn, $user, $password );  
  
// on est obligé de qualifier la bibliothèque de la table, avec un « . » comme séparateur  
$st = $db->query ( 'SELECT * FROM mabib.matable' );  
  
foreach ( $st->fetchAll () as $data ) {  
    echo $data['id'], ' ', $data['libelle'], '<br />' . PHP_EOL ;  
}
```

L'implémentation trop limitée de l'extension « pdo_ibm » en environnement IBM i ne permet pas d'envisager son utilisation sur des applications destinées à la production. Cette limitation constitue un frein dans le portage sur IBM i de certaines applications PHP (CMS, E-commerce...) s'appuyant sur PDO.

Choisir son connecteur DB2

■ PDO (...suite) :

— Exemple 4 : connexion à DB2 for i avec Zend Server Windows/Linux

```
$user = 'à définir'; $password = 'à définir'; $database = 'à définir';
$system = 'xxx.xxx.xxx.xxx'; // adresse IP de l'IBM i
$trace = 0; // les explications sur ce paramètre seront données dans la diapo suivante
$dsn = "odbc:DRIVER={iSeries Access ODBC Driver};SYSTEM=$system;DBQ=$database;TRACE=$trace";

try {
    $conn = new PDO ( $dsn, $user, $password );
    $conn->setAttribute ( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION ); // active la gestion d'exception PDO
    $conn->setAttribute ( PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC ); // préférable au mode FETCH_BOTH
    $conn->setAttribute ( PDO::ATTR_CASE, PDO::CASE_UPPER ); // pour récupérer les noms de colonne en majuscule
} catch ( PDOException $e ) {
    echo 'Connection failed: ' . $e->getMessage () . '<p/>';
}

Try {
    $st = $conn->query ( 'SELECT id, libelle FROM ma_table' );
    foreach ( $st->fetchAll () as $data ) {
        echo $data['ID'], ' ' , $data['LIBELLE'], '<br />' . PHP_EOL ;
    }
} catch ( PDOException $e ) {
    // insérez ici votre fonction de gestion d'erreur (voir exemples en lignes sur php.net)
}
```

Choisir son connecteur DB2

■ PDO (...suite) :

Dans l'exemple de la diapo précédente, nous avons utilisé le driver « iSeries Access ODBC Driver » fourni en standard avec le logiciel « iSeries Navigator d'IBM ».

Ce driver fournit le paramètre optionnel « TRACE », qui s'il est déclaré dans la variable \$dsn, permet d'activer le moniteur de performances. Les valeurs possibles pour ce paramètre sont les suivantes :

0 : pas de tracing (valeur par défaut si non paramétré)

1 : active le driver de tracing interne

2 : active le moniteur de base de données

4 : active le débogueur (STRDBG)

16 : active le tracing du job

La valeur du paramètre TRACE peut être le cumul de plusieurs des valeurs ci-dessus. Par exemple la valeur 6 permet d'activer simultanément les fonctions 2 et 4.

Le fait d'indiquer une valeur différente de zéro a pour effet de générer dans la bibliothèque QUSRSYS un fichier DB2 dont le nom est « QODB » suivi du numéro de travail. Ce fichier peut être analysé via SQL.

Choisir son connecteur DB2

Exemple de requête d'analyse pour le travail 385372:

```
SELECT * FROM QUSRSYS/QODB385372 WHERE QQRID = 1000
```

La colonne QQRID peut prendre l'une des valeurs suivantes :

1000 Record: SQL statement summary	3026 Record: Union merge
3000 Record: Arrival sequence	3027 Record: Subquery merge
3001 Record: Using existing index	3028 Record: Grouping
3002 Record: Temporary index created	3029 Record: Index ordering
3003 Record: Query sort	3005 Record: Table locked
3004 Record: Temporary file	3008 Record: Subquery processing
3006 Record: Access plan rebuild	3014 Record: Generic query information
3007 Record: Summary optimization data	3018 Record: STRDBMON and ENDDDBMON data
3010 Record: Host variable and ODP implementation	3019 Record: Retrieved detail (only with *DETAIL)
3021 Record: Bitmap created	3025 Record: DISTINCT processing
3022 Record: Bitmap merge	3030 Record: Query step processing
3023 Record: Temporal hash table created	5002 Record: SQL request executed by SQL Query Engine (SQE)

A noter : la fonctionnalité de tracing fournie avec le « iSeries Access ODBC Driver » n'a pas d'équivalent dans l'extension « ibm_db2 ». Mais on peut pallier ce manque en utilisant les techniques de monitoring de performances standard de l'IBM i, et notamment la commande CLP STRDBMON.

Pour de plus amples précisions sur les problématiques de performances de DB2 for i, se reporter au redbook IBM [SG246654](#).

Choisir son connecteur DB2

- **PDO** (...suite et fin) :
- Si vous souhaitez travailler avec PDO et le « iSeries Access ODBC Driver », je vous recommande la lecture de la documentation ci-dessous, qui vous donnera toutes les clés pour configurer correctement le DSN :
- <http://publib.boulder.ibm.com/infocenter/iseriess/v5r4/index.jsp?topic=%2Frzaik%2Frzaikconnstrkeywords%2Fgeneralprop.htm>
- Vous constaterez que l'on retrouve les mêmes paramètres que ceux que nous allons voir pour une connexion avec `db2_connect()`, mais avec une syntaxe légèrement différente.

Choisir son connecteur DB2

■ **ibm_db2**

- Extension PHP développée par IBM, elle fournit un support complet des spécificités de la base de données.
- Embarquée et activée en standard dans Zend Server for i
- Embarquée également sur Zend Server pour Windows, mais elle ne peut être activée dans ce contexte que si une instance de base de données DB2 est active sur le serveur Windows lors de l'installation du Zend Server. **Sur un Zend Server pour Windows, l'extension « ibm_db2 » ne sait dialoguer qu'avec une base DB2 pour Windows.**
- URL : <http://fr2.php.net/manual/fr/book.ibm-db2.php>
- Exemple : cf. diapo suivante

Choisir son connecteur DB2

■ ibm_db2 (suite)

— Exemple :

```
$user = ''; $password = '';  
$options = array ();  
$options ['DB2_ATTR_CASE'] = DB2_CASE_LOWER; // nom des colonnes retourné en minuscule  
$options ['i5_naming'] = DB2_I5_NAMING_OFF; // syntaxe SQL plutôt que système  
$options ['i5_lib'] = 'your_base' ; // bibliothèque par défaut pour les objets DB2 non qualifiés  
$conn = db2_connect ('*LOCAL', $user, $password, $options );  
if (! $conn) {  
    echo "Echec de la connexion. SQL Err:", db2_conn_error (), "<br />" . db2_conn_errormsg ();  
    exit ();  
}  
$sql = "SELECT * FROM mata ble";  
$stmt = db2_prepare ( $conn, $sql );  
if (db2_execute ( $stmt )) {  
    while ( $data = db2_fetch_assoc ( $stmt ) ) {  
        echo $data ['id'], ' ', $data ['libelle'], '<br />' . PHP_EOL;  
    }  
} else {  
    echo "Echec de la requête. ", db2_stmt_error ( $stmt ), ' ', db2_stmt_errormsg ( $stmt );  
}
```


Choisir son connecteur DB2

■ En résumé

Zend Server / OS	Serveur DB	db2_connect	PDO	Commentaire
Windows ou Linux	DB2 Express C	Oui	Oui	PDO avec IBM DB2 ODBC DRIVER
Windows ou Linux	DB2 for IBM i	Non	Oui	PDO avec ISERIES ACCESS ODBC DRIVER
IBM i	DB2 for IBM i	Oui	Non	PDO possible mais non recommandé pour l'instant
IBMi, Windows, Linux	Toute base autre que DB2 (MySQL, etc...)	Non	Oui	PDO couplé avec l'extension spécifique à chaque base de données

L'obligation de devoir choisir entre plusieurs connecteurs selon le type de plateforme de déploiement pose la question légitime de la portabilité et de la pérennité du code développé. On peut contourner cette difficulté en s'orientant vers l'une des 2 solutions suivantes :

1. Développer une couche d'abstraction qui encapsule la déclaration du connecteur et l'exécution du code SQL.
2. Utiliser une couche d'abstraction existante, comme par exemple celle que propose le composant Zend_DB du Zend Framework (sur lequel nous reviendrons dans quelques minutes).

Avant d'aborder ces sujets, finissons-en avec l'extension `ibm_db2`, en passant en revue les caractéristiques de la fonction `db2_connect`.

db2_connect()

CARACTÉRISTIQUES ET EXEMPLES

db2_connect()

- db2_connect() est une fonction PHP qui est liée à l'extension ibm_db2. Elle a pour effet de créer une variable de type "ressource" qui va permettre à l'interpréteur PHP de "dialoguer" avec la base de données
- Cette fonction reçoit 4 paramètres, les 3 premiers sont de type "string" et sont obligatoires, le 4ème est de type "array" et est optionnel :

