HSQLDB

1 ¿Qué es HSQLDB?

HSQLDB (**H**yperthreaded **S**tructured **Q**uery **L**anguage **D**atabase) es un sistema gestor de bases de datos libre escrito en Java. La suite ofimática OpenOffice.org lo incluye desde su versión 2.0 para dar soporte a la aplicación Base.

HSQLDB está basado en HypersonicSQL, un proyecto de Thomas Mueller abandonado en la actualidad.

Características de **HSQLDB**:

- Escrito por completo en Java
- Completo sistema gestor de bases de datos relacional
- Tiempo de arranque mínimo y gran velocidad en las operaciones: SELECT, INSERT, DELETE y UPDATE
- Sintaxis SQL estándar
- Integridad referencial (claves foráneas)
- Procedimientos almacenados en Java
- Triggers
- Tablas en disco de hasta 8GB

2 Primeros pasos en HSQLDB

Para instalar HSQLDB (linux y windows) vamos a seguir los siguientes pasos:

Debes descargarte la última versión de HSQL Database Engine que es la continuación de Hypersonic.

Suponiendo que estamos trabajando en Linux, después de descomprimir el fichero zip debes situarte en "hsqldb/bin".

Una vez dentro lo primero que deberías hacer es lanzar el Servidor con la siguiente orden:

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

Depués, en otro terminal podrías lanzar la interfaz gráfica:

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

En Type seleccionas **HSQL Database Engine Standalone.**

En el campo URL introduce lo siguiente: **jdbc:hsqldb:file:/tmp/mydb**

Finalmente si pulsas OK accederas a la Base de Datos.

Ahora Crea la tabla Alumno:

```
CREATE TABLE Alumno (Nombre VARCHAR, Apellidos VARCHAR)
```

Inserta un valor en la tabla, p. e.

```
INSERT INTO Alumno VALUES ('Pepe', 'Pérez')
```

Recuerda que para parar el Servidor y conservar los cambios debes ejecutar el comando sql "**SHUTDOWN**"

3 Consultas sobre una base de datos HSQLDB

Como en cualquier otro DBMS (Sistema de gestión de bases de datos), además de poder llamarlo desde nuestra aplicación, es interesante poder lanzar consultas sobre la base de datos de manera autónoma, ya sea para probar una query antes de incluirla en un programa, o para comprobar que los datos insertados por la aplicación son correctos.

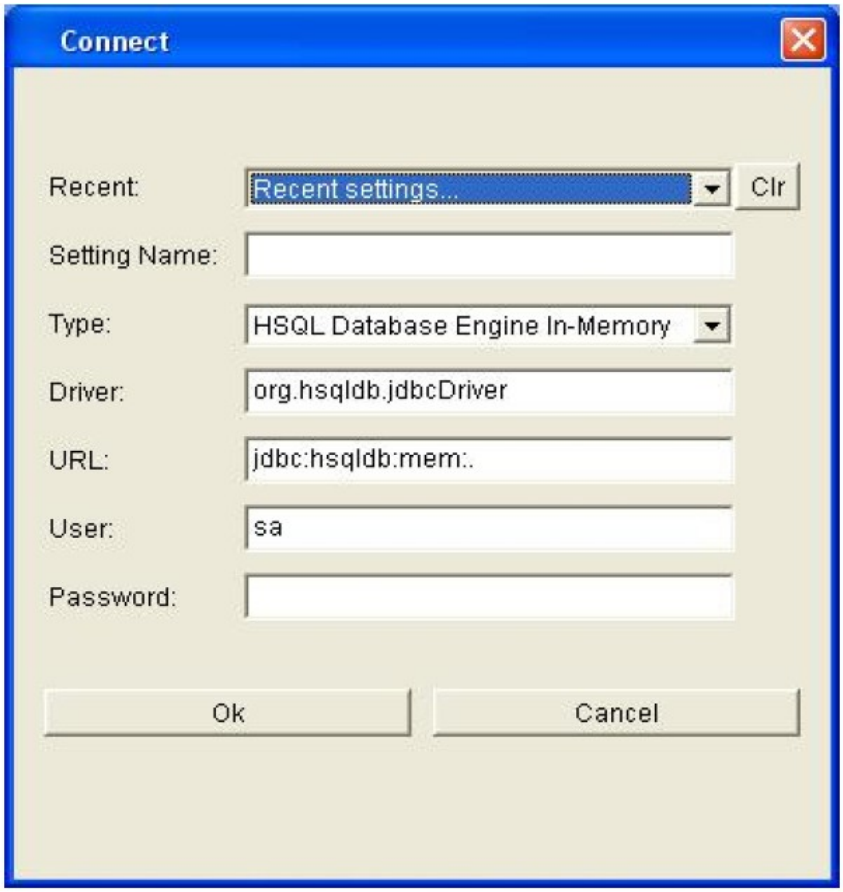
Ejemplos de productos que proporcionan esta funcionalidad son, entre otros Mysql Query Browser o SQLyog para MySQL, o TOAD para Oracle.

