

MySQL 5.0 : Un SGBDR mature ?

Chapeau

MySQL est le SGBD Open Source le plus populaire au monde. Sa cinquième version, sortie en octobre 2005, permet de mieux répondre aux problématiques d'entreprise. Au menu des nouveautés fonctionnelles : les vues, les procédures stockées, les déclencheurs, de nouveaux moteurs de stockage, la base de données INFORMATION_SCHEMA et diverses petites améliorations.

Avec toutes ces nouveautés la philosophie de MySQL reste la même : « simplicité et hautes performances ».

Fonctionnalités

Déjà, avant même la version 5, MySQL supportait de nombreuses fonctionnalités avancées lui permettant de répondre à un grand nombre de problématiques d'entreprise : Les requêtes imbriquées depuis MySQL 4.1, les transactions depuis MySQL 3.23 ainsi que les clés étrangères et l'intégrité référentielle.

	MySQL 4.1	MySQL 5.0	MySQL 5.1
Clés étrangères / Intégrité référentielle	Avec le moteur innnoDB	Avec le moteur innnoDB	Avec le moteur innnoDB
Réplication	Oui	Oui	Oui
Requêtes imbriquées	Oui	Oui	Oui
Vues		Oui	Oui
Procédures stockées		Oui	Oui
Déclencheurs		Oui	Oui
Partitionnement			Oui

Vues

Des vues pour faciliter la visibilité

Les vues sont la plupart du temps utiles pour donner aux utilisateurs l'accès à un ensemble de relations représentées sous la forme d'une table. Une vue est une table virtuelle ; les données de la vue sont en fait des champs de différentes tables regroupées, ou des résultats d'opérations sur ces champs.

Des vues pour améliorer la confidentialité

Une vue n'est pas forcément un regroupement de plusieurs tables mais peut être un sous ensemble d'une table (ou de plusieurs) ce qui permet de cacher des champs aux utilisateurs. Par exemple il ne sera pas forcément utile à tout le monde d'accéder aux champs indiquant les bénéfices réalisés sur un projet dans votre base comptable. Vous pouvez donc créer une vue contenant tous les champs de la table projet sauf le champs bénéfice.

L'approche avec MySQL 5 sera donc plus souple car elle ne force plus un découpage de table pour gérer la confidentialité et les droits donnés aux utilisateurs. Les vues permettront de remplir ce rôle.

Les vues compliquent les mises à jour

Insérer ou modifier des données dans une vue n'est pas aussi simple que de faire un update dans une table. Pour pouvoir le faire il faut réfléchir de façon plus poussée au modèle conceptuel de données :

Une vue n'est pas forcément accessible en insertion / modification. Pour cela il faut qu'il n'y ait pas d'incompatibilité logique à ce qu'elle le soit.

- Tous les champs des tables possédant des contraintes d'intégrités (index unique, clés primaires ..) doivent être présent
- La vue ne doit pas posséder de regroupement ou d'exclusion (GROUP BY , DISTINCT, UNION)
- Le plan d'exécution de la vue ne doit pas passer par une table temporaire

Les vues de MySQL 5 par la pratique

```
CREATE
[OR REPLACE]
VIEW view-name
[(column-list)]
AS select-statement
```

Créer une vue revient à appliquer un filtre à une ou plusieurs tables. Pour schématiser prenons une entreprise normale. Dans celle-ci il y a des employés regroupés sous le terme 'personnel'. On regroupe ces employés par 'catégorie' en fonction de leur activité (administratif, informatique, commercial,...).

On peut créer une vue contenant l'ensemble des informaticiens avec le code suivant :

```
CREATE VIEW personnellinformatique
AS SELECT a.nom AS nom,a.prenom AS prenom,b.service AS service
from categorie b, personnel a
where a.fkCategorie = 1 and a.fkCategorie = b.pkCategorie;
```

