

Perl – DBI

Récupérer les données : différentes méthodes

```
$ary_ref = $sth->fetchrow_arrayref;  
$ary_ref = $sth->fetch;      # alias
```

Récupère le prochain enregistrement et le retourne sous forme de référence sur un tableau. Les champs sont dans l'ordre qu'ils étaient dans la requête, et les NULL sont transformés en undef (c'est toujours le cas : je ne le mentionnerai pas!). C'est la façon la plus rapide de récupérer des enregistrements (particulièrement si utilisé avec `bind_columns`), mais pas nécessairement la plus claire.

```
@row_ary = $sth->fetchrow_array;
```

Récupère le prochain enregistrement et le retourne sous forme de tableau. Légèrement plus lent, parce que les données du tableau doivent être recopiées plutôt que simplement recopier la référence, mais souvent aussi plus clair.

```
$hash_ref = $sth->fetchrow_hashref;
```

Les enregistrements sont retournés sous forme de référence sur un tableau associatif. La clé utilisée pour récupérer la valeur d'un champ est le nom du champ. Ça donne une manipulation plus claire, moins ambiguë et plus robuste d'un enregistrement : par exemple, avec un `SELECT * FROM table`, un changement de l'ordre des champs dans la table n'affectera le code.

Un truc à surveiller : actuellement, la référence retournée est nouvelle pour chaque enregistrement, mais la documentation de DBI avertit qu'éventuellement, ce sera la *même* référence qui sera réutilisée, donc attention si vous utilisez cette construction pour assembler des enregistrements, par exemple dans un tableau.

```
$ary_ref = $sth->fetchall_arrayref;  
$ary_ref = $sth->fetchall_arrayref( $slice, $max_rows );
```

Retourne d'un coup tous les enregistrements retournés par votre requête. Utilisez pour les requêtes qui ne retournent que quelques enregistrements, mais attention : DANGER si vous recevez un grand nombre (ou un nombre inconnu) d'enregistrements...

```
$hash_ref = $sth->fetchall_hashref( $key_field );
```

Retourne les résultats d'un coup, encore une fois, mais en utilisant le champ `$key_field` pour accéder aux enregistrements. Le champ utilisé devrait évidemment être unique pour chaque enregistrement (la clé primaire étant donc un bon candidat...)

Exemple :

```
$sth = $dbh->prepare(« SELECT * FROM chanson »);  
$res = $sth->fetchall_hashref(« id »);  
print « La titre de la chanson no 42 est : » . $res->{'42'}->{'titre'} . «\n »;
```

```
$rv = $sth->rows;
```

Retourne le nombre d'enregistrement affecté par la dernière opération (par exemple dans le cas d'un UPDATE). Si tous les enregistrements d'un SELECT ont été retournés, ça va aussi retourner le nombre d'enregistrements (donc, inutile de mettre votre propre compteur...).

Optimisation et autres trucs

```
$rc = $sth->bind_param($p_num, $bind_value);
```

```
$rc = $sth->bind_col($column_number, \svar_to_bind);
```

« Attache » une variable a, respectivement, les paramètres d'une requête ou les résultats d'une requête. Permet de jouer avec les références mémoires de façon à optimiser la vitesse de traitement des requêtes. Utilise pour les traitements en « batch » d'un grand nombre de données. Trop complexe pour les petites applications du genre que vous aurez à faire.. ;)

Des raccourcis :

```
$ary_ref = $dbh->selectall_arrayref($statement);  
$hash_ref = $dbh->selectall_hashref($statement, $key_field);
```

```
$ary_ref = $dbh->selectcol_arrayref($statement);  
$ary_ref = $dbh->selectcol_arrayref($statement, \%attr);
```

```
@row_ary = $dbh->selectrow_array($statement);  
$ary_ref = $dbh->selectrow_arrayref($statement);  
$hash_ref = $dbh->selectrow_hashref($statement);
```

Série de méthodes *raccourcis* qui permettent de faire en une seule opération le prepare/execute/fetch.

Intéressante, mais tant qu'à faire des raccourcis, aussi bien regarder du côté de DBIx::Recordset ou Class::DBI, qui vont offrir une interface plus simple et/ou plus « portable » sur une base de données.

Construire dynamiquement des requêtes

```
$quoted_string = $dbh->quote($string);
```

Permet de « protéger » la chaîne de caractère à insérer dans une requête SQL.