Ch. 4. Considérations avancées

De Apache OpenOffice Wiki
< FR | Documentation | HSQLDB Guide

Chapitre 4.

Sommaire

Considérations avancées

Fred Toussi

HSQLDB Development Group

<ft@cluedup.com>

$Date: 2007/03/24 11:39:08 $

But de ce document

Beaucoup de questions fréquemment posées dans les forums et mailing-lists trouvent leurs réponses dans ce guide. Si vous voulez utiliser HSQLDB avec votre application, vous devez lire ce guide. Ce document couvre les problèmes relatifs au système. Pour les problèmes relatifs au SQL voyez le chapitre : Problèmes liés au SQL.

Connexions

La méthode normale d'accès à une base de données HSQLDB est par une interface de connexion JDBC. Un point de départ aux différentes méthodes

pour fournir les services des bases de données et y accéder se situe au chapitre Problèmes liés au SQL. Des détails et exemples sur la connexion JDBC sont fournis dans JavaDoc for `jdbcConnection` (http://hsqldb.org/doc/src/org/hsqldb/jdbc/jdbcConnection.html).

La version 1.7.2 a introduit une méthode uniforme de distinction entre différents types de connexions, abordant de nouvelles possibilités pour fournir l'accès à de multiples bases de données. L'identifiant de pilote commun est `jdbc:hsqldb:` suivi par un identifiant de protocole (`mem: file: res: hsql: http: hsqls: https:`) puis suivi par l'hôte et les identifiants de port dans le cas de serveurs, et enfin suivi de l'identifiant de la base de données.

Tableau 4.1. Composants des URL Hsqldb

Pilote et Protocole	Machine hôte et Port	Base de données
`jdbc:hsqldb:mem:`	non disponible	`accounts`
En minuscule, un identifiant d'un simple mot crée la base de données en mémoire lors de la première connexion. Par la suite, l'usage de la même URL de connexion se connecte à la base de données existante. L'ancienne forme de l'URL, `jdbc:hsqldb:.` crée ou se connecte à la même base de données que la nouvelle forme de l'URL, `jdbc:hsqldb:mem:.`		

`jdbc:hsqldb:file:`	non disponible	`MaBase`
		`/opt/db/accounts`
		`C:/data/MaBase`

Le chemin spécifie le fichier de la base de données. Dans les exemples ci-dessus le premier fait référence à un jeu de fichier MaBase.* dans le dossier d'où la commande Java pour lancer l'application a été émise. Le deuxième et le troisième exemple référencent les chemins absolus sur la machine hôte.		
`jdbc:hsqldb:res:`	non disponible	`/UnDossier` `/NomDeLaBase`
Les fichiers de base de données peuvent être chargés depuis un des jars spécifiés comme une partie de la commande Java de la même manière qu'on accède aux fichiers de ressources dans les programmes Java. Le dossier `/UnDossier` ci-dessus correspond à un répertoire dans un des jars.		

`jdbc:hsqldb:hsql:`
`jdbc:hsqldb:hsqls:`
`jdbc:hsqldb:http:`
`jdbc:hsqldb:https:`
	`//localhost`
	`//192.0.0.10:9500`
	`//dbserver.somedomain.com`
		`/an_alias`
		`/enrollments`
		`/quickdb`

L'hôte et le port spécifient l'adresse IP ou le nom de l'hôte du serveur et un numéro de port optionnel. La base de données à laquelle se connecter est		

spécifiée par un alias. Cet alias est une chaîne de caractères en minuscule définie dans le fichier des propriétés du serveur pour pointer sur une base de données existante sur le système de fichiers d'un serveur, ou une base de données en mémoire, éphémère, sur le serveur. Les lignes de l'exemple suivant dans `server.properties` ou `webserver.properties` définissent les alias de base de données définis ci-dessus et accessibles aux clients pour faire référence aux différents fichiers et base de données en mémoire.

```
database.0=file:/opt/db/accounts

dbname.0=an_alias



database.1=file:/opt/db/mydb

dbname.1=enrollments



database.2=mem:adatabase

dbname.2=quickdb
```