```
resource db2_connect (  
    string $database,  
    string $username,  
    string $password  
    [, array $options ])
```

- db2_pconnect() offre un fonctionnement similaire, mais elle ouvre une connexion persistante (nous y reviendrons dans un instant).

Le site YoungiProfessionals.com fournit des analyses intéressantes sur les performances et possibilités des différents connecteurs DB2 disponibles :

<http://www.youngiprofessionals.com/wiki/index.php/PHP/DB2>

db2_connect() – les 3 paramètres obligatoires

1. \$srvdbase

- Laisser à blanc (") ou à '*LOCAL' pour une connexion à la base de données locale par défaut de votre serveur IBM i
- Si renseigné, utiliser un nom de base de données déclaré dans WRKRDBDIRE

2. \$username

- Laisser à blanc (") pour utiliser par défaut l'utilisateur Apache (QTMHHTTP)
- Dans le cas contraire, déclarer un profil IBM i valide.
- Si le profil est renseigné, le transmettre en majuscule par sécurité

3. \$password

- Laisser à blanc ("), si \$database est fixé à blanc (ou à '*LOCAL'), et si \$username est fixé à blanc
- Dans le cas contraire, fournir le mot de passe correspondant au profil \$username

■ Exemples :

- **Si les paramètres sont à blanc, des valeurs par défaut seront utilisées**
 - `$conn = db2_connect("", "", "");`
 - `$conn = db2_connect('*LOCAL', "", "");`
- **Définir ses propres paramètres de connexion :**
 - `$conn = db2_connect('MYDBSERVER', 'MYUSER', 'MYPASSWORD');`

db2_connect() – le 4^{ème} paramètre facultatif

- Le 4^{ème} paramètre de la fonction db2_connect() est un paramètre facultatif de type tableau (array). Certaines valeurs de ce tableau sont spécifiques à la plateforme IBM i. Il s'agit des paramètres suivants :
 - **i5_naming** : permet de choisir entre la syntaxe « système IBM i » ou la syntaxe purement SQL. (plus d'infos sur la diapo suivante)
 - **i5_lib** : pour définir une bibliothèque DB2 par défaut : tous les objets DB2 (tables, vue, procédures stockées, etc...) non qualifiés sont automatiquement recherchés dans cette bibliothèque. A utiliser uniquement si le paramètre "i5_naming" est fixé à "Off"
 - **i5_libl** : pour définir une liste de bibliothèque DB2 par défaut. Il doit être utilisé conjointement avec le paramètre « i5_naming », qui doit être fixé à « On » dans ce cas.
 - **i5_commit** : permet de travailler en mode « autocommit » (qui est le mode par défaut) ou de passer en « commitment control » manuel. (plus d'infos sur l'une des diapos suivante).

db2_connect() – options i5_naming et i5_libl

Si vous souhaitez travailler avec une liste de bibliothèques (ce qui est une particularité de DB2 pour IBMi), vous pouvez le définir grâce au paramètre “i5_naming” que vous devez fixer à “ON”

DB2_I5_NAMING_ON

- C’est une constante égale à 1
- Active le nommage Système IBM i
- Les tables “qualifiées” le sont en utilisant le délimiteur slash (/) entre noms de bibliothèques et de tables
- Les tables non qualifiées sont identifiées en s’appuyant sur la “library list” du travail, qui peut être définie via le paramètre “i5_libl” (à ne pas confondre avec “i5_lib” qui ne doit pas être utilisé dans cette configuration).

DB2_I5_NAMING_OFF

- C’est une constante égale à 0 (valeur par défaut)
- Active le nommage SQL
- Les tables “qualifiées” le sont en utilisant le délimiteur point (.) entre noms de bibliothèques et de tables
- Les tables non qualifiées sont identifiées en utilisant la bibliothèque par défaut paramétrée via le paramètre “i5_lib” s’il est défini (s’il n’est pas défini, c’est la bibliothèque par défaut du profil utilisateur qui est retenue).

db2_connect() – options i5_naming et i5_libl

- Attention : les paramètres « i5_lib » et « i5_libl » ne peuvent être utilisés conjointement.
- Le choix de travailler avec une liste de bibliothèques ou pas est un choix d'architecture, qui doit être fait en considérant l'architecture existante, et celle vers laquelle vous souhaitez tendre.
- On peut par exemple utiliser « i5_lib », donc une seule bibliothèque, et définir dans cette bibliothèque une série de vues DB2 qui pointeront vers des tables situées dans d'autres bibliothèques. Cela permet de réduire la « visibilité » de vos applications PHP à une seule bibliothèque ne contenant que les éléments que vous souhaitez leur mettre à disposition. **Il s'agit là d'une très bonne pratique d'un point de vue de la sécurité, et pas seulement pour les applications PHP.**

db2_connect() – option i5_commit

- L'option « i5_commit » permet de définir le type de « Commitment control » souhaité
- Fonctionne uniquement si le paramètre “ibm_db2.i5_allow_commit” est fixé à “allow_commit” dans la configuration du Zend Server
- Options possibles :

Commitment control mode	PDO	ibm_db2	SQLRPGLE (avec SET OPTIONS), et mot clé "CMT" dans le DSN	SQLRPGLE (avec WITH)
Commit immediate	0	DB2_I5_TXN_NO_COMMIT = 1	*NONE	NC
Read committed	2	DB2_I5_TXN_READ_COMMITTED = 3	*CS	CS (Cursor Stability)
Read uncommitted	1	DB2_I5_TXN_READ_UNCOMMITTED = 2	*CHG	UR (Uncommitted Read)
Repeatable read	3	DB2_I5_TXN_REPEATABLE_READ = 4	*ALL	RR (Repeatable Read)
Serializable	4	DB2_I5_TXN_SERIALIZABLE = 5	*RS	RS (Read Stability)

The screenshot shows the Zend Server Administration interface. The top navigation bar includes links for Monitor, Applications, Rule Management, Server Setup, and Administration. Below this, there are tabs for Components, Extensions, Directives, Debugger, Monitor, Job Queue, and 5250 Bridge. The main content area is titled 'View: Popular All ibm_db2 Search Save Changes'. Under the 'ibm_db2' section, several configuration options are listed:

- ibm_db2.binmode** - This option controls the mode used for converting to and from binary data in the PHP application. Value: DB2_BINARY
- ibm_db2.i5_all_pconnect** - This option overrides i5 db2_connect full open and close in the PHP application, all db2 connections become persistent. Value: full open and close
- ibm_db2.i5_allow_commit** - This option controls the commit mode used for i5 schema collections in the PHP application. Value: no commit (highlighted with a red arrow)
- ibm_db2.i5_dbscs_alloc** - This option controls the internal ibm_db2 allocation scheme for large DBCS column buffers. Value: no commit
- ibm_db2.i5_ignore_userid** - This option overrides i5 db2_(p)connect userid and password in the PHP application. Value: allow_commit
- ibm_db2.i5_job_sort** - This option turns DB2 UDB CLI job sort mode on/off. Value: db2_(p)connect with spec
- ibm_db2.i5_instance_name** - On Linux and UNIX operating systems, this option defines the name of the instance to use for cataloged database connections. Value: Turns off DB2 UDB CLI job sort mode

db2_connect() – option i5_commit

DB2_AUTOCOMMIT_ON

- Une constante égale à 1 (valeur par défaut)
- Active le mode « autocommit »
- Avantage : les insert, update et delete impactent immédiatement la base de données, sans nécessiter l'utilisation de la fonction db2_commit()

DB2_AUTOCOMMIT_OFF

- Une constante égale à zéro
- Désactive le mode « autocommit »
- Permet de contrôler les mises à jour de base de données en appliquant les fonctions db2_commit() / db2_rollback() sur des groupes de requêtes de mise à jour (insert/update/delete) .

Attention : si vous utilisez le « commitment » manuel (plutôt que l'autocommit), il est impératif d'effectuer un commit() explicite en fin de script PHP. Si vous oubliez ce point, toutes les opérations de mises à jour non « commitées » seront annulées par un rollback automatique à la fin du script courant.

db2_pconnect()

- La fonction `db2_pconnect()` permet d'établir une connexion persistante. Le principal avantage de cette solution est la rapidité d'exécution inhérente au fait de partager une connexion base de données avec d'autres travaux.
- La fonction `db2_pconnect()` retourne une ressource de connexion en cas de succès et «faux» en cas d'échec.
- Les options de `db2_pconnect` sont strictement les mêmes que celles de `db2_connect`.
- Avec `db2_pconnect()`, contrairement à `db2_connect()`, la connexion à DB2 UDB ne se ferme pas lorsque le script PHP se termine. De même la fonction `db2_close()` est sans effet sur une connexion ouverte avec `db2_pconnect()`. Seule la fonction `db2_pclose()` est en mesure de fermer une connexion persistante.
- Pour que DB2 utilise la même ressource pour plusieurs exécutions de `db2_pconnect`, il faut que les paramètres transmis à `db2_pconnect()` soient strictement identiques. On rappelle que ces paramètres sont : le nom serveur de base de données, le profil de connexion, et son mot de passe, plus les éventuelles options complémentaires. Il est donc recommandé d'utiliser le même script d'amorçage pour effectuer les différentes connexions (script utilisé via la fonction PHP « require », par exemple).

db2_pconnect

- Les connexions persistantes nécessitent une surveillance particulière, car dans ce mode les SELECT peuvent déclencher des pseudo verrouillages de type *SHRRD (shared read). Il est possible de les identifier dans les travaux QSQSRVR, via la commande WRKOBJLCK. Les pseudo verrouillages sont utilisés par DB2 pour optimiser les performances de certains types de curseurs. Il peut être nécessaire de les supprimer “manuellement” avant de pouvoir procéder à certaines opérations de maintenance (sauvegarde notamment).
- Les connexions persistantes s'exécutent toujours dans un “mode serveur”, qui peut se révéler complexe à administrer. Plusieurs articles ont été consacrés à ce sujet :

<http://www.redbooks.ibm.com/abstracts/tips0658.html>

<http://www.mcpressonline.com/tips-techniques/database/techtipgrab-control-of-the-db2-qsqsrvr-jobs.html>

<http://www.mcpressonline.com/database/db2/finding-sql-servermode-connecting-jobs.html>

<http://www.youngiprofessionals.com/wiki/FastCGI>

db2_connect et db2_connect

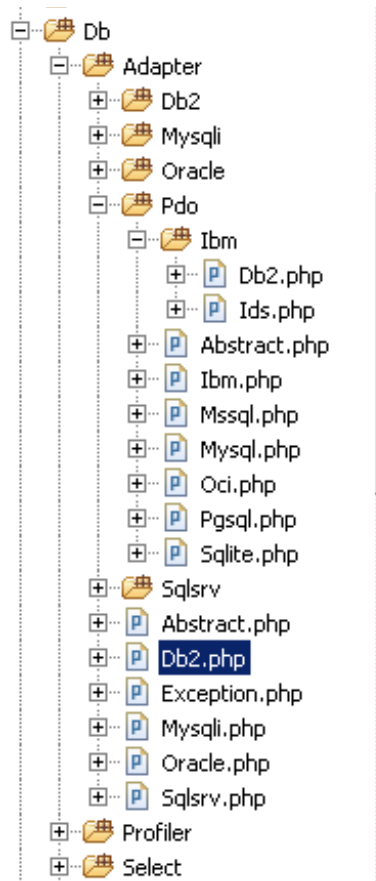
Quelques conseils pour l'architecture de vos applications :

- Pour éviter d'avoir un trop grand nombre de connexions bases de données ouvertes simultanément, évitez d'utiliser le profil de l'utilisateur connecté pour les opérations courantes (consultations et mises à jour de la base de données). Utilisez plutôt un profil générique (ou pas de profil du tout).
- Si vous voulez néanmoins vérifier que l'utilisateur est autorisé à se connecter via son profil utilisateur IBMi, effectuez une première connexion non persistante avec le profil de l'utilisateur pour vérifier qu'il est autorisé à entrer. Si l'utilisateur passe ce test avec succès, refermez sa connexion non persistante, et utilisez une connexion persistante couplée à un profil générique pour la suite des opérations.
- Chaque fois que vous n'avez plus besoin d'une requête, libérez les ressources qui lui sont affectées via la fonction `db2_free_result()`.
- N'utilisez qu'exceptionnellement la fonction `db2_exec()`, préférez-lui le couple de fonctions `db2_prepare()`, `db2_execute()`, et paramétrez vos requêtes en utilisant les points d'interrogation. Ne transmettez jamais les éléments variables de vos clauses WHERE par concaténation. Ce point sera abordé en détails dans un prochain chapitre.

Zend_DB

UN COMPOSANT EN DEVENIR

Zend_DB



- Zend_DB est l'un des quelques 70 composants du Zend_Framework (ZF) dans sa version 1.11.x.
- Il offre des connecteurs pour les principaux SGBD du marché (DB2, MySQL, etc...).
- Il est faiblement couplé avec les autres composants du ZF, ce qui signifie qu'il est facile de l'utiliser dans des projets PHP, indépendamment des autres composants du Zend Framework.
- L'étude de son code fournit un excellent support d'apprentissage du PHP, et permet de découvrir de nombreuses bonnes pratiques de codage (PHP et SQL).
- Il s'appuie sur la librairie « ibm_db2 » (donc db2_connect()) pour « attaquer » la base de données DB2 for i, quand il s'exécute sur serveur IBM i. Quand il s'exécute sur un serveur Windows ou Linux, il se sert du même connecteur (ou au choix de PDO) pour attaquer DB2 pour Windows/Linux. Pour information, cette détermination se fait au moyen du test suivant :