HSQLDB también tiene esta caraterística, usando una utilidad incluida en la distribución estándar.

Cuando lo descomprimimos el fichero .zip de HSQLDB, encontramos una estructura de carpetas, con el código, la documentación, los JAR compilados... y el directorio "bin". Dentro de esta carpeta, encontramos un fichero .bat y un .sh (para ejecutar desde Windows y Linux).

Suponiendo que trabajamos en Windows, nos vamos a fijar en el .bat. Si abrimos este fichero en un editor de texto, veremos que lo que hace es ejecutar una clase Java, cuyo nombre recibe como primer parámetro (admite hasta 9 parámetros, cuyo significado se puede comprobar en la documentación), y que se encuentra en el paquete "org.hsqldb.util".

Abrimos una consola y navegamos hasta el directorio "bin" de nuestra distribución de HSQLDB, y ejecutamos desde línea de comandos el script "runUtil.bat". Como parámetro le pasamos el nombre de la clase que nos interesa, en concreto "DatabaseManager". Como podéis ver en la siguiente figura, podemos configurar nuestra conexión con una base de datos cargada en memoria (ruta a la distribucion>runUtil.bat DatabaseManager):



4 Conexión con base de datos HSQLDBen memoria

Con esta opción la base de datos está en memoria, por lo que desaparece cuando muere nuestra aplicación. Se puede usar como almacén interno de datos temporal o para test unitarios. Por ejemplo, un test de *Junit* con el que queremos probar una clase que accede a base de datos. El test de *JUnit*puede crear una base de datos en memoria, rellenarla con unos datos conocidos y luego llamar a la clase bajo test pasándole la *Connection*. Una vez que la clase bajo test termine su trabajo, el test de *Junit* puede revisar la base de datos para ver si se han efectuado los cambios esperados o interrogar a la clase bajo test para ver si ha leido lo que tenía que leer.

Para la conexión sería necesaria configurar una *url* de la siguiente manera*:*

```
Class.forName("org.hsqldb.jdbcDriver");
try {
   Connection connection =
     DriverManager.getConnection("jdbc:hsqldb:mem:memoriadb");
   ...
} catch (SQLException e) {
   e.printStackTrace();
}
```

5 Conexión con una base de datos HSQLDB en fichero

Para esta opción no es necesario arrancar un servidor de *HSQLDB*. Nuestra propia aplicación símplemente indica el nombre del fichero y establece la conexión. Sobre el fichero sólo puede haber una aplicación trabajando con una única conexión, por lo que esta opción es buena para una base de datos interna a nuestra aplicación, a la que en principio nadie más debería querer acceder dsde java, la conexión se establece igual, pero la cadena de conexión cambia:

```
Class.forName("org.hsqldb.jdbcDriver");
try {
   Connection connection =
     DriverManager.getConnection("jdbc:hsqldb:file:ficherodb");
   ...
} catch (SQLException e) {
   e.printStackTrace();
}
```

En la *url* de conexión, *ficherodb* es el nombre incicial del conjunto de ficheros que usará *HSQLDB*. Podemos poner también un path si lo deseamos.

Es importante cerrar o apagar la base de datos con la *Sql* "*shutdown*" para obligar de alguna manera a que el fichero se escriba realmente en disco:

```
Statement st = connection.createStatement();
st.execute("shutdown");
connection.close();
```