L'ancienne forme pour l'URL de serveur, par ex. `jdbc:hsqldb:hsql//localhost` se connecte à la même base de données que la nouvelle forme d'URL `jdbc:hsqldb:hsql//localhost/` ou l'alias est une chaîne de caractères de longueur zéro. Dans l'exemple ci-dessous, les fichiers de base de données `lists.*` dans le répertoire `/home/dbmaster/` sont associés avec un alias vide :

```
database.3=/home/dbmaster/lists

dbname.3=
```

Propriétés de Connexion

Chaque nouvelle connexion JDBC à une base de données peut spécifier des propriétés de connexion. Les propriétés Utilisateur et Mot de passe sont toujours requises. Dans la version 1.8.0 on peut également utiliser les propriétés optionnelles suivantes.

Les propriétés de connexion peuvent être spécifiées soit en établissant la connexion par l'appel de la commande :

```
DriverManager.getConnection (String url, Properties info);
```

Soit la propriété peut être ajoutée à l'URL complet de connexion.

Tableau 4.2. Propriétés de connexion

get_column_name	`true`	Nom de colonne dans le ResultSet
Cette propriété est utilisée pour la compatibilité avec l'implémentation d'autres pilotes JDBC. Si vrai (la valeur par défaut), `ResultSet.getColumnName(int c)` renvoie le nom de la colonne sous-jacente. Quand la valeur est fausse, la méthode ci-dessus renvoie la même valeur que `ResultSet.getColumnLabel(int column)`. Exemple ci-dessous : `jdbc:hsqldb:hsql://localhost/enrollments;get_column_name=false` Quand on utilise une procédure stockée définie par l'utilisateur à l'intérieur d'un ResultSet, la valeur par défaut, vrai, est la seule utilisée pour cette propriété.		
ifexists	`false`	Etablit la connexion seulement si la base de données existe déjà.
A un effet seulement avec les bases de données `mem:` et `file:`. Quand la valeur est vrai, ne créera pas de nouvelle base de données s'il n'en existe pas déjà une pour l'URL donnée. Quand la valeur est faux (par défaut), une nouvelle base de données `mem:` ou `file:` sera créée si elle n'existe pas déjà. Régler la propriété à vrai est pratique en cas de dépannage comme aucune base de données n'est créée si l'URL est mal formée. Exemple ci-dessous : `jdbc:hsqldb:file:enrollments;ifexists=true`		
shutdown	`false`	Ferme la base de données quand la dernière connexion est fermée.
Ceci imite le comportement des versions 1.7.1 et plus anciennes. Quand la dernière connexion à une base de données est close, la base de données est automatiquement fermée. La propriété prend effet seulement lors de la première connexion à la base de données. C'est à dire la connexion qui ouvre la base de données. Elle n'a pas d'effet, si utilisée, sur les connexions ultérieures ou simultanées. Cette commande a deux usages. L'un est pour les suites de tests, où les connexions à une base de données sont réalisées à partir d'un contexte d'une machine virtuelle Java, et immédiatement suivies par un autre contexte. L'autre utilisation est pour les applications où il n'est pas facile de configurer l'environnement de fermeture de la base de données. Les exemples		

rapportés par les utilisateurs incluent des serveurs d'applications web, où la fermeture de la dernière connexion coïncide avec la fermeture de l'application web.

De plus, quand la connexion à une base de données "in-process" crée une nouvelle base de données, ou en ouvre une existante (par exemple si c'est la première connexion à la base de données faite par l'application), toutes les propriétés de la base de données définies par l'utilisateur peuvent être spécifiées comme des propriétés URL. On peut l'utiliser pour spécifier des propriétés qui forcent un langage SQL plus strict, ou pour changer la taille du cache ou des propriétés similaires avant que les fichiers de la base de données ne soient créés. Toutefois, dans le cas des nouvelles bases de données, il est recommandé d'utiliser la commande SET PROPERTY pour de tels réglages.