```
$this->_isI5 = (php_uname('s') == 'OS400') ? true : false;
```

Zend_DB (exemple de code fourni avec Zend Server)

```
require_once 'Zend/Db/Adapter/Db2.php';

$config = array(
    'dbname' => '*LOCAL',
    'username' => '',
    'password' => '',
    'os'=>'i5',
    'driver_options'=> array(
        'i5_commit' => DB2_I5_TXN_READ_UNCOMMITTED,
        'autocommit'=> DB2_AUTOCOMMIT_OFF,
        'i5_lib'=>'ZENDSVR'));

$conn = new Zend_Db_Adapter_Db2($config);

$sql = "SELECT * FROM ZENDSVR.SP_CUST WHERE CUST_ID > ?
        FOR FETCH ONLY";

echo '<table>';

$lower_limit = 1220; //from the CUST_ID value
$stmt = $conn->query($sql, $lower_limit); $i = 0 ;
while (($row = $stmt->fetch(Zend_Db::FETCH_ASSOC)) !==
    false) {
    if($i == 0){
        $ColumnNames = ShowTableHeader($row);
    }
    ShowTableRow($ColumnNames, $row );
    $i++;
}
echo '</table>';
```

```
function ShowTableHeader( $row ){
    $flds = count( $row );
    if ($flds > 0) {
        $fieldNames = array_keys($row);
        echo '<tr>';
        foreach ($fieldNames as $field) {
            echo "<td>{$field}</td>";
        }
        echo '</tr>';
    }
    return $fieldNames;
}

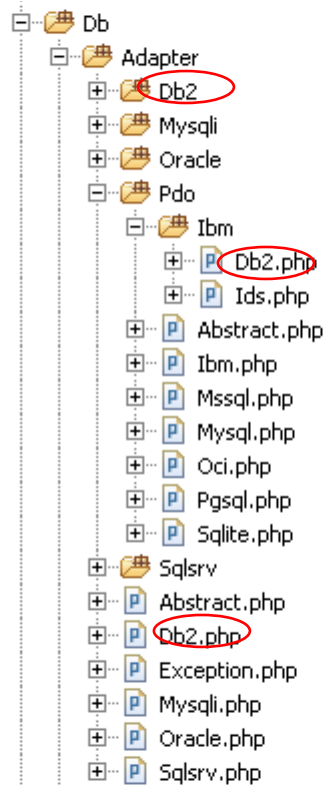
function ShowTableRow($ColumnNames , $row ){
    echo '<tr>';
    foreach ($ColumnNames as $ColumnName) {
        if(trim($row[$ColumnName]) === ''){
            echo "<td>&nbsp;</td>";
        }
        else {
            echo "<td>{$row[$ColumnName]}</td>";
        }
    }
    echo '</tr>';
}
```

Zend_DB

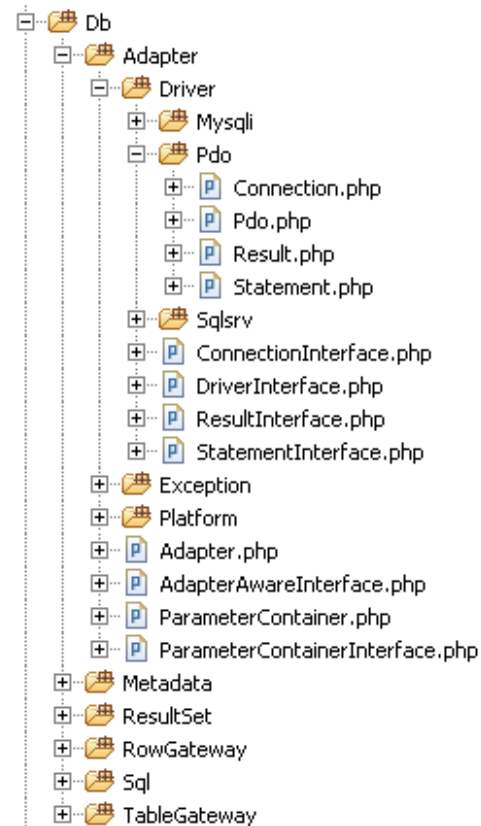
- C'est un composant puissant et la lecture de son code est très instructive, mais :
 - Ne proposant pas de connecteur adapté pour le « Iseries Access ODBC Driver », il ne permet pas d'attaquer une base DB2 for i à partir d'un Zend Server s'exécutant sous Windows. C'est dommage, car l'écriture d'un connecteur supplémentaire oblige à appliquer des patchs dans plusieurs classes du composant (travail laborieux).
 - L'implémentation de Zend_DB pour DB2 for i présente quelques lacunes, notamment dans l'utilisation de listes de bibliothèques, dans la gestion du caractère séparateur dans le cas où l'on veut utiliser la syntaxe système plutôt que SQL, ainsi que des problèmes de performance lorsque le composant extrait des métadonnées d'une table DB2. La plupart de ces lacunes peuvent être corrigées par l'application d'un patch proposé par Alan Seiden dans l'un de ses webinars:
<http://www.alanseiden.com/presentation%20slides/Your-first-Zend-Framework-Project-on-IBM-i.pdf>
 - Zend_DB est un composant en devenir qui fait partie des composants complètement réécrits pour le Zend Framework 2. La bêta du ZF 2 de mars 2012 laisse apparaître que le connecteur DB2 n'est pour l'instant pas implémenté (cf. diapo suivante), il est donc trop tôt pour évaluer ce composant, et mesurer l'impact d'une migration de ZF 1 vers ZF 2.

Zend_DB – évolution du composant Zend DB

ZF 1.11.11



ZF 2.0 Bêta 3 (mars 2012)



DB2 ?

<http://framework.zend.com/manual/fr/zend.db.html>

Zend_DB

- Zend_DB propose un jeu de classes permettant d'établir un pont entre le modèle relationnel de la base de données, et l'orientation Objet du langage PHP. Cette technique est couramment désignée par le terme d'ORM, pour « Object Relational Mapping ».
- Zend_DB n'est pas le seul composant à proposer l'ORM, d'autres projets PHP tels que Doctrine et Propel la proposent également. L'annonce récente de l'intégration de Doctrine dans le ZF 2 laisse planer un doute quant à l'avenir de l'approche ORM au sein même du composant Zend_DB. Là encore, des incertitudes planent sur l'orientation que prendra Zend_DB au sein de ZF 2.
- En ce qui concerne le projet Propel, il n'intègre pas de connecteur pour DB2. En revanche, Doctrine intègre un connecteur pour DB2 basé sur l'extension « ibm_db2 », connecteur qui ne prend pas en compte les spécificités de DB2 pour IBMi (syntaxe système, bibliothèques multiples).
- L'un des principaux reproches formulé à l'encontre des solutions de type ORM se situe au niveau des performances, qui ne sont pas toujours à la hauteur des attentes.
- Nous verrons un peu plus loin qu'il est relativement facile de développer sa propre solution ORM en appliquant les principes du design pattern « Active Record ».

Bonnes pratiques et
techniques avancées pour
DB2 ET PHP

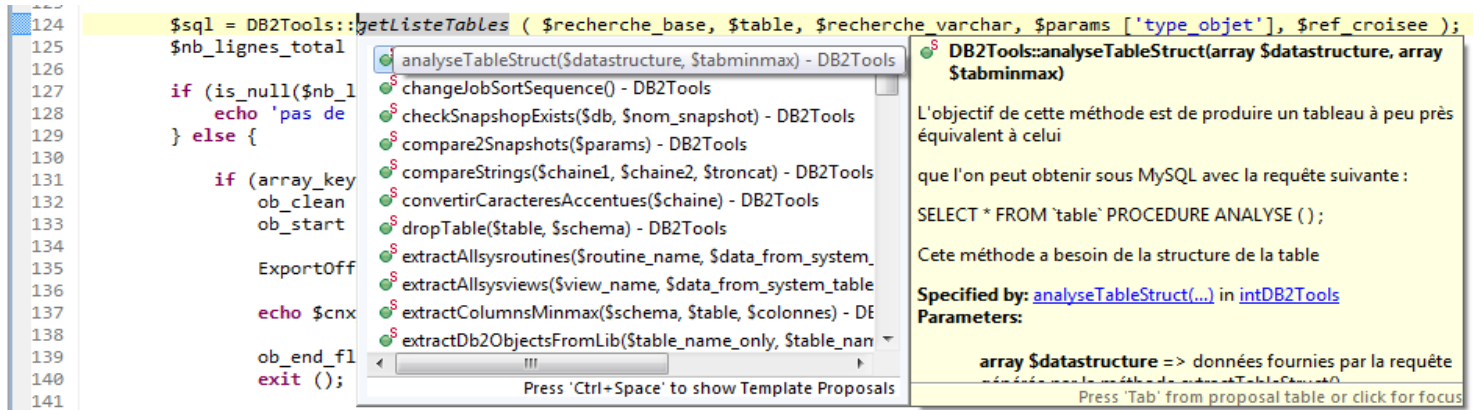
Au sommaire :

- Ne pas disséminer le code SQL
- Syntaxe système vs syntaxe SQL
- Requêtes paramétrées
- Pagination SQL et scroll cursor
- Créer son propre « wrapper »
- Exécution de commandes système IBM i via DB2 et PHP
- Données avec dates d'effet
- Procédures stockées DB2
- Verrouillage optimiste
- Design pattern Active Record
- C.R.U.D.

**NE PAS DISSEMINER
LE CODE SQL**

Ne pas disséminer le code SQL

- Une erreur couramment répandue dans les projets PHP consiste à disséminer le code SQL un peu partout dans le code des projets. Il en résulte une perte de visibilité, qui peut compliquer les analyses d'impact, et empêche de se constituer une base de connaissances réutilisable par l'ensemble de l'équipe de développement.
- Pour éviter cela, on peut créer des classes PHP abstraites qui contiendront une méthode par requête SQL. Le fait de procéder ainsi peut être perçu comme contraignant, mais il permet de se constituer une base de connaissances qu'il sera facile d'exploiter grâce au mécanisme d'auto-complétion fourni par Zend Studio :



SYNTAXE SYSTÈME CONTRE SYNTAXE SQL

Syntaxe Système contre syntaxe SQL

- Nous avons vu que DB2 for i offre la possibilité de travailler selon la syntaxe système ou la syntaxe SQL :
 - Syntaxe système : un slash sépare nom de bibliothèque et nom de table, et on a la possibilité de définir une liste de bibliothèques pour les objets DB2 non qualifiés
 - Syntaxe SQL : un point sépare nom de bibliothèque et nom de table, et on a la possibilité de définir une bibliothèque par défaut pour les objets DB2 non qualifiés
- Vous pouvez être tenté de travailler selon la syntaxe système, mais un responsable de la sécurité pourrait bien vous demander, un jour ou l'autre, de « swapper » vos applications sur la syntaxe SQL. Si vous devez pour ce faire repasser dans toutes vos scripts pour modifier les requêtes SQL... En appliquant la règle de « non dissémination » du code SQL vue précédemment, vous pouvez cependant circonscrire l'incendie. Mais vous pouvez faire mieux : en utilisant un caractère séparateur autre que le point et le slash dans vos requêtes, et en utilisant une méthode de classe (ou une fonction) qui fait automatiquement la conversion de ce caractère selon le type de syntaxe choisi, vous réglez définitivement le problème. Par exemple, dans mes projets, j'utilise le mot clé {SEPARATOR}, que je remplace automatiquement (via une fonction PHP) par un point ou un slash (selon le contexte) avant l'exécution de toute requête SQL.

REQUETES SQL PARAMÉTRÉES

Requêtes SQL paramétrées

- La technique dite des « requêtes SQL paramétrées » constitue le socle que tout développeur SQL doit connaître pour produire un développement de qualité. C'est une technique qui peut s'utiliser aussi bien en PHP qu'en RPG.
- Les requêtes SQL paramétrées offrent tous les avantages :
 - Sécurité : car elles protègent contre les attaques dites « par injection SQL »
 - Souplesse d'écriture : nul besoin avec cette technique de concaténer laborieusement les paramètres à l'intérieur des requêtes SQL
 - Performances : une requête paramétrée, si elle est exécutée plusieurs fois avec des valeurs différentes, va bénéficier du plan d'accès déterminé par le moteur SQL de DB2 lors de la première exécution de la requête. Les exécutions suivantes d'une même requête seront dès lors beaucoup plus rapides (le chemin d'accès n'étant calculé qu'une fois).
- La technique de l'attaque par injection SQL a été décrite dans de nombreux articles disponibles sur internet. Nous ne l'approfondirons pas ici, faute de temps, aussi je vous recommande la lecture des articles suivants :
 - Wikipédia : http://fr.wikipedia.org/wiki/Injection_SQL
 - PHP Security Consortium : <http://phpsec.org/library/>

Requêtes SQL paramétrées

Avant de voir ce qu'est une requête SQL paramétrée, étudions son antithèse, avec l'exemple ci-dessous que l'on peut qualifier de « très mauvaise pratique » :

