Ch. 1. Installer et utiliser Hsqldb

De Apache OpenOffice Wiki
< FR | Documentation | HSQLDB Guide

Sommaire

Chapitre 1. Installer et utiliser Hsqldb

Fred Toussi HSQLDB Development Group

<ft@cluedup.com (mailto:ft@cluedup.com)>

$Date: 2007/05/30 18:34:46 $

Contents

Introduction

Le paquet HSQLDB jar est stocké dans le dossier /lib et contient plusieurs composants et programmes. Pour chaque programme, différentes commandes sont disponibles.

Composants du paquet Hsqldb jar

- HSQLDB RDBMS
- Pilote HSQLDB JDBC
- Gestionnaire de bases de données (Database Manager) (versions Swing et AW)
- Outil de requêtes (Query Tool) (AWT)
- Sql Tool (ligne de commande)

Les composants HSQLDB RDBMS et Pilote JDBC fournissent les fonctionnalités de base. Les autres sont des outils de base de données généraux pouvant être utilisés avec n'importe quel moteur de base de données ayant un pilote JDBC.

Outils d'éxécution

(Running Tools)

Tous les outils peuvent être exécutés de façon standard pour les classes Java archivées. Dans l'exemple suivant la version AWT du gestionnaire de base de données, le fichier `hsqldb.jar` est dans le dossier `../lib` relatif au dossier courant.

```
   java -cp ../lib/hsqldb.jar org.hsqldb.util.DatabaseManager
```

Si le fichier `hsqldb.jar` était dans le dossier courant, la commande changerait en:

```
   java -cp hsqldb.jar org.hsqldb.util.DatabaseManager
```

Classes principales pour les outils Hsqldb

- `org.hsqldb.util.DatabaseManager`
- `org.hsqldb.util.DatabaseManagerSwing`
- `org.hsqldb.util.Transfer`
- `org.hsqldb.util.QueryTool`
- `org.hsqldb.util.SqlTool`

Certains outils, comme le gestionnaire de base de données ou SQL Tool, peuvent utiliser des arguments en ligne de commande ou entièrement dépendre d'elles. Vous pouvez ajouter l'argument -? à la ligne de commande pour obtenir une liste de tous les arguments disponibles pour ces outils. Parmi les fonctionnalités du gestionnaire de base de données figure une interface utilisateur graphique pour explorer celle-ci de façon interactive.

Exécution de Hsqldb

HSQLDB peut-être exécuté de différentes façons. Ces dernières sont en général réparties entre les modes serveurs et les modes "In-Process" (aussi appelés modes "Standalone"). Le fichier jar utilise un sous-programme différent pour exécuter HSQLDB dans chacun des modes.

Chaque base de données HSQLDB est constituées de 2 à 5 fichiers, tous nommés à l'identique mais avec des extensions différentes, et situés dans le même dossier. Par exemple, la base nommée "test" est constituée des fichiers suivants:

- `test.properties`
- `test.script`
- `test.log`
- `test.data`
- `test.backup`

Le fichier des propriétés contient des réglages généraux de la base de données. Le fichier script contient la définition des tables et d'autres objets de la base de données, plus les données pour les tables non en cache (non-cached tables). Le fichier log contient les changements récents de la base de données. Le fichier data contient les données pour les tables en cache et le fichier backup est un backup compressé (zip) du dernier état stable (consistent state) du fichier data. Tous ces fichiers sont essentiels et ne doivent jamais être enlevés. Si la base de données n'a pas de tables en cache, les fichiers `test.data` et `test.backup` ne seront pas présents. En plus de ces fichiers, une base de données HSQLDB peut être reliée à tout fichier texte formaté, telles des listes CSV, (CSV = Comma Separated Value = Données séparées par une virgule) n'importe ou sur le disque.

Pendant l'exécution de la base de données "test", le fichier `test.log` est utilisé pour écrire les changements faits aux données. Ce fichier est supprimé après une fermeture normale (voir SHUTDOWN). Autrement (après une fermeture

anormale), ce fichier est relu au démarrage suivant pour rétablir les changements. Un fichier `test.lck` est également utilisé pour enregistrer le fait que la base de données est ouverte. Il est effaçé après une fermeture normale. Dans certaines circonstances, un fichier `test.data.old` est créé et supprimé par la suite.