Fichiers de propriétés

HSQLDB s'en remet à un jeu de fichiers de propriétés pour différents réglages. Depuis la version 1.7.0 la nomenclature des propriétés a été rationalisée et nombre de nouvelles propriétés ont été introduites.

Dans tous les fichiers de propriétés, les valeurs sont sensibles à la casse. Toutes les valeurs en dehors des noms de fichiers ou de pages sont requises en minuscule (c.a.d. server.silent=`FALSE` n'aura pas d'effet, mais server.silent=`false` fonctionnera).

Les fichiers de propriétés et les paramètres stockés en interne sont les suivants :

Tableau 4.3. Fichiers de propriétés Hsqldb Server

Nom du fichier	Adresse	Fonction
`server.properties`	Le répertoire d'où vient la commande pour exécuter la classe `Server`	Les réglages pour exécuter HSQLDB comme un serveur de base de données communicant par le protocole HSQL
`webserver.properties`	Le répertoire d'où vient la commande pour exécuter la classe `WebServer`	Les réglages pour exécuter HSQLDB comme un serveur de base de données communicant par le protocole HTTP
`<dbname>.properties`	Le répertoire où sont situés tous les fichiers pour une base de données	Réglages pour chaque base de données individuelle

Les fichiers de propriétés pour exécuter les serveurs ne sont pas créés

automatiquement. Vous devez créer vos propres fichiers contenant les paires server.property=`value` pour chaque propriété.

Le fichier des propriétés de chaque base de données est généré par le moteur de base de données. Ce fichier peut être édité après fermeture de la base de données. Dans la version 1.8.0, la plupart de ces propriétés peuvent être changés via les commandes SQL.

Les propriétés de Serveur et de Serveur Web

Dans les deux fichiers `server.properties` et `webserver.properties`, les valeurs prises en charge et leur valeur par défaut sont comme suit :

Tableau 4.4. Propriétés d'un fichier de propriétés

Valeur	Par défaut	Description
server.database.0	`test`	Chemin et nom de fichier du premier fichier de base de données à utiliser
server.dbname.0	`""`	Alias du serveur en minuscule pour le premier fichier de base de données
server.urlid.0	`NONE`	L'urlid de SqlTool utilisée par le script d'initialisation UNIX. (Cette propriété n'est pas utilisée si vous exécutez le mode Server/Webserver sur une plate-forme autre qu'UNIX, ou si vous n'utilisez pas notre script d'initialisation UNIX).
server.silent	`true`	Pas de messages explicites affichés sur la console.
server.trace	`false`	Messages de "trace" JDBC affichés sur la console.

Dans la version 1.8.0, chaque serveur peut servir jusqu'à dix bases de données différentes simultanément. La propriété server.database.0 définit le nom de fichier / chemin alors que server.dbname.0 définit l'alias en minuscule utilisé par les clients pour se connecter à cette base de données. Le chiffre 0 est incrémenté pour la deuxième base de données, et ainsi de suite. Les valeurs pour la propriété server.database.{0-9} peuvent utiliser les préfixes `mem:`, `file:` ou `res:` et les propriétés sont décrites plus haut dans la section Connexions. Par exemple,

```
database.0=mem:temp;sql.enforce_strict_size=true;
```

Les valeurs spécifiques à server.properties sont :

Tableau 4.5. Propriétés du fichier Propriété Serveur

Valeur	Par défaut	Description
server.port	`9001 (normal) or 554 (if TLS encrypted)`	Port TCP/IP utilisé pour échanger avec les clients. Toutes les bases de données sont servies sur le même port.
server.no_system_exit	`true`	Pas d'appel `System.exit()` lors de la fermeture de la base de données.