```
$conn = db2_connect("LOCAL", "", "");  
if (!$conn) {  
    echo "Echec de la connexion. SQL Err:", db2_conn_error(), "<br />" . db2_conn_errormsg();  
    exit();  
}  
$sql = "SELECT CODE, NOM, POIDS FROM BIBPROD.TABPROD WHERE CODE = '" . $_REQUEST["code"] . "'";  
$stmt = db2_prepare($conn, $sql);  
if (db2_execute ($stmt)) {  
    while ( $row = db2_fetch_assoc ( $stmt ) ) {  
        print "{$row['NOM']}, {$row['RACE']}, {$row['POIDS']}<br />" . PHP_EOL;  
    }  
} else {  
    echo db2_stmt_error ($stmt), ' ', db2_stmt_errormsg ($stmt);  
}
```

La variable `$_REQUEST` contient – peut être, ou peut être pas – un poste de tableau « code », que l'on colle directement dans la requête sans filtrage d'aucune sorte. C'est une grave faille de sécurité, qui ouvre la porte aux « attaques par injection SQL ».

Requêtes SQL paramétrées

Il suffisait pourtant de pas grand-chose pour que la requête soit sécurisée, démonstration :

```
// ... pour gagner de la place, j'ai retiré le code définissant la connexion DB2
if (array_key_exists("code", $_REQUEST) && is_string($_REQUEST["code"])) {
    $code = trim($_REQUEST["code"]) ; // suppression des blancs devant et derrière
    if (strlen ($code) > 20 ) {
        $code = substr($code , 0 , 20) ; // si on suppose que la colonne CODE fait 20 caractères...
    }
} else {
    $code = "";
}

$sql = "SELECT CODE, NOM, POIDS FROM BIBPROD.TABPROD WHERE CODE = ? " ;
$stmt = db2_prepare($conn, $sql);

if (db2_execute ( $stmt, array($code) )) {
    while ( $row = db2_fetch_assoc ( $stmt ) ) {
        print "{$row['NOM']}, {$row['RACE']}, {$row['POIDS']}<br />" . PHP_EOL;
    }
} else {
    echo db2_stmt_error ($stmt), ' ', db2_stmt_errormsg ($stmt);
}
```

Les explications sont données sur la diapo suivante.

Requêtes SQL paramétrées

- Dans le code de la diapositive précédente, nous avons fait l'effort d'appliquer quelques règles de filtrage simples sur le contenu de la variable `$_REQUEST`. Ce n'est pas suffisant pour protéger contre tous les types d'attaques, mais c'est mieux que de ne rien faire.
- La valeur récupérée dans `$_REQUEST` n'est pas concaténée avec la requête SQL. Elle est stockée dans une variable, après avoir vérifié qu'elle était bien dans un format compatible avec celui de la colonne CODE (en l'occurrence il s'agit d'un type « string »). On limite aussi la taille de la variable PHP à la longueur maximale de la colonne CODE car certaines versions de DB2 ne tolèrent pas les dépassements de capacité.
- La variable `$code` est transmise à SQL via la fonction `db2_execute()`, c'est donc DB2 qui se chargera de la combiner avec la requête en appliquant ses puissantes fonctions de parsing et de contrôle de validité. L'attaque par injection SQL est dès lors impossible.
- A noter : en ce qui me concerne, je travaille rarement avec la variable globale `$_REQUEST`, je préfère travailler avec `$_POST` et `$_GET`, car je considère que je dois maîtriser en permanence la manière dont les paramètres sont transmis à mon script PHP. Mais on touche là à des notions d'architecture applicative, qui dépassent le cadre de notre sujet du jour, aussi ai-je utilisé `$_REQUEST` pour simplifier l'exemple.
- L'imagination des pirates informatiques est sans limites, l'attaque par injection SQL n'est qu'un des nombreux types d'attaque possibles, mais c'est l'une des techniques les plus faciles à contrer, si l'on applique la technique qui a été présentée ici.


Requêtes SQL paramétrées

- Les requêtes paramétrées fonctionnent sur le même principe avec PDO, et elles s'appliquent très bien à tous les types de requêtes. Exemple ci-dessous avec la recopie des postes d'un tableau \$liste dans une table DB2, via une requête de type INSERT exécutée avec PDO :