Optimisation des vues de MySQL 5

```
CREATE
[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
[DEFINER = { user | CURRENT_USER }]
[SQL SECURITY { DEFINER | INVOKER }]
VIEW view-name
[(column-list)]
AS select-statement
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

- La clause facultative «ALGORITHM» n'est pas standard. Elle permet d'optimiser votre code.
- MERGE utilise la requête SQL ayant servie à la création de la vue comme base d'opération.
- TEMPTABLE utilise une table temporaire créée pour stocker les résultats.

Par défaut l'optimiseur MySQL décide lui-même quelle option choisir (UNDEFINED). Pour améliorer les performances, il est généralement plus intéressant de le définir statiquement afin d'éviter à l'optimiseur de le découvrir à chaque exécution.

«DEFINER» assigne un créateur à la vue.

«SQL SECURITY» définit quelles seront les droits de l'utilisateur, lors de l'exécution de la vue.

Deux valeurs sont possibles:

- DEFINER permet d'exécuter la vue avec les droits du créateur.
- INVOKER permet d'exécuter la vue avec ses propres droits.

Pour finir, la clause facultative CHECK OPTION permet de ne modifier que la vue ou du moins des informations qui respectent les contraintes de la vue. Ainsi dans notre exemple, avec la clause CHECK OPTION, il ne sera pas possible de modifier au travers de la vue « personnelInformatique » une personne ne travaillant pas dans le service informatique. Manque

Le seul léger bémol est que la version 5.0 de MySQL ne disposera pas des vues matérialisées. Les vues matérialisées sont des données physiquement dupliquées dans le SGDB. Par exemple le résultat d'un calcul n'est plus à refaire pour chaque accès. Leur utilité se fait particulièrement sentir lors de traitements de tables très volumineuses .

Comparaison avec d'autres SGBD

	MySQL	IBM DB2	Oracle	SQL Server
Basic	Oui	Oui	Oui	Oui
UNION ALL	Non	Oui	Oui	Oui
JOINS	Oui	Oui	Oui	Oui
INSTEAD OF	Non	Oui	Oui	Oui
UPDATEABLE_KEY	Oui	Non	Non	Non

Possibilité des vues avec MySQL 5.0

UPDATEABLE_KEY est une fonctionnalité de MySQL permettant de modifier une clef primaire par le biais d'une vue.

Procédures stockées et fonctions

Les procédures stockées sont des listes de commandes qui peuvent être compilées et stockées sur le serveur. Elles permettent de déplacer une partie de la logique métier d'une application de base de données du client vers le serveur. Les clients n'ont plus besoin de soumettre à nouveau toute la commande, mais font simplement référence à la procédure stockée. Cela se traduit par une amélioration de la sécurité, une diminution de la redondance du code, et une augmentation des performances.

Des procédures stockées pour améliorer la sécurité

Elles peuvent fournir une protection contre les attaques d'injection SQL, principalement contre celles qui utilisent un opérateur AND ou OR pour ajouter des commandes à une valeur de paramètre d'entrée valide. Les programmes clients n'accédant plus directement aux tables. Toutes les opérations de gestion des données sont effectuées via des procédures stockées.

Des procédures stockées pour centraliser les requêtes

Différentes applications peuvent accéder à la même base de données et avoir les mêmes fonctionnalités. Les procédures stockées peuvent alors servir à factoriser ce code SQL commun ce qui permet de diminuer la redondance et facilite la maintenance du code.

Des procédures stockées pour augmenter les performances

Les commandes n'ont pas à être analysées plusieurs fois, elles sont (pré)compilées sur le serveur et bien moins d'informations transitent sur le réseau, le trafic y est donc limité. Les procédures stockées sont particulièrement utiles quand les clients qui accèdent à la base de données ne sont pas sur le même serveur.

Le principal inconvénient des procédures stockées, et qu'elles nous rendent complètement dépendantes de l'éditeur de la base de données, puisqu'il n'existe pas de langage universel de développement de procédures stockées.

Une migration simplifiée par le respect du standard SQL 2003

Les procédures stockées de MySQL 5 respectent certaines recommandations du standard SQL 2003. La syntaxe est donc très proche de la syntaxe de DB2 ce qui permet une migration facile entre les deux outils. La migration en provenance d'Oracle ou de MS SQL Server impliquera plus de travail manuel étant donné que leurs bases de données ne respectent pas autant le standard.

Astuce : Pour migrer d'Oracle vers MySQL on peut utiliser les outils de migration Oracle vers DB2 puis passer de DB2 à MySQL qui sont très proches.

Syntaxe