Note

Lorsque le moteur ferme la base de données, il créé des fichiers temporaires avec l'extension `.new`, qu'il renomme ensuite en une de celles listées ci-dessus.

Modes de serveur

Les modes de serveur fournissent une accessibilité maximale. Le moteur de la base de données s'exécute dans une JVM (Machine Virtuelle Java) et écoute les connexions de programmes sur le même ordinateur, ou d'autres sur le réseau. Plusieurs programmes différents peuvent se connecter au serveur et récupérer ou mettre à jour l'information. Les programmes clients (Applications programs (clients)) se connectent au serveur en utilisant le pilote HSQLDB JDBC. Dans la plupart des modes de serveur, le serveur peut traiter jusqu'à dix base de données simultanément. (the server can serve up to 10 databases that are specified at the time of running the server.)

Les modes de serveur peuvent utiliser des présélections de propriétés ou des arguments en ligne de commande, comme détaillé dans le chapitre Considérations avancées. Il y a trois modes de serveur, basés sur les protocoles de communication Client / Serveur.

Serveur Hsqldb

C'est la méthode préférée pour exécuter un serveur de bases de données. C'est également la plus rapide. Un protocole de communication propriétaire est utilisé dans ce mode. Une commande similaire à celles utilisées pour exécuter les outils (ou Running Tools ?) et décrite plus haut est utilisée pour lancer le serveur. L'exemple suivant de la commande pour lancer le serveur le démarre avec une base de données (par défaut) constituée de fichiers nommés "mydb.*".

```
   java -cp ../lib/hsqldb.jar org.hsqldb.Server -database.0 file:mydb -dbname.0 xdb
```

L'argument de ligne de commande `-?` fournit une liste des arguments disponibles.

Serveur Web Hsqldb

Ce mode est utilisé quand l'accès à l'ordinateur hébergeant le serveur de bases de données est restreint au protocole HTTP. L'unique raison pour utiliser le mode Serveur Web tient dans les restrictions imposées par les pare-feux sur des machines client ou serveur. Il ne devrait pas être utilisé quand il n'y a pas de telles restrictions. Le Serveur Web HSQLDB est un serveur web particulier qui permet aux clients JDBC de se connecter via HTTP. Depuis la version 1.7.2 ce mode supporte également les transactions.

Pour exécuter un serveur web, dans la ligne de commande ci-dessus, remplacez la classe principale pour le serveur par la suivante :

```
org.hsqldb.WebServer
```

L'argument de ligne de commande `-?` fournit une liste des arguments disponibles.

Servlet Hsqldb

Voir sur Wikipédia : http://fr.wikipedia.org/wiki/Servlet

Ce dernier mode utilise le même protocole que le mode Serveur Web. Il est utilisé quand une servlet d'un moteur séparé (ou du serveur de l'application) comme par exemple Tomcat ou Resin fournit un accès à la base de données. Le mode Servlet ne peut être démarré indépendamment de l'application servlet. La classe `Servlet`, dans le fichier HSQLDB.jar, doit être installé sur l'application du serveur pour fournir la connexion. La base de données est spécifiée par le biais d'une propriété de l'application coté serveur. Voir le fichier source `org.hsqldb.Servlet.java` pour plus de détails.

On ne peut accéder aux deux modes Serveur Web et Servlet qu'en utilisant le pilote JDBC côté client. Ils ne fournissent pas un frontal Web à la base de données. Le mode Servlet ne peut servir qu'une base de données.

Veuillez noter que vous ne pouvez pas normalement vous servir de ce mode si vous utilisez un moteur de base de données dans un serveur d'applications.

Se connecter à une base de données exécutée comme un serveur. Un fois qu'un serveur HSQLDB est actif, les programmes clients peuvent s'y connecter en utilisant le pilote HSQLDB JDBC contenu dans `hsqldb.jar`. L'information complète de "comment se connecter à un serveur" est fournie dans la documentation Java jdbcConnection (http://hsqldb.org/doc/src/org/hsqldb/jdbc/jdbcConnection.html)(située dans le dossier `/doc/src` de la distribution HSQLDB. Un exemple courant est la connexion au port par défaut (9001) utilisé par le protocole hsql sur une machine en local (the same machine).