```
$nom_table = "mabib.matable" ;
$nb_lignes_creees = 0 ;
try {
    $query = "insert into {$nom_table} (code, libelle) values (?, ?) " ;
    $st = $db->prepare($query); // préparation de la requête SQL avant la boucle
    foreach( $liste as $key=>$value ) {
        $st->execute(array($key, $value) ) ; // exécution de la requête avec les valeurs extraites de chaque occurrence de $liste
        $nb_lignes_creees++;
    }
    unset($st);
} catch (PDOException $e) {
    // gestion d'erreur
}

echo "nombre de lignes créées => " . $nb_lignes_creees . "<br/>" . PHP_EOL ;
```

```
echo 'Error : ' . $e->getMessage() . "<br/>" . PHP_EOL;
echo 'Code : ' . $e->getCode() . "<br/>" . PHP_EOL;
echo 'File : ' . $e->getFile() . "<br/>" . PHP_EOL;
echo 'Line : ' . $e->getLine() . "<br/>" . PHP_EOL;
echo 'Trace : ' . $e->getTraceAsString() . "<br/>" . PHP_EOL;
```



Requêtes SQL paramétrées

- Pour conclure sur ce sujet :
- N'utilisez qu'exceptionnellement la fonction `db2_exec()`, préférez-lui le couple de fonctions `db2_prepare()`, `db2_execute()`, et paramétrez vos requêtes en utilisant les points d'interrogation. Ne transmettez jamais les éléments variables de vos clauses `WHERE` par concaténation. La remarque vaut également si vous travaillez avec PDO, ou tout autre connecteur base de données.
- Si vous décidez de sélectionner une application open-source pour développer votre activité, effectuez systématiquement un audit de code. Ne laissez pas des failles de sécurité mettre en péril votre activité. Si vous détectez des failles de sécurité de type « injection SQL » dans une application que vous envisagiez d'utiliser, vous avez 2 options :
 1. Soit vous rejetez l'application purement et simplement,
 2. Soit vous considérez que cette application est vitale pour votre activité, et alors vous n'avez pas d'autre choix que de la corriger, ou d'aider l'équipe à l'origine du projet pour qu'elle fasse les corrections (par un soutien logistique et/ou financier).

LISTES AVEC PAGINATION

Listes avec pagination

Structure des tables et vues DB2

Saisissez une bibliothèque (facultatif):

Saisissez une table SQL (facultatif):

Types d'objets:

Schéma	Table (sqlname)	Table (sysname)	Type	Nb.Cols.	Buffer	Description	Propriétaire
QSYS2	CATALOG_NAME	SYSCAT2	V	1	130		QSYS
QSYS2	CHARACTER_SETS	SYSCHRSET1	V	8	914		QSYS
QSYS2	CHARACTER_SETS_S	SYSCHRSET2	V	8	914		QSYS
QSYS2	INFORMATION_SCHEMA_CATALOG_NAME	SYSCAT1	V	1	130		QSYS
QSYS2	SQL_FEATURES	SYSFEATURE	T	7	2543		QSYS
QSYS2	SQL_LANGUAGES	SYSLANGS	T	7	1792		QSYS
QSYS2	SYSCATALOGS	LOCATIONS	V	6	111		QSYS
QSYS2	SYSCCHARSETS	SYSCCHARSET	T	4	520		QSYS
QSYS2	SYSCCHKCST	SYSCCHKCST	V	3	2262		QSYS
QSYS2	SYSCOLUMNS	SYSCOLUMNS	V	36	5076		QSYS

<< Préc. | 1-10 | 11-20 | 21-30 | 31-40 | 41-46 | Suiv. >>
(Affichage 1 à 10 sur 46)

Exemple de liste avec pagination

L'écran ci-dessus est un exemple de pagination effectuée sur les tables systèmes DB2. Cet affichage peut s'effectuer en utilisant une technique « full SQL » ou une technique de type « scroll cursor ». Nous allons étudier ces deux techniques dans les diapos suivantes.

Listes avec pagination

- **ATTENTION** : le code suivant ne permet pas de gérer une pagination

```
SELECT * FROM table FETCH FIRST 10 ROWS ONLY
```

Le code SQL ci-dessus renvoie les 10 premières lignes du jeu de données identifié par le SGBD. Cette technique est totalement inadaptée à la gestion de listes avec pagination.

- Avec MySQL, vous pouvez gérer facilement une pagination au moyen du code SQL suivant :

```
SELECT * FROM table LIMIT 10, 20
```

(renvoie les lignes 10 à 20 du jeu de données identifié par le SGBD)

- Avec DB2, c'est un peu plus compliqué.

Listes avec pagination

- Les données DB2 « chargées » dans l'avant dernière diapositive sont chargées via la requête SQL suivante :

```
SELECT A.*  
FROM QSYS2/SYSTABLES A  
WHERE A.TABLE_SCHEMA = ? AND (A.TABLE_NAME LIKE ? OR A.SYSTEM_TABLE_NAME LIKE ?)
```

- Cette requête ne peut gérer à elle seule la pagination, qui est effectuée ici par pages de 10 lignes. Pour gérer la pagination avec une technique « full SQL », la requête ci-dessus doit être légèrement modifiée, et encapsulée dans une autre requête :

```
SELECT foo.* FROM (  
    SELECT row_number() over (ORDER BY TABLE_NAME) as rn,  
    A.*  
    FROM QSYS2/SYSTABLES A  
    WHERE A.TABLE_SCHEMA = ? AND (A.TABLE_NAME LIKE ? OR A.SYSTEM_TABLE_NAME LIKE ?)  
) AS foo  
WHERE foo.rn BETWEEN ? AND ?
```

La clause « order by »,
si vous en voulez une,
doit être placée ici

Listes avec pagination

- La technique « full SQL » présentée dans la diapo précédente donne de bons résultats sur des tables de taille raisonnable (difficile de donner un chiffre précis car cela dépend beaucoup de la puissance du (des) processeurs(s) de votre serveur IBM i). Mais elle présente quelques défauts :
 - Elle est « intrusive » dans le sens où elle nécessite de modifier la requête SQL pour y insérer un certain nombre d'éléments (modification du début du SELECT, inclusion du tri dans la clause OVER...).
 - Avec cette technique, DB2 a tendance à s'effondrer sur les tables de grande taille, donc si vous rencontrez des difficultés avec cette technique, je vous recommande de recourir à la technique du curseur scrollable qui donne de bons résultats dans la plupart des cas.
- On peut noter que cette technique est utilisée par le composant Zend_DB du Zend Framework pour la pagination avec DB2, d'une manière moins « intrusive » que celle présentée dans la diapo précédente, mais qui présente de grosses lacunes en termes de performances :

```
$limit_sql = "SELECT z2.*  
FROM (  
SELECT ROW_NUMBER() OVER() AS \"ZEND_DB_ROWNUM\", z1.*  
FROM (  
" . $votre_requete_sql_ici . "  
) z1  
) z2  
WHERE z2.zend_db_rownum BETWEEN " . ($offset+1) . " AND " . ($offset+$count);
```

Tri incorrect, car il est impossible d'insérer les colonnes à trier dans la clause OVER

La pagination s'applique **sur le jeu de données renvoyé par votre requête** encapsulée ici. Elle ne s'applique pas directement sur votre requête, donc attention aux perfs sur les tables de grande taille !!!

Listes avec pagination

- La technique du curseur scrollable, qui est plus performante sur les tables de grande taille, est relativement « verbeuse ». Je vous en donne un échantillon sur la diapo suivante, mais pour des exemples plus « détaillés », je vous recommande de vous reporter aux exemples de code que j'ai déposés sur les pages suivantes :
 - Technique du curseur scrollable pour la librairie « ibm_db2 » :
 - <http://fr.php.net/manual/fr/function.db2-prepare.php>
 - Technique du curseur scrollable pour PDO :
 - <http://fr2.php.net/manual/fr/pdo.prepare.php>
- En plus d'être performante sur les tables de grande taille, la technique du curseur scrollable présente l'avantage de ne pas être intrusive par rapport au code de la requête SQL à exécuter (votre requête SQL ne subit aucune modification).
- A noter : la technique du curseur scrollable fonctionne très bien sous PDO avec les bases DB2 for i et DB2 Express C. Mais j'ai détecté un bug récemment, peut être lié à ma version d'ODBC pour Windows. Après pas mal de tests, j'ai trouvé une solution de contournement que j'ai indiquée sur php.net, dans la page consacrée à PDO Prepare (cf. lien ci-dessus). Le principe consiste à effectuer un premier positionnement sur l'offset zéro, quand on est sur l'affichage d'offset de valeur supérieure à zéro.

Listes avec pagination

Exemple de code pour l'utilisation de curseur scrollable avec db2_connect() :

```
$st = db2_prepare ( $db, $sql, array('cursor' => DB2_SCROLLABLE) );  
$rows = array() ; // initialisation du tableau servant à stocker les données extraites  
if ( $st ) {  
    if (db2_execute ( $st, $args )) {  
        for (      $tofetch = $nbl_by_page,  
                    $row = db2_fetch_assoc ( $st, $offset);  
                    $row !== false && $tofetch-- > 0;  
                    $row = db2_fetch_assoc ( $st )  
                ){  
            $rows [] = $row;  
        }  
    }  
    db2_free_stmt($st);  
}
```

A noter : j'ai posté un exemple plus complet de code sur php.net, dans la page consacrée à DB2 Prepare (cf. lien indiqué sur la diapo précédente). Attention, prenez bien l'exemple le plus récent, le précédent contenant un bogue.

LE PROJET MACARONDB

Le projet MacaronDB

- Le terme « wrapper » désigne en anglais un « emballage ».
- Il est intéressant de disposer d'un jeu de fonctions permettant d'alléger le code PHP de vos projets. Le « wrapper » va avoir pour rôle de masquer la mécanique d'exécution de vos requêtes SQL. Cet outil peut se présenter sous la forme d'une classe embarquant un jeu de méthodes telles que celles que j'ai définies dans l'interface suivante :

```
function selectOne($db, $sql, $args = array(), $fetch_mode_num = false);
function selectBlock($db, $sql, $args = array());
function selectKeyValuePairs($db, $sql, $args = array());
function executeCommand($db, $sql, $args = array(), $count_nb_rows = true);
function executeSysCommand($cmd);
function callProcedure($db, $proc_name, $proc_schema, &$args = array(), $return_resultset = false);
function getStatement($db, $sql, $args = array());
function getFetchAssoc($st);
function getPagination($db, $sql, $args, $offset, $nbl_by_page, $order_by = '');
function getScrollCursor($db, $sql, $args, $offset, $nbl_by_page, $order_by = '' );
function export2CSV($db, $sql, $args = array());
function export2XML($db, $sql, $args = array(), $tag_line = '', $gen_header=true);
function export2insertSQL($db, $sql, $args = array());
function getLastInsertId($db, $sequence = '');
function valueIsExisting($db, $table, $nomcol, $valcol, $where_optionnel = '');
function valueIsExistingOnOtherRecord($db, $table, $nomcol, $valcol, $idencours, $where_optionnel = '');
function getInfoDatabase($db);
function countNbRowsFromTable($db, $table, $schema = '');
function countNbRowsFromSQL($db, $sql, $args = array());
function MyDBException($db, $DBexc, $sql = '', $args = array());
```

Le projet MacaronDB

Exemple d'utilisation avec la méthode `selectOne()` qui est dédiée à la récupération d'une ligne de table (ou vue) et à la restitution de l'information sous la forme d'un tableau PHP (à une dimension) :

Sans MacaronDB