Les valeurs spécifiques à webserver.properties sont :

Tableau 4.6. Propriétés du fichier Propriété WebServeur

Valeur	Par défaut	Description
server.port	`80`	Port TCP/IP utilisé pour échanger avec les clients.
server.default_page	`index.html`	La page web par défaut pour le serveur.
server.root	`./`	Les adresses des pages fournies.
.<extension>	`?`	Les entrées multiples telles que .html=text/html définissent les types mime des fichiers statiques fournis par le serveur web. Voyez la source de `WebServer.java` pour une liste.

Toutes les valeurs ci-dessus peuvent être spécifiées en ligne de commande de démarrage du serveur en omettant le préfixe `server`.

Démarrer un serveur depuis votre application

Si vous voulez démarrer le serveur depuis votre application, sans ligne de commande ni fichier batch, vous devez créer une instance du serveur ou du serveur web, puis assigner les propriétés sous forme d'une chaîne de caractères et enfin démarrer le serveur. Vous en avez un exemple dans `org.hsqldb.test.TestBase source.`

- **Note**

Mise à jour : Si vous avez déjà des fichiers de propriétés personnalisés,

accordez les valeurs à la nouvelle convention de dénomination. Notez l'usage des chiffres à la fin des propriétés de server.database.n et server.dbname.n.

Propriétés d'une base de données individuelle

Chaque base de données a son propre fichier `<NomDeLaBase>.properties`, faisant lui-même partie d'un petit groupe de fichiers comprenant aussi `<NomDeLaBase>.script` et `<NomDeLaBase>.data`. Les fichiers de propriétés contiennent des paires Clé/Valeur pour quelques paramètres importants.

Dans la version 1.8.0 une nouvelle commande SQL permet à la plupart des propriétés de base de données d'être modifiées comme suit :

```
SET PROPERTY "property_name" property_value
```

Les propriétés qui peuvent être modifiées par la commande SET PROPERTY sont indiquées dans le tableau ci-dessous. D'autres propriétés sont indiquées comme `PROPERTIES FILE ONLY` et peuvent être modifiées seulement par édition du fichier .properties après une fermeture et avant un redémarrage. Seules les valeurs définies par l'utilisateur listées ci-dessous devraient jamais être modifiées. Le changement de toute autre valeur pourrait résulter en des dysfonctionnements inattendus dans les opérations de la base de données. La plupart de ces valeurs ont été introduites en vue de nouvelles fonctionnalités depuis la version 1.7.0 :

Tableau 4.7. Database-specific Property File Properties ???

Valeur	Par défaut	Description
readonly	`false`	Toute la base de données est en lecture seule.
Si la propriété est Vrai, la base de données ne peut pas être modifiée pendant l'utilisation. Ce réglage peut être changé en `true` si la base de données doit être ouverte depuis un CD. Avant de changer ce réglage, la base de données doit être fermée avec la commande `SHUTDOWN COMPACT` pour assurer la consistance et la compacité des données. `(PROPERTIES FILE ONLY) mais peut être utilisé comme une propriété de connexion pour ouvrir une base de données normale en lecture seule.`		
hsqldb.files_readonly	`false`	Les fichiers de la base de données ne seront pas en écriture.
Si la propriété est Vrai, les données dans les tables MEMORY peuvent être modifiées et de nouvelles tables MEMORY peuvent être ajoutées. Toutefois ces changements ne sont pas sauvegardés quand la base de données est close. Les tables CACHED et TEXT sont toujours en lecture seule quand ce		