```
CREATE
[DEFINER = { user | CURRENT_USER }]
PROCEDURE sp_name([parametres])
    LANGUAGE SQL
    | [NOT] DETERMINISTIC
    | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
    | SQL SECURITY { DEFINER | INVOKER }
    | COMMENT 'string'
routines
```

Déclencheurs

Les déclencheurs (triggers) sont des ordres de déclenchement d'opérations quand un évènement survient sur une table.

Des déclencheurs pour maintenir la cohérence des données

Ils sont souvent utilisés pour assurer la cohérence des données dans la base, en réalisant des contraintes qui doivent porter sur plusieurs tables.

Les déclencheurs combinés aux transactions permettent de créer tous les mécanismes d'intégrité référentielle. La norme SQL 3 a d'ailleurs imposé l'utilisation des déclencheurs. Initialement prévu pour la version 5.1, l'équipe de développement a finalement profité d'une avance sur le calendrier de développement pour proposer une version simplifiée des déclencheurs dans la version 5.0.

Syntaxe de base

```
CREATE TRIGGER trigger_nom
[DEFINER = { user | CURRENT_USER }]
{ BEFORE | AFTER }
{ INSERT | UPDATE | DELETE }
ON table name
FOR EACH ROW
triggered SQL statement
```

Le premier élément entrant en compte est le nom (trigger_nom). L'action qui est déclenchée l'est à la suite d'un évènement (ex : insertion d'un nouvel enregistrement dans une table). Le second paramètre (BEFORE ou AFTER) indique si le déclencheur doit être lancé avant ou après l'évènement.

Les déclencheurs peuvent être activés durant l'appel à un INSERT, un UPDATE ou un DELETE.

```
CREATE TRIGGER trig_livre
BEFORE INSERT
```

Le déclencheur est lié à une table que nous définissons avec le mot clef ON :

```
CREATE TRIGGER trig_livre
BEFORE INSERT
```

ON livre

On définit alors les instructions à effectuer une fois le déclencheur activé.

```
CREATE TRIGGER trig_livre
BEFORE INSERT
ON livre
FOR EACH ROW
BEGIN
    INST1;
    INST2;
END;
```

Prenons un cas pratique :

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
-> FOR EACH ROW SET @sum = @sum + NEW.amount;
```

Limitation des déclencheurs liées à la version 5.0

Comme nous l'avons indiqué plus amont le support des déclencheurs est un plus dans la version 5.0 de MySQL : C'était prévu pour la version 5.1. Il n'en reste pas moins que c'est une version light qui souffre à ce jour de quelques limitations :

- On ne peut associer un déclencheur à une vue ou à une table temporaire,
- Les déclencheurs ne peuvent pas non plus faire appel à une procédure stockée ou à des fonctions.

INFORMATION_SCHEMA

La base INFORMATION_SCHEMA, standard du SQL:2003, fournit un accès aux métadonnées du serveur MySQL.

Les métadonnées sont les informations sur les données, telles que le nom des bases de données, le nom des tables, le type des données, d'index, les droits d'accès, etc. INFORMATION_SCHEMA est une base de données virtuelle, vous ne pouvez y accéder qu'en lecture. En effet, les tables qui la composent, ne sont en fait que des vues. Vous ne verrez donc pas sur le disque dur de fichiers associés.

Par exemple, pour récupérer le code des procédures stockées, de la base de données FRESHDAZ, il faut pour cela, sélectionner les colonnes ROUTINE_NAME et ROUTINE_DEFINITION de la table information_schema.ROUTINES:

```
mysql> SELECT R.ROUTINE_NAME, R.ROUTINE_DEFINITION
-> FROM information_schema.ROUTINES R
-> WHERE R.ROUTINE_TYPE = 'PROCEDURE' AND R.ROUTINE_SCHEMA = 'FRESHDAZ';
```

Goodies and co

Amélioration des traitements mathématiques

Avec MySQL 4.1 le résultat de la commande SELECT .01 * .01 renvoyait 0. MySQL 5 utilise une bibliothèque permettant des calculs plus précis. Ainsi 0.0001 est traité comme une valeur exacte et non plus comme une valeur approximative.

Nouveau moteur de stockage

Un nouveau moteur de stockage a été ajouté depuis MySQL 5.0.3 : FEDERATED Storage Engine. Il permet d'accéder à des données présentes sur des bases de données distantes.

Conclusion

MySQL 5 apporte un réel confort en terme de fonctionnalités par rapport à ses versions antérieures. Les déclencheurs, les vues et les procédures stockées propulsent MySQL dans le monde des SGBD matures et permet de soutenir la comparaison avec Oracle, PostgreSQL et autres.

Avec cette nouvelle version MySQL élargit son public en ne s'adressant plus principalement aux architectes d'applications Web mais à tous les développeurs.