```
$sql = 'votre requête SQL ici' ;
$args = 'tableau des arguments à incorporer dans la
        clause where';
$stmt = db2_prepare ( $db, $sql );
if (! $st) {
    self::MyDBError ( $db, 'selectOne/db2_prepare',
        $sql, $args );
    $data = null;
} else {
    if (! db2_execute ( $st, $args )) {
        self::MyDBError ( $db, 'selectOne/db2_execute',
            $sql, $args );
        $data = null;
    } else {
        $data = db2_fetch_assoc ( $st );
    }
}
db2_free_stmt($st);
```

Avec MacaronDB

```
$sql = 'votre requête SQL ici' ;
$args = 'tableau des arguments à incorporer dans la
        clause where';
$data = $db->selectOne ( $sql, $args);
```

A noter : la méthode `selectBlock()` fonctionne sur le même principe, mais elle est dédiée à la récupération de plusieurs lignes, et renvoie donc un tableau à 2 dimensions.

Le projet MacaronDB

- Les principaux avantages fournis par un projet comme MacaronDB sont :
 - La réduction du nombre de lignes de code à écrire,
 - L'obtention d'un code plus homogène qui contribue à assurer une bonne maintenabilité
 - La possibilité de disposer d'une librairie open-source, librement utilisable et personnalisable selon vos besoins, qui a été développée et est maintenue par des spécialistes de DB2 et de la plateforme IBM i.
 - Une empreinte mémoire faible, du fait du petit nombre de classes en jeu.
- Pour l'instant MacaronDB contient les connecteurs BD suivants :
 - DB2 for i, en utilisant l'extension « `ibm_db2` », pour une utilisation sur Zend Server tournant sur IBM i
 - DB2 for i, en utilisant l'extension PDO et le « Iseries Access ODBC Driver », pour une utilisation à partir d'un Zend Server pour Windows (non testé sur Linux)
 - DB2 Express C, en utilisant l'extension PDO et le « IBM DB2 ODBC DRIVER », pour une utilisation sur Zend Server pour Windows (non testé sur Linux)
 - MySQL 5, en utilisant l'extension `PDO_MySQL`.

(d'autres connecteurs bases de données devraient voir le jour prochainement).

Le projet MacaronDB

MacaronDB est un projet sous licence libre « New BSD ».

Il est téléchargeable librement à partir de ce dépôt Github :

<https://github.com/gregja/macarondb>

EXECUTION DE COMMANDES SYSTÈME AVEC DB2

Exécution de commandes système IBM i avec DB2

- Depuis la V5R4, vous pouvez lancer des commandes systèmes IBM i en utilisant DB2 (et donc PHP).
- Par exemple, si vous souhaitez récupérer dans une table DB2 la liste des profils utilisateurs IBM i, vous pouvez procéder ainsi :

```
function cmdsys_rtvusrprf_all () {  
    $cmd = 'DSPUSRPRF USRPRF(*ALL) OUTPUT(*OUTFILE) OUTFILE(QTEMP/TMPPROFILE) OUTMBR(*FIRST  
    *REPLACE) ' ;  
    $cmd_length = strlen($cmd);  
    return "CALL QCMDEXC ('{$cmd}', {$cmd_length})" ;  
}  
$cmd_sys = cmdsys_rtvusrprf_all();  
echo $cmd_sys , '<br />' . PHP_EOL;  
$result = db2_exec($conn, $cmd_sys);  
if ($result) {  
    print "La table a été créée correctement.<br />" . PHP_EOL;  
} else {  
    print "La table n'a pas été créée.<br />" . PHP_EOL;  
}
```

Exécution de commandes système IBM i avec DB2

- Cette manière d'exécuter des commandes systèmes IBM i fonctionne aussi bien avec `db2_connect()` qu'avec PDO, et elle ne nécessite l'emploi d'aucune boîte à outils complémentaires
- Autre exemple d'utilisation possible, vous voulez détecter des déphasages entre des environnements de recette, de préproduction et de production... simplement en récupérant la liste de objets IBMi et/ou DB2 des bibliothèques de chaque environnement (via la technique vue dans la diapo précédente), puis en les comparant via un bon vieil algorithme de « matching », comme dans l'exemple ci-dessous :

Liste des écarts constatés entre les bibliothèques : I515/BIBREF1 et I515/BIBREF2

Nbre d'objets identiques : 40

Nbre d'objets en écart : 2

N°	REF1 = I515/BIBREF1							REF2 = I515/BIBREF2							Cause
	Nom table REF1	Nom court	Attr.	Type	Nb.Fields	Nb.Key Fields	Record Length	Nom table REF2	Nom court	Attr.	Type	Nb.Fields	Nb.Key Fields	Record Length	
0	BCHCHILD01	BCHCHILD01	IX	D	4	1	22								non trouvé sur REF2
1	BCHCHILD02	BCHCHILD02	IX	D	4	1	22								non trouvé sur REF2

Exécution de commandes système IBM i avec DB2

La méthode utilisée pour extraire les objets IBM i d'une bibliothèque est la suivante:

```
function extract_ibmi_objects_from_lib($library, $tmp_table) {  
    $tmp_table = strtoupper(trim($tmp_table)) ;  
    $library = strtoupper(trim($library));  
    $cmd = 'DSPOBJD OBJ('.$library.'/*ALL) OBJTYPE(*ALL) DETAIL(*BASIC) OUTPUT(*OUTFILE)  
    OUTFILE(QTEMP/'.$tmp_table.') OUTMBR(*FIRST *REPLACE) ' ;  
    $cmd_length = strlen($cmd);  
    return "CALL QCMDEXC ('{$cmd}', {$cmd_length})" ;  
}
```

On peut dès lors exécuter cette commande via DB2 sur 2 bibliothèques différentes d'un même serveur IBM i, ou encore sur 2 bibliothèques de 2 serveurs IBM i différents. Il ne reste plus qu'à comparer les 2 jeux de données en les parcourant séquentiellement et parallèlement, pour détecter les écarts (technique de « matching »).

A noter : la méthode `executeSysCommand()` du projet MacaronDB met en œuvre tout ce que je viens de décrire.

Exécution de commandes système IBM i avec DB2

Attention, la comparaison du contenu de 2 bibliothèques par la technique du « matching » réserve quelques surprises si on n'y prend pas garde, du fait de la manière dont PHP et DB2 for i trient alphabétiquement les chaînes de caractères:

tri DB2 for i	tri PHP et Windows
P0AAA00	P00AA00
P00AA00	P0AAA00

Une différence aussi minime rend inopérante la technique du « matching » par comparaison alphabétique, sauf si vous laissez à DB2 l'entière responsabilité de déterminer si les données sont bien triées ou pas. Donc, plutôt que de comparer les chaînes de caractères via PHP, créez une fonction PHP qui recevra les chaînes de caractères 1 et 2 à comparer. A l'intérieur de cette fonction, tronquez les 2 chaînes à comparer à la longueur de la plus courte des deux, et exécutez le code SQL ci-dessous via votre « wrapper » préféré :

```
$sql = <<<BLOC_SQL
SELECT CASE WHEN '{$chaine1}' > '{$chaine2}' THEN 1 ELSE
          CASE WHEN '{$chaine1}' < '{$chaine2}' THEN -1 ELSE 0 END
        END as RESULT
FROM SYSIBM.SYSDUMMY1
BLOC_SQL;
```

Exécution de commandes système IBM i avec DB2

- Note de dernière minute : la technique présentée ici pour l'exécution de commandes système est très pratique mais elle présente un petit inconvénient qui est que rien ne permet à DB2 de savoir si la commande système s'est bien exécutée ou pas. Donc, dans les exemples qui précèdent, si le DSPOBJD ou le DSPUSRPRF se sont mal passés, vous ne vous en rendrez compte que lorsque vous essaierez d'accéder à la table temporaire, via un SELECT SQL par exemple.

GESTION DES DATES D'EFFET

Gestion des dates d'effet

- La récupération de données soumises à date d'effet n'est pas une technique nouvelle, pourtant elle est méconnue et peu utilisée.
- La technique fonctionne pratiquement de la même façon avec DB2 et avec MySQL.
- Cette technique entre dans la catégorie des sous-requêtes dites « scalaires ». Exemple de sous-requête scalaire permettant de récupérer dans une table TABLIB un libellé correspondant à un code qui lui est stocké dans la table TABCOD (exemple simplifié dans lequel les 2 tables ont les mêmes ID) :

```
SELECT
    A.ID,
    (SELECT B.LIBELLE FROM TABLIB B WHERE A.ID = B.ID) AS LIBELLE
FROM TABCOD A
```

Gestion des dates d'effet

- Supposons maintenant que vous souhaitiez récupérer, en plus du libellé, le prix de vente stocké dans une table TABPRX, ayant pour clé l'identifiant du produit (colonne ID) et une date d'effet (colonne DATEFF), vous pouvez écrire ceci :

```
SELECT
  A.ID,
  (SELECT B.LIBELLE FROM TABLIB B WHERE A.ID = B.ID) AS LIBELLE
  (SELECT C.PXVENT FROM TABPRX C
   WHERE C.ID = A.ID
        AND C.DATEFF = ( SELECT MAX(D.DATEFF) FROM TABPRX D
                        WHERE D.ID = A.ID AND D.DATEFF <= CURRENT_DATE )
  ) AS PRIX_VENTE
FROM TABCOD A
```

J'avais consacré plusieurs articles à la problématique des sous-requêtes scalaires et des dates d'effet sur le site Foothing.net, je vous invite à les lire si vous estimez avoir besoin de mieux maîtriser ces techniques :

[Traiter des données soumises à date d'effet, avec et sans SQL. - FOOTHING](#)

Gestion des dates d'effet

Note de dernière minute : Christian Massé (Volubis), nous a expliqué dans sa présentation sur les requêtes récursives que de nouvelles techniques étaient apparues récemment pour gérer notamment les données soumises à date d'effet. Je vous invite à lire le support de sa présentation, en particulier le chapitre consacré aux CTE (Common Table Expressions).

PROCEDURES STOCKEES

Les Procédures stockées

- Les procédures stockées DB2 se classent en 2 catégories :
 - Les procédures « full SQL » écrites en PL/SQL
 - Les procédures « externes », servant d'intermédiaire pour l'appel de programme écrits en langage RPG, CL, Cobol, Java...
- Nous n'évoquerons pas plus avant les procédures « full SQL », ce n'est pas l'envie qui me manque (car j'en pense beaucoup de bien), c'est le temps, malheureusement ☺.
- Nous allons en revanche nous attarder sur les procédures stockées externes, car elles constituent la voie royale pour rendre accessibles aux applications web, vos composants « métier », souvent riches et complexes écrits en RPG ou Cobol.
- La documentation de PHP propose des exemples d'utilisation assez complets :
 - Pour « ibm_db2 » : <http://fr.php.net/manual/fr/function.db2-bind-param.php>
 - Pour PDO : <http://fr.php.net/manual/fr/pdo.prepared-statements.php>
- Comme point de départ, il nous faut un programme RPG, alors en voici un (cf. diapo suivante)

Les Procédures stockées

```
*****
* DESCRIPTION DSPPGMREF d'un pgm reçu en paramètre
*
* NOM DU PROGRAMME PGMREFPRC
*****
h usrprf(*owner) datfmt(*iso)
d Requete          s          300
d Commande         s          200
d PGMREF           PR          Extpgm('QCMDEXC')
d String           1000      Const
d                  Options(*Varsize)
d Len              15P 5 Const
```

Ce programme reçoit un nom de programme en paramètre, effectue un DSPPGMREF sur ce programme pour générer une table temporaire, puis exploite cette table temporaire via SQL pour en extraire un jeu de données qu'il va renvoyer en sortie, sous la forme d'un « result set » directement exploitable par la procédure stockée appelante. On pouvait faire plus simple, mais le mélange des techniques est ici intéressant d'un point de vue pédagogique.