Exemple_1.1. Code Java pour se connecter au serveur local ci-dessus mentionné

```
    try {
        Class.forName("org.hsqldb.jdbcDriver" );
    } catch (Exception e) {
        System.out.println("ERROR: failed to load HSQLDB JDBC driver.");
        e.printStackTrace();
        return;
    }

    Connection c = DriverManager.getConnection("jdbc:hsqldb:hsql://localhost/xdb", "sa", "");
```

Dans certaines circonstances, vous devrez utiliser la ligne suivante pour vous connecter au pilote :

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

A noter que dans l'URL de la connexion ci-dessus, il n'est pas fait mention du fichier de la base de données, comme c'était le cas pour démarrer le serveur. Au lieu de cela on utilise la valeur définie pour dbname.0. Aussi veuillez vous référer au chapitre Considérations avancées pour l'URL de la connexion quand il y a plus d'une base de données par instance de serveur.

Considérations de sécurité Quand HSQLDB est exécuté comme serveur, l'accès réseau doit être protégé en conséquence. Les adresses IP sources doivent être restreintes par l'usage de filtrage TCP, de programmes Pare-Feu ou de pare-feux indépendants. Si le trafic doit passer à travers un réseau non protégé (comme Internet), le flux doit être crypté (par exemple par VPN, ssh tunneling, ou TLS (http://hsqldb.org/doc/guide/ch07.html) en utilisant les variantes SSL enabled HSQLS et HTTPS des modes Serveur et Serveur Web). Seuls les mots de passe sécurisés doivent être employés, et d'une façon plus importante, le mot de passe de l'utilisateur par défaut doit être changé depuis la chaîne vide par défaut. Si vous fournissez intentionnellement des données au public, la connexion réseau largement ouverte au public (wide-open public network connection) doit être utilisée exclusivement pour accéder aux données publiques par des comptes en lecture seule. (c.-à -d. Que ni les données sécurisées ni les compte privilégiés ne doivent utiliser cette connexion). Ces considérations s'appliquent aussi aux serveurs HSQLDB exécutés avec le protocole HTTP.

Mode « In-Process » (Stand-alone)

Ce mode exécute le moteur de base de données comme une partie de votre programme d'application dans la même Machine Virtuelle Java (JVM). Pour la plupart des applications ce mode est plus rapide, puisque les données ne sont pas converties et envoyées sur le réseau. L'inconvénient principal est qu'il n'est pas possible par défaut de se connecter à la base de données en dehors de l'application. En conséquence vous ne pouvez vérifier le contenu de la base de données avec des outils externes comme un gestionnaire de base de données pendant que votre application "tourne". Depuis la version 1.8.0, vous pouvez lancer une instance de serveur dans un processus depuis la même machine virtuelle que votre application et ménager ainsi un accès externe à votre base

de données "In-Process".

La manière préconisée d'utiliser le mode In-Process dans une application est d'utiliser une instance de serveur HSQLDB pour la base de données tout en développant l'application, et ensuite basculer dans le mode In-Process pour le déploiement.

Une base de données dans le mode In-Process est exécutée depuis JDBC, avec le chemin du fichier de la base de données spécifié dans l'URL de connexion. Par exemple, si le nom de la base de données est testdb et que ses fichiers sont situés dans le même dossier que la commande délivrée pour lancer l'application, le code suivant sera utilisé pour la connexion :

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

Le format du chemin de fichier de la base de données peut être spécifié en utilisant le caractère slash (/) indifféremment sous les plates-formes Windows ou Linux. Ainsi les chemins relatifs ou les chemins qui se réfèrent au même dossier sur le même disque peuvent être identiques. Par exemple si le chemin sous Linux de votre base de données est `/opt/db/testdb` et que vous créez une arborescence identique sur le disque C: d'une plate-forme Windows, vous pouvez utiliser la même URL pour les deux systèmes.

```
   Connection c = DriverManager.getConnection("jdbc:hsqldb:file:/opt/db/testdb", "sa", "");
```

Lors de l'utilisation des chemins relatifs, ces derniers seront pris en compte depuis l'emplacement duquel la commande shell (http://fr.wikipedia.org/wiki/Shell_(informatique)) de démarrage de la JVM aura été exécutée. Se référer à la documentation javadoc avec jdbcConnection (http://hsqldb.org/doc/src/org/hsqldb/jdbc/jdbcConnection.html)pour plus de détails.