réglage est Vrai. (PROPERTIES FILE ONLY)		
hsqldb.cache_file_scale	1	Etablit la taille limite du fichier de données. Une fois réglée, cette limite peut être repoussée jusqu'à 8 Go.
Cette propriété peut-être réglée à 8 pour augmenter la taille limite du fichier .data de 2 à 8 Go. Pour appliquer le changement à une base de données existante, il faut d'abord exécuter la commande SHUTDOWN SCRIPT, puis la ligne Propriété=Valeur ci-dessous doit être ajoutée au fichier .properties avant de rouvrir la base de données. `hsqldb.cache_file_scale=8` La propriété peut-être réglée avec la commande SQL (contrairement au changement de la valeur dans le fichier des propriétés) quand la base de données n'a pas de tables en cache (CACHED) (par ex. une nouvelle base de données). (SET PROPERTY) The property can be set with the SQL command (as opposed to changing the value in the properties file) when the database has no CACHED tables (e.g. a new database). (SET PROPERTY)		
sql.enforce_size	false	Taille et remplissage (trimming and padding) des colonnes de texte
Cette propriété n'est plus prise en charge. Utilisez sql.enforce_sctrict_size		
sql.enforce_strict_size	false	Renforcement de la taille et remplissage des colonnes de texte
Se conforme aux standards SQL pour la taille et la précision du type des données. Quand la propriété est Vrai, toutes les valeurs CHARACTER, VARCHAR, NUMERIC et DECIMAL contenues dans une ligne affectée par une déclaration INSERT INTO ou UPDATE sont confrontées avec la taille spécifiée dans la définition SQL de la table. Il se produit une exception si la valeur est trop grande. De plus toutes les valeurs CHARACTER qui sont plus courtes que la taille spécifiée sont remplies avec des espaces. Les valeurs TIMESTAMP(0) et TIMESTAMP(6) sont également permises de façon à spécifier la résolution des fractions de seconde des valeurs. Quand la propriété est fausse (par défaut), stocke la chaîne exactement comme elle est insérée. (SET PROPERTY)		
sql.tx_no_multi_rewrite	false	Gestion d'une transaction
Dans le mode par défaut READ_UNCOMMITED, une transaction peut ré-écrire des lignes insérées ou mises à jour par une autre transaction non validée (commit). Régler cette propriété à Vrai générera une exception quand une		

telle écriture sera tentée. (SET PROPERTY)		
hsqldb.cache_scale	14	Exposant du cache mémoire
Indique le nombre maximum de lignes des tables en cache contenues en mémoire, calculé comme 3*(2^Valeur) (trois multiplié par (deux puissance Valeur)). La valeur par défaut résulte en 3*16384 lignes de toutes les tables en cache contenues en mémoire à chaque instant. La plage de valeurs est comprise entre 8 et 18. (SET PROPERTY). Si la valeur est réglée via SET PROPERTY elle devient effective après la prochaine commande de base de données SHUTDOWN ou CHECKPOINT. (SET PROPERTY)		
hsqldb.cache_size_scale	10	Exposant du cache mémoire
Indique la taille moyenne de chaque ligne dans le cache mémoire utilisé avec les tables en cache, calculé comme 2^Valeur (deux à la puissance Valeur). Ce résultat est multiplié par le nombre maximum de lignes définit par hsqldb.cache_scale pour former le nombre maximum d'octets pour toute les lignes dans le cache mémoire. La valeur par défaut donne un résultat de 1024 octets par ligne. Cette valeur, combinée avec le nombre de lignes par défaut, aboutit au fait qu'approximativement 50 Mo du fichier .data est stocké dans le cache mémoire. La plage de valeurs est comprise entre 6 et 20. (SET PROPERTY). Si la valeur est réglée via SET PROPERTY elle devient effective après la prochaine commande de base de données SHUTDOWN ou CHECKPOINT. (SET PROPERTY)		
hsqldb.log_size	200	Taille de l'historique (log) quand un checkpoint est exécuté.
La valeur est la taille en mégaoctets que le fichier .log peut atteindre avant qu'un checkpoint automatique n'intervienne. Un checkpoint ré-écrit le fichier .script et vide le fichier .log. La valeur peut être changée via la commande SQL SET LOGSIZE nnn.		
runtime.gc_interval	0	forced garbage collection (activation du "ramasse miette".)
This setting forces garbage collection each time a set number of result set row or cache row objects are created. The default, "0" means no garbage collection is forced by the program. This should not be set when the database engine is acting as a server inside an exclusive JVM. The setting can be useful when the database is used in-process with the application with some Java Runtime Environments (JRE's). Some JRE's increase the size of the memory heap before doing any automatic garbage collection. This setting would prevent any unnecessary enlargement of the heap. Typical values for this setting would probably be		

between 10,000 to 100,000. (PROPERTIES FILE ONLY)

une proposition : Ce paramètre déclenche une routine de suppression d'objets plus utilisés, afin de libérer de la mémoire. cf. cet article : http://gfx.developpez.com/tutoriel/java/gc/.

