Présentation et utilisation d'HSQLDB

par Baptiste Wicht (home)

Date de publication : 12 Octobre 2006

Dernière mise à jour : 19 Mars 2009

I - Présentation

HSQLDB est une base de données écrite en Java. C'est une base de données embarquée, vous pouvez donc la transporter directement avec votre programme. Cette base de données est déjà utilisée dans de nombreux projets (Open Office, JBoss, Hibernate), donc vous ne risquez rien à l'utiliser. Elle est tout à fait gratuite et ses sources sont disponibles. HSQL supporte 95% des fonctions de **JDBC**.

HSQL peut aussi fonctionner en client/serveur.

Certains de mes bouts de code ne seront compatibles qu'avec les dernières versions d'HSQL. Mais vous pouvez tout à fait mettre à jour votre base HSQL sans aucun problème, vous profiterez ainsi des dernières nouveautés.

Fonctionnement

Toutes les données sont écrites en mémoire pour améliorer la vitesse d'accès, ce qui ne permet donc pas le traitement d'énormes bases de données, qui risquerait de causer une OutOfMemoryError. Lors de la fermeture de la connexion à la base de données (voir plus loin), HSQL va écrire toutes ses données volatiles dans des fichiers. Malgré cela, il est aussi possible de créer des tables en dur sur le disque.

De plus, ne craignez pas de perdre vos données en cas de problème, toutes les requêtes sont contenus dans un fichier log qui est chargé sur la base en cas de problème et vous permet ainsi de récupérer tout ou une partie des données. Vous pouvez modifier la taille de ce fichier (200Mo par défaut) via la requête :

```
SET LOG_SIZE new_size
```

II - Utilisation

L'utilisation d'HSQL n'est pas compliquée, comme HSQL respecte les standards **SQL**, c'est très ressemblant à une autre base de données.

Installation

Pour commencer, il faut que vous téléchargiez **le zip de l'application**, que vous le décompressiez et que vous ajoutiez hsqldb.jar (du répértoire lib) à votre classpath. C'est la seule chose à faire pour pouvoir accéder à votre base de données.

Il est intéressant de noter que ce zip contient une documentation très complète, des exemples et un petit soft graphique (voir plus loin) pour la création de requêtes.

Connexion à la base

Il faut commencer par charger le driver **JDBC** d'HSQL :

```
Class.forName("org.hsqldb.jdbcDriver").newInstance();
```

Ensuite, on ouvre une nouvelle connexion à la base de données. Remplacez database par le nom que vous voulez donner à votre base, si elle n'existe pas, elle sera automatiquement créée, sinon, elle va s'ouvrir. Le login est sa et il n'y a pas de mot de passe :