```
/free

Commande = 'DSPPGMREF PGM(' + %trim(CODBIB) + '/' +
%trim(CODPGM)
+ ') OUTPUT(*OUTFILE) OBJTYPE(*ALL) '

+ ' OUTFILE(QTEMP/PGMREF1) OUTMBR(*FIRST *REPLACE) ' ;

CALLP(E) PGMREF(Commande:%len(Commande));

if not(%error()) ;

Requete = 'SELECT WHFNAM, WHLNAM, WHSNAM, WHRFNO, '
+ ' WHFUSG, WHRFNM, WHRFSN, WHRFFN, WHOBJT '
+ ' FROM QTEMP/PGMREF1 WHERE WHOBJT = 'F'' ;

EXEC SQL

PREPARE REQ1 FROM :Requete ;

EXEC SQL

DECLARE C1 CURSOR FOR REQ1 ;

EXEC SQL

OPEN C1 ;
```

Les Procédures stockées

- Maintenant que nous avons notre programme RPG, nous pouvons l'encapsuler dans une procédure stockées externe :

```
CREATE PROCEDURE GJABASE/PGMREFPRCS (  
  IN CODBIB CHAR(10) ,  
  IN CODPGM CHAR(10) )  
  DYNAMIC RESULT SETS 1  
  LANGUAGE RPGLE  
  SPECIFIC GJABASE/PGMREFPRCS  
  NOT DETERMINISTIC  
  MODIFIES SQL DATA  
  CALLED ON NULL INPUT  
  EXTERNAL NAME 'GJABASE/PGMREFPRC'  
  PARAMETER STYLE SQL ;  
  
COMMENT ON SPECIFIC PROCEDURE GJABASE/PGMREFPRCS IS 'Appel pgm PGMREFPRC' ;
```

A noter : le logiciel « System i Navigator » d'IBM fournit d'excellents assistants pour créer des procédures stockées « full SQL » ainsi que des procédures stockées externes telle que celle ci-dessus.

Les Procédures stockées

- Il ne nous reste plus qu'à écrire un peu de code PHP pour invoquer la procédure stockée DB2, et afficher le contenu du result set renvoyé par cette procédure :

```
$sql = 'CALL GJABASE.PGMREFPRCS (?, ?)';  
$stmt = db2_prepare ( $conn, $sql );  
$cod_pgm = 'mon_prog';  
$cod_bib = 'ma_bib.';  
db2_bind_param ( $stmt, 1, "cod_bib", DB2_PARAM_IN );  
db2_bind_param ( $stmt, 2, "cod_pgm", DB2_PARAM_IN );  
if (db2_execute ( $stmt )) {  
    while ( $row = db2_fetch_array ( $stmt ) ) {  
        print " {$row[0]}, {$row[1]}, {$row[2]} <br />".PHP_EOL;  
    }  
}
```

D'autres exemples d'utilisation sont fournis dans la documentation officielle :

<http://fr2.php.net/manual/fr/function.db2-bind-param.php>

Les Procédures stockées

Un aspect déplaisant de la fonction `db2_bind_param()` réside dans l'obligation qui est faite de transmettre comme 3^{ème} paramètre un nom de variable, et non pas la variable elle-même (obligation qui n'existe pas avec PDO). Cela complique nettement l'écriture d'un composant générique de type « wrapper ». Si vous êtes confronté à ce problème, je vous invite à étudier le code source de la classe `DB2_IBMi_DBWrapper` intégrée au projet MacaronDB, et particulièrement la méthode `callProcedure()` qui propose une manière de contourner le problème (cf. extrait de code ci-dessous) :

```
$args_inc = 0 ;
foreach ($args as $key=>$arg) {
    // crée artificiellement une variable ayant pour nom le contenu de la variable $key
    $$key = $args[$key]['value'] ;
    $args_inc++ ;
    $arg['type'] = strtolower($arg['type']);
    switch ($arg['type']) {
        case 'out': {
            db2_bind_param($st, $args_inc, $key, DB2_PARAM_OUT);
            break;
        }
        case 'inout': {
            db2_bind_param($st, $args_inc, $key, DB2_PARAM_INOUT);
            break;
        }
        default:{
            db2_bind_param($st, $args_inc, $key, DB2_PARAM_IN);
            break;
        }
    }
}
```

GESTION DES VERROUILLAGES

Gestion des verrouillages

- Dans le cadre d'un développement en RPG, il est facile de mettre en œuvre la technique du verrouillage physique des enregistrements bases de données, en utilisant les ordres de lecture de base de données natifs.
- Dans le cadre d'un développement utilisant les techniques du web, il est impossible de mettre en œuvre un verrouillage physique des enregistrements bases de données, du fait notamment que les transactions HTTP fonctionnent en mode « stateless » (sans état).
- Il existe une technique relativement simple et élégante pour pallier l'absence de verrouillage physique, et qui n'est pas propre à DB2, c'est la technique dite du « verrouillage optimiste ».
- Pour que la technique du verrouillage optimiste puisse être mise oeuvre, il faut que les tables de la base de données respectent une certaine normalisation. Par exemple, il est indispensable que chaque table possède dans ses colonnes, soit un numéro de version, soit des zones mouchards dédiées au stockage des informations suivantes : « qui a mis à jour cette donnée, et quand ? »

Gestion des verrouillages

- Le principe est finalement très simple et se décompose en plusieurs étapes (prenons pour exemple l'écran de mise à jour d'un produit) :
 1. L'utilisateur affiche l'écran de mise à jour d'un produit. La requête SQL d'extraction de la fiche produit va récupérer les informations définissant le produit, ainsi que le contenu des zones mouchards de dernière mise à jour de cette fiche produit.
 2. L'utilisateur modifie des informations de la fiche produit dans un formulaire HTML, puis il valide ce formulaire ce qui a pour effet de déclencher la série d'opérations suivantes :
 1. Le script PHP côté serveur contrôle la validité des informations saisies. Si des anomalies sont détectées, le formulaire est réaffiché avec des messages d'erreur. Si aucune anomalie n'est détectée, le script passe à l'étape suivante
 2. Le script PHP déclenche l'exécution d'une requête SQL de type UPDATE qui aura pour éléments de clé (dans la clause WHERE) l'identifiant de l'enregistrement modifié, ET les colonnes « mouchards » de la dernière modification connue. Si la requête échoue, cela signifie que la ligne dans la table SQL n'existe plus, OU, que cette ligne a subi une modification par un autre utilisateur (ou un autre travail) entre le moment où la ligne a été extraite de la base et le moment où on a tenté de la mettre à jour. Si cela se produit, on informe l'utilisateur qu'il a été pris de vitesse par quelqu'un d'autre, et on peut lui proposer plusieurs possibilités : soit abandonner la transaction, soit la réactualiser avec les dernières informations en base avant de procéder à une nouvelle tentative de mise à jour. Si aucune anomalie ne s'est produite, alors la ligne a bien été modifiée en base, et l'utilisateur peut passer à autre chose.

DESIGN PATTERN ACTIVE RECORD

Le design pattern Active Record

- Le design pattern Active Record est particulièrement bien adapté au développement de programmes de gestion, et notamment de modules de type C.R.U.D. (dont nous parlerons juste après).
- La définition proposée par Wikipédia a le mérite d'être relativement claire:
- *En génie logiciel, le patron de conception (design pattern) active record (enregistrement actif en anglais) est une approche pour lire les données d'une base de données. Les attributs d'une table ou d'une vue sont encapsulés dans une classe. Ainsi l'objet, instance de la classe, est lié à un tuple de la base. Après l'instanciation d'un objet, un nouveau tuple est ajouté à la base au moment de l'enregistrement. Chaque objet récupère ses données depuis la base; quand un objet est mis à jour, le tuple auquel il est lié l'est aussi. La classe implémente des accesseurs pour chaque attribut. »*

http://fr.wikipedia.org/wiki/Active_record_%28patron_de_conception%29

Le design pattern Active Record

- L'implémentation d'Active Record que je propose dans le projet MacaronDB ne couvre pas l'intégralité des fonctionnalités que l'on retrouve dans son implémentation pour des projets plus avancés tels que Ruby on Rails, mais j'ai repris les éléments qui me semblaient les plus pertinents pour les projets que j'ai à mettre en œuvre en environnement IBM i.
- Reprenons l'exemple proposé dans l'article de Wikipédia, adapté à une implémentation PHP via MacaronDB :

```
/* du point de vue d' ActiveRecord, cela donne : */  
$a = new Piece($db) ;  
$a->nom = "Pièce test";  
$a->prix = 123.45;  
$a->save() ;
```

```
/* d'un point de vue SQL et du point de vue du Wrapper de MacaronDB, cela donne : */  
$arg = array('Pièce test', 123.45) ;  
$sql = "INSERT INTO pieces (nom, prix) VALUES (?, ?)" ;  
$db->executeCommand($sql, $arg) ;
```


Le design pattern Active Record