hsqldb.nio_data_file	`true`	Utilisation des méthodes d'accès nio pour le fichier .data
Quand HSQLDB est compilé et exécuté depuis Java 1.4 ou ultérieur, régler cette propriété à `false` évitera l'usage des méthodes d'accès nio, avec pour résultat une vitesse quelque peu réduite. Si le fichier de données dépasse les 256 Mo lors de la première ouverture, les méthodes d'accès nio ne sont pas utilisées. De plus, si le fichier devient plus gros que la quantité de mémoire de l'ordinateur qui doit être allouée pour l'accès nio, les méthodes d'accès non-nio sont utilisées. `(SET PROPERTY)`. Si cette propriété est utilisé avant de définir une table en cache, elle s'applique à la session en cours, sinon elle prend effet après un SHUTDOWN et redémarrage ou un CHECKPOINT.		
hsqldb.default_table_type	`memory`	type de table créé avec CREATE TABLE sans qualifiant
La commande CREATE TABLE créé une table mémoire par défaut. Définir la valeur "cached" pour cette propriété résultera en la création d'une table en cache par défaut. D'autres qualifiants comme CREATE MEMORY TABLE ou CREATE CACHED TABLE ne sont pas du tout affectés par cette propriété. `(SET PROPERTY)`		
hsqldb.applog	`0`	Niveau de l'historique (journal) de l'application
Le niveau par défaut zéro indique : Pas de journal. Le niveau un archive les événements relatifs à la persistance, incluant tous les échecs. Ces événements sont journalisés dans un fichier terminant par .app.log		
textdb.*	`0`	Propriétés par défaut pour les nouvelles tables texte
Propriétés qui prévalent sur celles par défaut du moteur de base de données pour les tables texte nouvellement créées. Les réglages de la commande `SET <NomDeLaTable> SOURCE <chaîne source>` de la table texte annulent les deux réglages par défaut et du moteur et des propriétés de la base de données. Les propriétés individuelles textdb.* sont énumérées dans le chapitre Tables texte. `(SET PROPERTY)`		

Quand la connexion à une base de données "in-process" créé une nouvelle base

de données, ou en ouvre une existante (par ex. la première connexion à la base de données par l'application), toutes les propriétés de la base de données définies par l'utilisateur listées dans cette section peuvent être précisées comme des propriétés URL.

Note

Mise à jour : Depuis la version 1.7.0, les adresses des fichiers de la base de données ne peuvent plus être outrepassées par les chemins définis dans le fichier des propriétés. Tous les fichiers appartenant à une base de données doivent résider dans le même répertoire.

La propriété sql.compare_in_locale=true n'est plus prise en charge. Si la ligne existe dans un fichier .properties, elle basculera la base de données dans l'assemblage par défaut en cours. Voir la commande SET DATABASE COLLATION[2]

Dans le cas de l'utilisation avec OpenOffice.org, certaines valeurs par défaut des propriétés seront différentes. Les propriétés et valeurs sont :

- hsqldb.default_table_type=cached
- hsqldb.cache_scale=13
- hsqldb.log_size=10;
- hsqldb.nio_data_file=false
- sql.enforce_strict_size=true

Commandes SQL pour les propriétés de base de données

Quelques propriétés de base de données sont établies avec des commandes SQL dédiées commençant par SET.