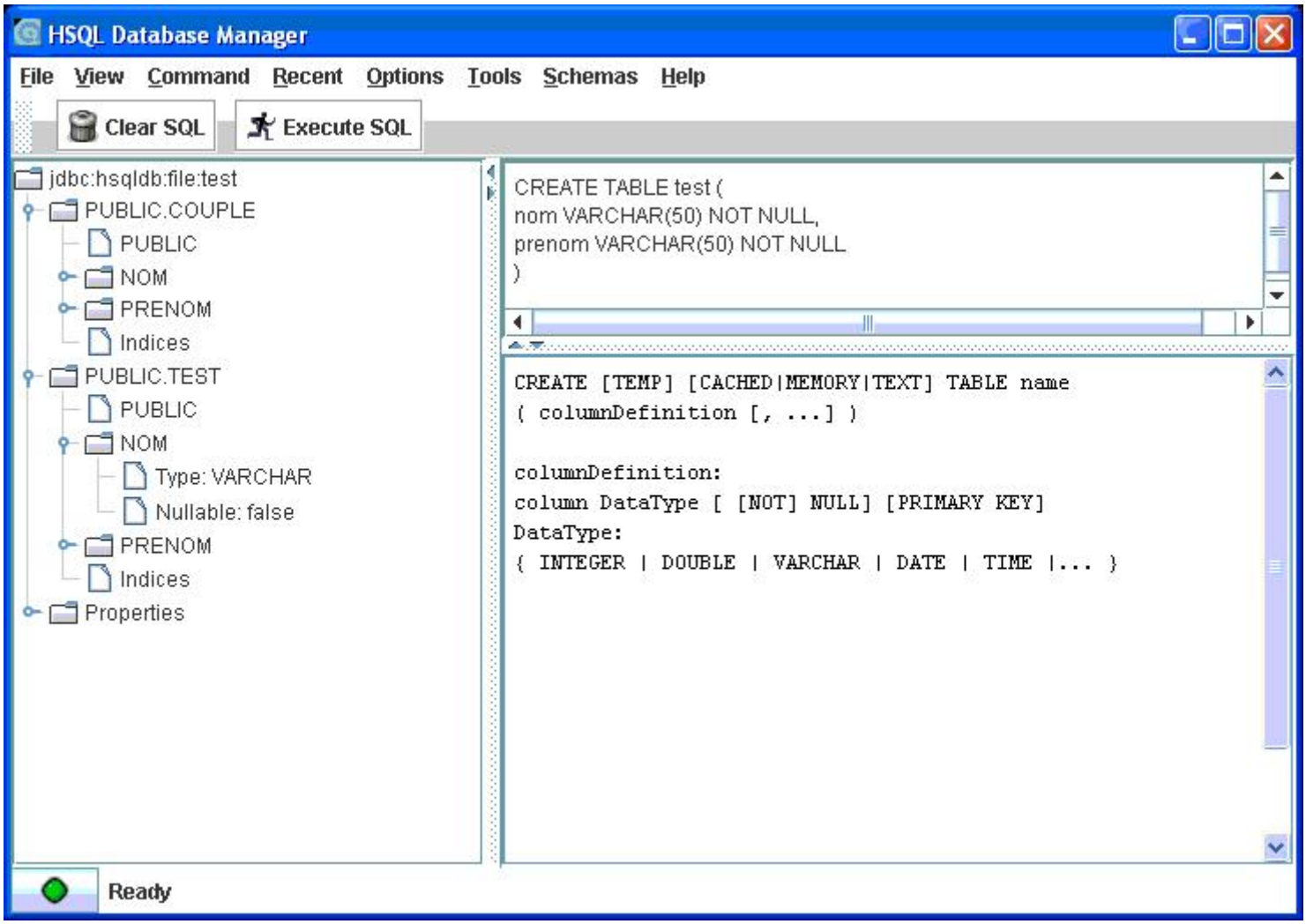
```
Connection connexion = DriverManager.getConnection("jdbc:hsqldb:file:database", "sa",  "");
```

On a aussi la possibilité de créer des tables entièrement en mémoire, donc avec aucun fichier sur le disque dur. Ces bases peuvent être utilisé à titre temporaire pour le stockage d'informations pendant une application. Pour faire cela, il suffit de spécifier mem à la place de file dans l'url de connexion :

```
Connection connexion = DriverManager.getConnection("jdbc:hsqldb:mem:database", "sa",  "");
```

HSQL Database Manager

Pour exécuter des requêtes sur votre base de données, vous n'êtes pas obligé de le faire par programmation, vous pouvez aussi utiliser HSQL Database Manager, qui est un outil que vous avez directement avec HSQL. Vous pouvez le lancer via "runManagerSwing.bat" qui se trouve dans le dossier demo.



HSQL Database Manager

Le grand intérêt de cet outil repose dans les exemples de requêtes SQL qu'il intègre, vous pourrez donc directement voir la syntaxe dans l'application. Vous pouvez ainsi exécutez facilement des requêtes sur votre base et voir clairement à quoi ressemble votre base.

On peut regretter l'absence de coloration syntaxique dans cet outil, mais il est quand même bien pratique. Par contre, il ne permet que de faire des requêtes, vous ne pourrez pas, à l'image d'Easy PHP, créer des tables via un assistant intuitif sans taper une ligne de code SQL, il vous faudra quand même écrire les requêtes, mais vous serez aidé par l'affichage de la syntaxe.

Fermeture de la connexion à la base

Pour dire à la base de données de sauvegarder toutes les données qu'elle a en mémoire sur le disque dur, il faut exécuter la commande SHUTDOWN :

```
Statement statement = connexion.createStatement();
statement.executeQuery("SHUTDOWN");
statement.close();
```

Et bien entendu fermer ensuite la connexion à la base de données :

```
connexion.close() ;
```

HSQL va écrire toutes ses données dans des fichiers (SCRIPT et PROPERTIES) et ensuite se fermer. Si vous n'exécutez pas le SHUTDOWN, vous allez perdre toutes les données de la session. Au démarrage, HSQL va faire le contraire, il va charger toutes les données des fichiers dans la mémoire.

Depuis HSQL 1.8, on peut aussi utiliser SET AUTOCOMMIT TRUE pour dire que dès que toutes les connexions à la base de données sont fermées, on clôt la base et on sauvegarde toutes les données.

