Adaptateurs SQL

Spécifications fonctionnelles et techniques

Emmanuel Bourgerie

Table des matières

1 Avant-propos....2
1.1 Licence....2
2 Problématique....3
3 Objectifs....4
3.1 Plannification....4
3.2 Fonctionnalités....4
3.3 Implémentation....4
3.4 Modèle....4
4 Développer un adaptateur....5
4.1 Plugin....5
4.1.a Convention de nommage....5

1 Avant-propos

Ce document public a pour objectif de présenter l'avancée majeure de DbMaster 0.8 : les adaptateurs SQL. Il fournit les spécifications nécessaires pour l'équipe de développement de DbMaster ainsi qu'aux développeurs tiers souhaitant améliorer le support de leur SGBD.

1.1 Licence

Pour toute question, vous pouvez me contacter directement : manu.dwarf@gmail.com

2 Problématique

À l'heure où ce document est rédigé, DbMaster 0.7 est en cours de finition.

Pour rappel, il s'appuie sur la technologie Qt, et plus particulièrement sur le framework QtSql. Celui-ci fournit toutes les classes nécessaires pour se connecter à une base de données relationnelle, préparer/exécuter une requête, parcourir une table, etc.

Or, ces drivers ne sont pas conçus pour interroger et modifier directement la structure d'une base. Il est nécessaire d'utiliser les requêtes type CREATE TABLE, qui sont standardisées pour la plupart. Certains manques dans la norme SQL (aucune commande pour connaître la structure d'une table) sont comblés par des instructions spécifiques à chaque SGBD.

Afin de palier à ce manque, DbMaster s'appuiera sur des plugins particuliers, les adaptateurs SQL. Un adaptateur permettra d'abstraire ces commandes spécifiques en respectant une API donnée.

3 Objectifs

3.1 Planification

Une préversion des adaptateurs devra être fournie pour DbMaster 0.8. L'objectif est de pouvoir tester leur implémentation, afin de pouvoir finaliser l'API pour DbMaster 0.9. Quelques adaptateurs pourront être fournis, en priorité pour les SGBD libres majoritairement utilisés (à choisir parmi MySQL, PostgreSQL, Firebird et SQLite).

3.2 Fonctionnalités

Voici une liste (à compléter) de fonctionnalités qui seront couvertes par l'API pour la v0.8 :

- Liste des tables (par schéma ou sur l'ensemble de la base)
- Liste des schémas (#948)
- Liste des BDD pour un serveur donné (#978)

Pour la v0.8 l'objectif fonctionnel est de remplacer QSqlDriver. Les versions suivantes permettront l'ajout de fonctionnalités supplémentaires (notamment la modification de la structure).

QSqlDriver sera toujours utilisé, mais seulement pour la connexion et la gestion des requêtes. Les adaptateurs doivent se cantonner à la structure des bases.

3.3 Implémentation

La mise en place des adaptateurs doit être minimaliste.

Exemple :

```
1:QSqlDatabase *db = [...];
2:Wrapper w(db);
3:QStringList schemas = w.schemas();
```

3.4 Modèle

DbManager représente l'arbre de connexion par un modèle. Afin d'en faciliter le parcours, les items embarquent leur « type » via Qt::UserRole.

Représentation	**Qt::UserRole**
• Connexion	`DbManager::DbItem`
• Schémas	
• [Liste des schémas]	`DbManager::SchemaItem`
• Tables	`DbManager::TableItem`
• [Liste des tables]	
• Vues	`DbManager::ViewItem`
• [Liste des vues]	
• Tables sys	`DbManager::SysTableItem`
• [Liste des tables sys]	

4 Développer un adaptateur

Cet exemple est fourni pour MySQL.

DbmMysqlWrapper.h (extrait)

```
1:public class DbmMysqlWrapper : public Wrapper {
2:
3:public:
4:    [...]
5:    QStringList tables(QString schema);
6:    [...]
7:}
```

DbmMysqlWrapper.cpp (extrait)

```
1:QStringList DbmMysqlWrapper::tables(QString schema) {
2:    QStringList tables;
3:    QString sql = QString("select table_name from information_schema.tables where
  table_schema = '%1' and table_type = 'BASE TABLE'").arg(schema);
4:    QSqlQuery q(*db);
5:    q.exec(sql);
6:    while(q.next())
7:        tables.append(q.value(0).toString());
8:
9:    return tables;
10:}
```

4.1 Plugin

Ne pas perdre de vue qu'un adaptateur est un plugin. Par conséquent, il doit remplir toutes les informations afférentes : auteur, version, code, etc.

4.1.a Convention de nommage

Il est demandé aux développeurs tiers de respecter une certaine convention de nommage pour les codes de plugins.

Rappel : chaque plugin doit pouvoir être identifié de manière unique. Cette contrainte est d'autant plus importante pour les adaptateurs que DbMaster stocke, pour chaque connexion, l'ID d'adaptateur correspondant.

Nommage conseillé :

`[Nom du fournisseur/éditeur].[Nom du SGBD].WRAPPER`

Exemple :

`DBM.MYSQL.WRAPPER`

DBM étant réservé à l'équipe de DbMaster. Nous demandons aux développeurs tiers de ne pas utiliser ce préfixe.