Bases de données en mémoire seule

Il est possible de lancer HSQLDB d'une manière où la base de données n'est pas persistante et existe seulement dans la RAM (Random Access Memory ou mémoire vive). Comme aucune information n'est écrite sur le disque, ce mode ne devrait servir qu'aux calculs internes des données de l'application, dans des applets (http://fr.wikipedia.org/wiki/Applet) ou certaines applications spécialisées. Ce mode est spécifié par le protocole mem: .

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

Vous pouvez également lancer une instance de serveur en mémoire seule en spécifiant la même URL dans `server.properties`. Cet usage est extraordinaire et limité à des applications spécifiques ou le serveur de base de données est utilisé uniquement pour échanger des informations entre les clients, ou pour des données non-persistantes.

Généralités

Fermer la base de données

Toutes les bases de données exécutées dans les différents modes peuvent être fermées avec la commande SHUTDOWN, traitée comme une commande SQL. Depuis la version 1.7.2, les bases de données "In-Process" ne sont plus fermées quand la dernière connexion à la base de donnée est explicitement fermée via JDBC, une commande SHUTDOWN est requise. Dans la 1.8.0, une propriété de connexion, shutdown=true, peut être spécifiée à la première connexion à la base de données (la connexion qui ouvre la base de données) pour forcer un shutdown quand la dernière connexion s'est fermée.

Quand une commande SHUTDOWN est transmise, toutes les transactions actives sont annulées (rolled back). Une méthode particulière de fermeture de la base de données est via la commande SHUTDOWN COMPACT. Cette commande ré-écrit le fichier `.data` qui contient l'information stockée dans les tables "CACHED" (mises en cache - voir Cache (http://fr.wikipedia.org /wiki/Cache)) et les compacte à leur taille minimum. Cette commande devrait être transmise périodiquement,et plus spécialement quand beaucoup d'insertions, de mises à jour ou de suppressions ont été réalisées sur les tables en cache. Les changements de structure de la base de données, comme supprimer ou modifier des tables en cache déjà peuplées d'enregistrements, ou effectuer des opérations sur les index créent également de larges espaces inutilisés dans les fichiers qui peuvent être récupérés par l'utilisation de cette commande.

Utiliser de multiples bases de données dans une machine virtuelle Java

Dans les exemples ci-dessus chaque serveur fournit seulement une base de données et seulement une base de données en mémoire peut être créée. Toutefois, depuis la version 1.7.2, HSQLDB peut servir plusieurs bases de données dans plusieurs modes de serveur et permet les accès simultanés à de multiples bases de données "in-process" et en mémoire seule. Ces possibilités sont décrites dans le chapitre Considérations avancées.

Créer une nouvelle base de données

Quand une instance de serveur est lancée, ou quand une connexion est faite à une base de données Â« in-process Â», une nouvelle base de données vide est créée s'il n'existe pas de base de données au chemin donné.

Cette caractéristique a pour effet secondaire de troubler de nouveaux utilisateurs. S'il y a une erreur dans la spécification du chemin à une base de données existante, une connexion est malgré tout établie à une nouvelle base de données. A des fins de dépannage, vous pouvez spécifier la propriété de connexion ifexists=`true` pour permettre seulement la connexion à une base de

données existante et éviter d'en créer une nouvelle. Dans ce cas, si la base de données n'existe pas, la méthode `getConnection()` retournera une exception.