Tableau 4.8. Propriétés de commandes SQL

SET WRITE_DELAY {{TRUE | FALSE} | <secondes> | <millisecondes> MILLIS
La valeur par défaut est TRUE et indique que les changements journalisés à la base de données sont synchronisés au système de fichiers toutes les 20 secondes. FALSE indique qu'il n'y a pas de délai et à chaque validation une opération de synchronisation de fichier est accomplie. Des valeurs numériques depuis 0 peuvent également être spécifiées pour le délai de synchronisation. Le but de cette commande est de contrôler la quantité de données perdues en cas de plantage total du système. Un délai d'une seconde signifie qu'au plus les données écrites sur disque la dernière seconde avant le crash sont perdues. Toutes les données écrites préalablement ont été synchronisées et devraient être récupérables.

Ce réglage devrait être défini sur la base de la fiabilité du matériel utilisé pour exécuter le moteur de la base de données, le type de disque système utilisé, l'éventualité de panne d'alimentation, etc. De même la nature des donnée stockées doit être considérée. En général, quand le système est très fiable, le réglage peut être laissé à sa valeur par défaut. S'il n'est pas très stable ou que les données sont critiques, un réglage d'une ou deux secondes devrait suffire. Seulement dans les pires scénarios ou avec les données les plus critiques, il faut régler ce paramètre à zéro ou FALSE puisqu'il ralentit le moteur à la vitesse de la synchronisation du fichier sur le disque. Des valeurs sous la barre des 10 millisecondes peuvent être définies en ajoutant MILLIS à la commande, mais en pratique un délai de 100 millisecondes apporte une fiabilité de 99,99999% avec en moyenne un crash système tous les 6 jours.
SET LOG_SIZE <valeur numérique>
Le moteur tient un journal de tous les changements effectués à la base de données au fur et à mesure. Ce journal est synchronisé au disque sur la base de la propriété WRITE_DELAY ci-dessus. Le journal n'est jamais réutilisé à moins qu'il n'y ait une fermeture inopinée, par ex. le processus de base de données s'est terminé sans SHUTDOWN, ou bien il a été arrêté par l'utilisation de SHUTDOWN IMMEDIATELY. La taille maximum du fichier .log est 200 Mo par défaut. Quand la taille maximum est atteinte, une opération CHECKPOINT est réalisée. Cette opération sauvegardera les autres fichiers de la base de données en un état permanent et supprimera l'ancien fichier .log. Une valeur de zéro signifie qu'il n'y a pas de limites pour la taille du fichier .log.
SET CHECKPOINT DEFRAG <valeur numérique>
Quand les lignes de tables en cache sont mises à jour ou supprimées, les "trous" sont la plupart du temps réutilisés. Toutefois, avec le temps, des espaces inutilisés sont laissés dans le fichier .data, spécialement quand de grandes tables sont supprimées ou que leur structure est modifiée. Une opération CHECKPOINT ne récupère normalement pas les espaces vides, alors que CHECKPOINT DEFRAG le fait toujours. Cette propriété détermine quand se produit un CHECKPOINT normal, soit parce qu'initié par un administrateur soit parce que la taille du fichier .log dépasse sa limite. La valeur numérique est le nombre de méga-octets d'espaces vides enregistrés dans le fichier .data qui déclenchera l'opération de défragmentation. Les valeurs basses engendrent des DEFRAG plus fréquents.

Une valeur de zéro signifie qu'aucun DEFRAG automatique ne sera accompli. La valeur par défaut est 200 méga-octets d'espace perdu.
SET REFERENTIAL INTEGRITY {TRUE | FALSE}
Cette propriété est TRUE par défaut. Si un paquet (volume) (bulk) de données doit être chargé dans la base de données, cette propriété peut être définie FALSE durant le chargement du paquet de données. Ce qui permet de charger les données pour les tables reliés dans n'importe quel ordre. La propriété doit être redéfinie à TRUE après le chargement du paquet de données. Si les données chargées ne sont pas garanties conformes aux contraintes d'intégrité référentielles, des requêtes SQL doivent être exécutées après le chargement pour identifier et modifier chaque ligne non conforme.