Mysql - Requêtes SELECT et optimisation

Date de dernière mise à jour : 27/08/2007 à 01:12

Source : http://www.vulgarisation-informatique.com/mysql-select.php.
Distribution interdite sans accord écrit d'Anthony ROSSETTO (http://www.vulgarisation-informatique.com/contact.php)

Pour sélectionner des données contenues dans une base de données Mysql, onutilise les requêtes de type **SELECT**

.

Nous allons utiliser la table **membres**

quenous avons créée tout à l'heure (j'ai rajouté des données pour l'exemple) :

Table membres

ID
pseudo
passe
mail

1
webmaster
password
mail@fai.fr

2
test
1234
test@essai.fr

3
Anthony
passe
anthony@fai.fr

Nous allons rajouter une deuxième table qui s'intitule **profil**
(vous verrez pourquoi nous avons crée une deuxième table plus bas). Voici sastructure :

Table profil

ID
abonne_newsletter
signature
config

1
1
C'est moi le web ;-)
P4P800...

3

1
Test
anthony@fai.fr

Commençons les sélections de base :

Imaginions que nous souhaitons sélectionner tous les champs de la table **membres**
où le pseudo est 'webmaster'. La requête sera la suivante :

```
SELECT
pseudo,passe,mail
  FROM
membres
WHERE
  pseudo=
'webmaster'
```

On aurait pu aussi faire ceci (pour sélectionner tous les champs). Ceci està titre d'exemple, ne l'utilisez pas sur votre serveur mysql car vous perdriezen performances :

```
SELECT
*
  FROM
membres
WHERE
  pseudo=
'webmaster'
```

Pour effectuer une requête sur une base de données, on utilise la fonctionmysql_query("requete");

Exemple avec la requête que je viens de vous présenter :

```
$requete=
mysql_query("
SELECT
pseudo,passe,mail
  FROM
membres
WHERE
  pseudo=
'webmaster'
");
```

Les jointures :

Une jointure est une sélection de données sur plusieurs tables. Cette solution évite la redondance des données et permet donc d'économiser de la

place, voire de gagner des performances si elle est bien utilisée.

Exemple :

Nous voulons pour chaque membre obtenir son email, sa signature et sa config. Les deux tables ont en commun un champ : ID (les ID de la table **membres** correspondent à l'ID du profil)

La requête sera celle-ci :

```
$requete=
mysql_query("SELECT
membres.mail,profil.signature,profil.config
 FROM
membres
LEFT JOIN
profil
ON
membres.id=profil.id
");
```

L'ordre du **LEFT JOIN** est important. Si l'ID dans la table profil n'existe pas (par exemple l'ID numéro 2 ici), l'enregistrement est quand même retourné (avec une valeur de type **NULL**). Si nous avions inversé l'ordre de jointure et avions écrit **FROM profil LEFT JOIN membres**, l'enregistrement n'aurait pas été retourné puisque rien ne se trouve dans la table **profil** avec un ID de 2.

Les opérations mathématiques :

Mysql permet de retourner des calculs mathématiques. Par exemple l'addition de 5+9 :

```
mysql_query("SELECT
5+9
")
;
```

Cette requête va retourner 14.

Sélectionner la valeur maximum d'un champ :

```
mysql_query("SELECT
max(champ)
FROM
table
")
;
```

Sélectionner la valeur minimum d'un champ :

```
mysql_query("SELECT
min(champ)
FROM
table
")
;
```

Sélectionner la valeur moyenne d'un champ :

```
mysql_query("SELECT
AVG(champ)
FROM
table
")
;
```

Sélectionner la somme de toutes les valeurs d'un champ :

```
mysql_query("SELECT
SUM(champ)
FROM
table
")
;
```

Compter le nombre de valeurs d'une table (ici on compte le nombre de lignes qui ont la valeur "1" sur le champ "champ":

```
mysql_query("SELECT
COUNT
(*)
FROM
table
WHERE
  champ=1
")
;
```

On aurait pu aussi faire ceci :

```
$requete=
mysql_query("SELECT
*
FROM
table
WHERE
  champ=1
")
;$nbre_lignes=
mysql_num_rows(
$requete
);
```

Les opérations sur les chaînes :

Plutôt que de joindre avec un peu de texte et PHP deux valeurs retournées par Mysql, autant clarifier le code PHP et utiliser l'option **CONCAT** de Myql :

```
mysql_query("SELECT
CONCAT(
'valeur du premier champ: '
, champ1 ,
'valeur du deuxième champ :'
, champ2)
FROM
table
")
;
```

La clause GROUP BY :

La clause **GROUP BY**
se combine avec **COUNT**
et permet d'effectuer des calculs sur chaque itération. Je m'explique :

Prenons une table nommée **forum**
.

Voici sa structure :

Table forum

topic_id
reponse
sujet
message

1

test
Ceci est un test

1
1
test
Ah bon ?

2

le site vulgarisation-informatique
Un autre test

Lorsque un nouveau topic est créé, le champ "reponse" prend pour valeur 0, dans le cas contraire (lorsqu'on répond à un topic), le champ "reponse" vaut 1. Le champ "topic_id" représente le numéro de topic.

Nous voulons compter le nombre de messages par topic. Nous ferons donc une requête comme ceci :

```
mysql_query('SELECT
COUNT(reponse)
FROM
forum
GROUP BY
topic_id
')
;
```

La clause LIMIT :

La clause LIMIT permet de ne pas retourner tous les enregistrements d'une table. Par exemple si vous souhaitez afficher les 10 premières news de votre site, voici un exemple de requête :

```
mysql_query('SELECT
titre,date
FROM
news
LIMIT
0,10
')
;
```

Dans la clause LIMIT, le premier nombre (ici 0) représente l'enregistrement de départ à afficher, et le deuxième nombre (ici 10), le nombre d'enregistrements à retourner.

La clause DISTINCT :

Elle permet d'éviter de retourner des doublons. Dans notre cas nous souhaitons retourner tous les numéros de topic mais une seule fois(pas de doublons) :

```
mysql_query('SELECT DISTINCT
 topic_id
FROM
forum
')
;
```

Optimisation des requêtes SELECT :

Optimiser Mysql est une opération facile. La rapidité de traitement en suivant ces conseils peut être multipliée par plus de 1000 !

La clause **SQL_SMALL_RESULT**
- SQL_BIG_RESULT
:

Cette clause permet avec les clauses **DISTINCT**
et **GROUP BY**
de dire à Mysql qu'il y aura surement peu ou beaucoup d'enregistrements à retourner.

Exemple :

```
mysql_query('SELECT SQL_SMALL_RESULT
DISTINCT
   topic_id
FROM
forum
LIMIT
 20
')
;
```

Nous n'avons ici que 20 enregistrements à retourner, la clause **SQL_SMALL_RESULT**
convient parfaitement.

Les index :

Un index est en quelque sorte une table des matières. Lorsque une table contient un nombre suffisant d'enregistrements (plus de 50), un index est la solution ultime pour doper les performances de Mysql.

Les index sont à placer sur les colonnes utilisées dans les clauses WHERE, GROUP BY, et LEFT JOIN notamment. Nous pouvons par exemple ajouter un index sur le champ ID dans les tables **membres**
et **profil**
.

Source : http://www.vulgarisation-informatique.com/mysql-select.php.
Distribution interdite sans accord écrit d'Anthony ROSSETTO (http://www.vulgarisation-informatique.com/contact.php)