Utiliser le moteur de la base de données

Une fois la connexion établie à une base de données et quel que soit le mode, on utilise les méthodes JDBC pour interagir avec la base de données. La documentation Java pour jdbcConnection (http://hsqldb.org/doc/src/org/hsqldb/jdbc/jdbcConnection.html)[1] (http://hsqldb.org/doc/src/org/hsqldb/jdbc/jdbcConnection.html), jdbcDriver (http://hsqldb.org/doc/src/org/hsqldb/jdbcDriver.html)[2] (http://hsqldb.org/doc/src/org/hsqldb/jdbcDriver.html), jdbcDatabaseMetadata (http://hsqldb.org/doc/src/org/hsqldb/jdbc/jdbcDatabaseMetaData.html)[3] (http://hsqldb.org/doc/src/org/hsqldb/jdbc/jdbcDatabaseMetaData.html), jdbcResultSet (http://hsqldb.org/doc/src/org/hsqldb/jdbc/jdbcResultSet.html)[4] (http://hsqldb.org/doc/src/org/hsqldb/jdbc/jdbcResultSet.html), jdbcStatement (http://hsqldb.org/doc/src/org/hsqldb/jdbc/jdbcStatement.html)[5] (http://hsqldb.org/doc/src/org/hsqldb/jdbc/jdbcStatement.html), et jdbcPreparedStatement (http://hsqldb.org/doc/src/org/hsqldb/jdbc/jdbcPreparedStatement.html)[6] (http://hsqldb.org/doc/src/org/hsqldb/jdbc/jdbcPreparedStatement.html)liste toutes les méthodes JDBC supportées ensemble et l'information spécifique à HSQLDB. Les méthodes JDBC sont grossièrement réparties ainsi : Méthodes relatives à la connexion, méthodes de métadonnées et méthodes d'accès à la base de données. Ces dernières utilisent les commandes SQL pour agir sur la base de données et retournent les résultats soit comme un type primitif Java (? Java primitive type) ou comme une instance de la classe `java.sql.ResultSet` .

Vous pouvez utiliser un gestionnaire de base de données ou d'autres outils Java d'accès aux bases de données pour consulter votre base de données et la mettre à jour par des commandes SQL. Ces programmes utilisent un pilote JDBC interne pour envoyer vos commandes au moteur de la base de données et afficher les résultats dans un format intelligible.

HSQLDB utilise un langage SQL aussi proche que possible des normes SQL92 et SQL200n tout en gardant un moteur de base de données léger.

La liste complète des commandes SQL est traitée au chapitre Syntaxe SQL.

Les différents types de tables

HSQLDB supporte les tables temporaires TEMP et trois types de tables persistantes.

Les tables TEMP ne sont écrites sur disque que pendant la durée de vie de l'objet Connexion. Le contenu de chaque table TEMP est visible uniquement depuis la connexion utilisée. Depuis la 1.8.0 la définition des tables TEMP est conforme au type GLOBAL TEMPORARY du standard SQL. La définition de la table persiste mais toutes les nouvelles connexions ne voient que leurs propres copies de cette table (qui est vide au départ). Lorsque la connexion est lancée,

le contenu est nettoyé par défaut. Si les déclarations de définition de table incluent ON COMMIT PRESERVE ROWS, le contenu sera préservé lors de l'exécution d'un commit.

Les trois types de tables persistantes sont : MEMORY , CACHED et TEXT, appelés également ici "table mémoire", "table en cache" ou "table texte".

Le type par défaut de la commande CREATE TABLE génère des tables mémoire. Le contenu de ces dernières est entièrement en mémoire vive mais chaque changement à la structure ou aux données est immédiatement reflété dans le fichier <NomDeLaBase>.script . Le fichier de script est lu à l'ouverture de la base de données, et les tables mémoire sont recréées avec tout leur contenu. Contrairement aux tables temporaires, les tables mémoire (la valeur par défaut) sont persistantes.

Les tables en cache sont créées par la commande CREATE CACHED TABLE. Seule une partie des données ou des index réside en mémoire, permettant de grandes tables qui occuperaient autrement plusieurs centaines de méga-octets de mémoire. Un autre avantage des tables en cache est que le moteur de la base de données met moins de temps à démarrer dans le cas de grandes tables. Le désavantage des tables en cache est la réduction de vitesse. N'utilisez pas de tables en cache si l'ensemble de vos données est relativement petit. Dans une application constituée de grandes et de petites tables, il est plus efficient d'utiliser la valeur par défaut (le mode MEMORY) pour les petites tables.