Création d'une table

Pour la création d'une table, tout est similaire à la syntaxe SQL, mais je vais quand même reprendre les bases.

Il suffit d'employer un simple statement et sa méthode executeUpdate pour créer une table. Par exemple :

```
Statement statement = connexion.createStatement() ;
statement.executeUpdate("CREATE TABLE test (" +
 "colonne1 INT , colonne2 INT"
)");
```

Cela va tout simplement créer une table test avec deux colonnes de type int.

Pour créer une clé primaire, il vous suffit de mettre PRIMARY KEY après la clé et pour créer une clé à multiple colonne vous pouvez faire :

```
CREATE TABLE test(
 Colonne1 INT,
 Colonne2 INT,
 CONSTRAINT Colonne3 PRIMARY KEY(Colonne1, Colonne2)
)
```

Pour utiliser un index, il vous suffit d'ajouter INDEX(nomDeLaColonne) à la fin de la requête :

```
CREATE TABLE test(
 Colonne1 INT,
 Colonne2 INT,
 INDEX(Colonne1)
)
```

Pour que l'une de vos colonnes s'incrémente automatiquement à chaque nouvel enregistrement, il vous suffit d'utiliser cette syntaxe :

```
CREATE TABLE test(
 Colonne1 INT GENERATED BY DEFAULT AS IDENTITY(START WITH 1) PRIMARY KEY,
 Colonne2 INT,
)
```

Pour conserver l'intégrité référentielle de votre base, vous pouvez tout à fait employer les FOREIGN KEY de SQL qui sont gérées par HSQL. Par exemple :

```
CREATE TABLE tests(
 test_id INT PRIMARY KEY,
 test_nom VARCHAR(100)
)

CREATE TABLE others(
 test_id INT PRIMARY KEY,
 other_nom VARCHAR(100),
 FOREIGN KEY(test_id) REFERENCES tests (test_id)
)
```

Récupération de résultats

La récupération de résultats est toute simple, il suffit de créer un statement et d'employer la méthode executeQuery avec la requête en paramètre, qui va nous renvoyer un ResultSet.

```
Statement  statement = connexion.createStatement();
ResultSet resultat = statement.executeQuery("SELECT * FROM table WHERE condition");
```

On pourra ensuite la parcourir avec une boucle while :

```
while(resultat.next()){
 System.out.println(resultat.getXXX("colonne")) ;
}
```

Avec les méthodes getXXX, vous recevez directement un objet du type que vous voulez, les types sont directement des types Java, donc pas de problème de conversion.

Par contre, vous ne pourrez pas employer les méthodes last(), first(), beforeFirst(), ... avec HSQL. Ce que vous auriez pu faire avec d'autres base de données, mais on peut tout de même s'en passer sans trop de problèmes.

Modification, suppression et insertion

Les insertions, modifications ou suppressions de données se font aussi de manière tout à fait simple. Il suffit de créer un statement et d'employer la méthode executeUpdate avec votre requête passée en paramètre pour qu'elle soit exécutée dans la base de données.

```
Statement statement = connexion.createStatement() ;
statement.executeUpdate("INSERT INTO note (nom) VALUES ('Nul')");
```

Vous pouvez récupérer le résultat de executeUpdate pour savoir combien de lignes ont été affectées par la modification.

Aller plus loin

Je ne vais pas mettre ici toutes les possibilités d'HSQL, puisque cela reviendrait presque à expliquer toutes les commandes SQL. Par contre, la documentation est très bien faite, puisqu'elle liste toutes les commandes SQL supportées. Ce tutoriel explique déjà toutes les bases nécessaires pour débuter avec HSQL, si vous avez besoin de plus d'informations, vous pouvez vous référez à la documentation officielle.

III - Conclusion

Conclusion

L'utilisation d'HSQL n'est pas compliquée, il faut simplement se faire au fonctionnement en mémoire et comprendre quelques subtilités. Les possibilités d'HSQL sont presque aussi grandes que celles d'une base de données normale. HSQL est donc tout à fait adapté à une utilisation dans vos programmes, si vous avez besoin d'une base de données embarquée et pour un nombre pas trop élevé de données.

Remerciements

Merci à **elitost** et **le y@m's** pour leurs corrections ainsi qu'à **ChristopheJ** et **Ricky81** pour leurs commentaires.

Liens

- **Documentation de HSQL**
- **Télécharger HSQL**
- **SQL De A à Z par SqlPro**
- **La FAQ JDBC**