```
/* Si l'on souhaite procéder à la mise à jour de la pièce ayant pour identifiant 10 */
$a = new Piece($db) ;
$a->load(10) ;
$a->nom = "Pièce test";
$a->prix = 123.45;
$a->save() ;

/* d'un point de vue SQL et point du vue du wrapper de MacaronDB, cela donne : */
$args = array('Pièce test', 123.45, 10) ;
$sql = "UPDATE pieces set nom = ?, prix = ? where id = ?" ;
$db->executeCommand($sql, $args) ;
```

A noter : dans l'implémentation d'Active Record pour MacaronDB, la méthode `save()` a pour effet de vérifier que l'enregistrement lu par la méthode `load()` a bien été trouvé. S'il n'est pas trouvé, la requête de mise à jour de la base de données ne sera pas un `UPDATE` mais un `INSERT` (comme dans la diapo précédente). On peut modifier ce comportement en écrivant ceci (à la place de la méthode `save()`) :

```
if ($a->isLoading()) {
    $a->update() ;
} else {
    // message d'erreur à implémenter
}
```

Le design pattern Active Record

La classe ActiveRecord implémentée dans MacaronDB fournit le jeu de méthodes suivantes :

```
function load($id);  
function save();  
function update();  
function create();  
function delete();  
function isLoading();  
function isCreated();  
function isUpdated();  
function isDeleted();  
function isSaved();  
function getDatas();  
function setDatas($datas);  
function getFormElements();  
function isKeyUsed($key);  
function getCrudSelectDefault();  
function getCrudSelectField();  
function getTableName();  
function getClassName();  
function getModeIncrId();  
function getLastIdInserted();
```

Le rôle de chacune de ces méthodes est expliqué dans les commentaires de la classe DBActiveRecord (cf. fichier DBActiveRecord.php).

A noter : la méthode getFormElements renvoie un tableau formaté pour permettre la production de formulaires HTML (en s'appuyant sur le projet HTML Quickform 2).

Le design pattern Active Record

Dans l'exemple ci-contre, vous avez ci-dessous le code SQL DB2 de création de la table PIECES, et à droite le code de création de la classe Pieces dérivée d'Active Record :

```
CREATE TABLE GJABASE.PIECES (  
    ID INTEGER  
        GENERATED ALWAYS AS IDENTITY  
        (START WITH 1, INCREMENT BY 1),  
    NOM CHAR(30) NOT NULL DEFAULT '',  
    PRIX DECIMAL(11, 5) NOT NULL DEFAULT 0,  
    CRE_DATE DATE NOT NULL DEFAULT CURRENT_DATE ,  
    CRE_TIME TIME NOT NULL DEFAULT CURRENT_TIME ,  
    CRE_USID CHAR(20) NOT NULL DEFAULT USER ,  
    UPD_DATE DATE NOT NULL DEFAULT CURRENT_DATE ,  
    UPD_TIME TIME NOT NULL DEFAULT CURRENT_TIME ,  
    UPD_USID CHAR(20) NOT NULL DEFAULT USER ,  
    STATUT CHAR (1 ) NOT NULL WITH DEFAULT ' '  
) ;  
  
CREATE INDEX GJABASE.PIECES01 ON GJABASE.PIECES (ID) ;  
CREATE INDEX GJABASE.PIECES02 ON GJABASE.PIECES  
    (NOM, ID) ;  
COMMENT ON TABLE GJABASE.PIECES IS 'Liste des Pièces'  
;
```

```
class PieceModel extends DBActiveRecord  
    implements intDBActiveRecord {  
    public function __construct($base) {  
        $this->table_name = 'PIECES';  
        $this->schema_name = 'GJABASE';  
        $this->fields_name = array ('id', 'nom', 'prix',  
            'cre_date', 'cre_time', 'cre_usid',  
            'upd_date', 'upd_time', 'upd_usid',  
            'statut' );  
        $this->key_name = 'id';  
        $this->key_value = null;  
        $this->user_key = $this->key_name ;  
        $this->description_field = 'nom';  
        $this->autofill_on_update = array(  
            'upd_date'=>'*date',  
            'upd_time'=>'*time',  
            'upd_usid'=>'*user'  
        ) ;  
        $this->autofill_on_insert = array(  
            'cre_date'=>'*date',  
            'cre_time'=>'*time',  
            'cre_usid'=>'*user'  
        ) ;  
        parent::__construct ( $base );  
    }  
}
```

C.R.U.D.

CREATE - RETRIEVE - UPDATE - DELETE

Le C.R.U.D.

- Le sigle C.R.U.D. est utilisé par les développeurs anglophones pour désigner les 4 opérations unitaires que l'on retrouve de manière récurrente dans la plupart des modules de gestion:
 - C (Create) : la création de nouvelles lignes dans une table SQL
 - R (Retrieve) : la récupération de ligne (génération pour affichage)
 - U (Update) : la mise à jour de ligne de table SQL
 - D (Delete) : la suppression de ligne de table SQL
- L'utilisation de boîte à outils dédiées à la génération et la maintenance de modules de type CRUD se révèle vite indispensable, dès lors que l'on souhaite industrialiser le développement d'un projet. D'un point de vue SQL, de quoi a-t-on besoin :
 - De l'une des techniques de pagination vues précédemment (pour l'affichage des listes) (qui sont intégrées dans le projet MacaronDB)
 - Des fonctionnalités équivalentes à celles proposées par le design pattern « Active Record » pour les opérations de maintenance unitaire de la base de données

Le C.R.U.D.

- D'un point de vue de la génération des pages webs, de quoi a-t-on besoin :
 - De fonctions de génération de formulaire (je vous recommande le projet HTML Quickform, et plus particulièrement la version 2. Récemment réécrit en PHP 5, et bien qu'encore en bêta, il est déjà pleinement opérationnel)
http://pear.php.net/package/HTML_QuickForm2
 - De connaissances en HTML et CSS (pour la mise en forme des pages webs)
 - De connaissances plus ou moins approfondies en Javascript, selon le type d'ergonomie recherché. Par exemple, plus vous voudrez vous orienter vers une ergonomie de type RIA (Rich Internet Application), et plus vous devrez élever votre niveau de compétence en Javascript. Vous aurez tout intérêt dans ce cas à vous tourner vers un framework Javascript tel que JQuery, Dojo, ExtJS ou YahooUI, pour ne citer que les plus connus. Vous pourrez ainsi mettre en œuvre assez rapidement des interfaces à l'ergonomie « enrichie » de fonctionnalités telles que :
 - l'auto-complétion,
 - des champs de type « ascenseur » (slider),
 - des affichages de type « tabulaire » ou « accordéon »,
 - des champs de saisie de date avec calendrier (datepicker),
 - la réactualisation de portions d'écran sans rechargement complet de la page,
 - etc...






















Le C.R.U.D.

La constitution d'une boîte à outils adaptée à vos besoins peut vous permet de produire très rapidement des écrans de recherche et de mise à jour tels que celui ci-dessous :

Critères de recherche

Recherche par code :

Recherche par libellé :

	Code	Libellé
  	R10	Agence Lisieux
  	R30	
  	R32	
  	R35	
  	R37	
  	R38	
  	R40	

<< Préc. | 1-20 | 21 | Suiv. >>
(Affichage 1 à 7 sur 21)

Modification d'une donnée

* Code agence

R10

* Nom agence

Agence Lisieux

* Banque rattachée

513200

* signale les champs obligatoires.

© Copyright 2011 Six-Axe Consultants | contact contact@six-axe.fr

Ressources bibliographiques pour DB2

- Le site IBM Data Management propose régulièrement de très bons articles (en anglais) présentant des techniques SQL avancées pour DB2:
 - <http://ibmdatamag.com>
- Deux très bons ouvrages (en français), très complémentaires, pour découvrir de nombreuses techniques SQL avancées. Chaque technique est présentée dans différentes versions (une par SGBD). Contrairement au premier, le second ouvrage ne donne pas d'exemples pour DB2, mais de nombreuses techniques présentées pour MySQL peuvent être facilement adaptées à DB2). :
 - SQL, le livre de recettes, par Anthony Molinaro
 - SQL à 200 %, par Andrew Cumming et Gordon RussellCes 2 ouvrages peuvent être commandés en PDF sur <http://librairie.immateriel.fr>
- Un très bon ouvrage (en anglais) pour apprendre à reconnaître les mauvaises pratiques SQL, et la manière de les corriger :
 - SQL Antipatterns: Avoiding the Pitfalls of Database Programming, par Bill Karwin
 - Disponible en PDF ici : <http://pragprog.com/book/bksqla/sql-antipatterns>

Ressources bibliographiques pour PHP

- Un ouvrage au contenu très riche, édité chez Sitepoint :
PHP Master, Write Cutting-Edge Code
 - par Lorna Mitchell, Davey Shafik & Matthew Turland , Ed. Sitepoint
 - <http://www.sitepoint.com/books/phppro1/>
- Packt Publishing propose un riche catalogue d'ouvrages consacrés à PHP (je vous recommande notamment « PHP JQuery Cookbook ») :
 - [Search results for: 'php' \(packtpub.com\)](#)
- PHP Architect propose également un fond très intéressant d'ouvrages dédiés à PHP (plusieurs ouvrages sont édités avec le soutien de Zend) :
 - <http://www.phparch.com/books/> (je recommande notamment « PHP Playbook »)
- L'éditeur américain Oreilly propose un impressionnant catalogue dématérialisé, avec la possibilité de consulter les livres en ligne, selon différentes formules d'abonnement. On peut y consulter de nombreux ouvrages consacrés à PHP :
 - <http://www.oreilly.com>

Ressources en ligne

- Le site DeveloperWorks propose de très bons articles consacrés à PHP :
- <http://www.ibm.com/developerworks/topics/php/>

La plupart des articles proposés par Developerworks sont excellents. Si vous ne savez pas par quoi commencer, je vous recommande de lire en priorité les articles écrits par Wez Furlong (notamment sur PDO), Jack Herrington (notamment sur l'utilisation de XML avec PHP), et Nathan A. Good (notamment sur les « good habits » et les « design patterns »).

- Les redbooks proposés en téléchargement libre par IBM constituent une précieuse source d'information. Plusieurs redbooks sont consacrés à PHP et à DB2 :
 - <http://www.redbooks.ibm.com>

Avez-vous des questions ?

DB2 et PHP – Les bonnes pratiques

Merci de votre attention.