Les tables texte sont prises en charge depuis la version 1.7.0 et utilisent le format CSV (Comma Separated Value = Valeurs séparées par une virgule) ou un autre fichier texte délimité comme source de données. Vous pouvez choisir un fichier CSV existant, tel un export d'une autre base de données ou d'un programme, comme source de la table texte. Vous pouvez également créer un fichier vide qui sera peuplé par le moteur de la base de données. Les tables texte sont performantes en termes d'usage de la mémoire puisqu'elles mettent en cache seulement une partie des données texte et tous les index. La source de données de la table texte peut toujours être réassignée à un autre fichier si nécessaire. Deux commandes sont requises pour installer une table texte, comme détaillé dans le chapitre Tables texte.

Avec les base de données en mémoire seule (voir plus haut), les deux déclarations de tables MEMORY et CACHED sont traitées comme des déclarations de tables mémoire non persistantes. Les déclarations de tables texte ne sont pas permises dans ce mode.

Contraintes et Index

HSQLDB prend en charge les contraintes PRIMARY KEY, NOT NULL, UNIQUE, CHECK et FOREIGN KEY. De plus, il supporte les index UNIQUE ou ordinaires. Cette prise en charge est relativement importante et couvre les contraintes multi-colonnes, les index, plus les mises à jour et suppressions en cascade pour

les clés externes.

HSQLDB créé des index internes pour prendre en charge les contraintes PRIMARY KEY, UNIQUE et FOREIGN KEY : Un index unique est créé pour chaque contrainte PRIMARY KEY ou UNIQUE ; un index ordinaire est créé pour chaque contrainte de clé externe (FOREIGN KEY). Par voie de conséquence, vous n'avez pas à créer d'index dupliqué défini par l'utilisateur sur le même jeu de colonnes couvert par ces contraintes. Ceci résulterait en une consommation inutile de la mémoire et de la puissance de calcul. Voir le chapitre "Problèmes liés au SQL" pour plus d'information.

Les index sont cruciaux pour une vitesse adaptée des requêtes. Lors de l'utilisation de requêtes joignant plusieurs tables, il doit y avoir un index sur chaque colonne jointe de chaque table. Si on utilise des conditions de plages de valeurs ou d'égalité, par exemple `SELECT ... WHERE acol >10 AND bcol = 0`, un index est requis sur la colonne acol utilisée dans cette condition. Les index n'ont pas d'effet sur les clause ORDER BY ou certaines conditions LIKE.

En règle générale, HSQLDB possède une vitesse de traitement interne des requêtes de plus de 100 000 lignes par seconde. Toute requête s'exécutant en plusieurs secondes devrait être vérifiée et des index devraient être ajoutés aux colonnes des tables concernées si nécessaire.

Prise en charge du SQL

La syntaxe SQL prise en charge par HSQLDB est essentiellement celle spécifiée par les normes SQL (92 et 200n). Toutes les fonctionnalités de la norme ne sont pas supportées et il y a certaines extensions propriétaires. Dans la version 1.8.0 le comportement du moteur est beaucoup plus conforme aux standards que dans les versions antérieures. Les principaux changements sont :

- Traitement correct de la valeur NULL dans les colonnes de jointure, les contraintes UNIQUE et les conditions de requêtes
- Traitement correct des requêtes sélection avec JOIN et LEFT OUTER JOIN
- Traitement correct des fonctions d'agrégations contenues dans les expressions ou contenant des arguments d'expression

Les commandes prises en charge sont listées dans le chapitre Syntaxe SQL. Pour un guide bien écrit pour SQL agrémenté d'exemples vous pouvez consulter PostgreSQL: Introduction and Concepts (http://www.postgresql.org/files/documentation/books/aw_pgsql/index.html) de Bruce Momjian, qui est disponible sur le web. La majorité des champs d'applications SQL couverts dans ce livre s'appliquent aussi à HSQLDB. Il y a des différences entre les mots-clés pris en charge par l'un et l'autre moteur (OUTER, OID's, etc.) et des usages différents (IDENTITY/SERIAL, TRIGGER, SEQUENCE, etc.).

Prise en charge de JDBC

Depuis la version 1.7.2, le support de JDBC2 a été sensiblement étendu et certaines fonctions de JDBC3 sont également prises en charge. Les classes qui s'y rapportent sont minutieusement documentées. Voyez la documentation Java pour les classes org.hsqldb.jdbcXXXX (http://hsqldb.org/doc/src/index.html).
