Manejo de conectores (1ª Parte)

1. Introducción

Los conectores son elementos que sirven para acceder desde un programa escrito en el lenguaje de programación que sea a datos almacenados en un origen de datos que puede ser de diferente naturaleza. Entendemos por acceso a datos el proceso de recuperación y/o manipulación de datos extraídos de un origen de datos local o remoto que puede ser una base de datos relacional o no, o archivos de muchas fuentes distintas como archivos XML, ficheros de texto, hojas de cálculo, un RSS, etc...

En esta unidad nos centraremos en los orígenes de datos basados en bases de datos relacionales pero veremos que los conceptos aprendidos en esta unidad nos servirán para, en posteriores temas, acceder a otros tipos de bases de datos y orígenes de datos.

Partimos de la premisa que el curso pasado manejamos bases de datos relacionales alojadas en servidor, practicamos en concreto con SQL Server, y el acceso lo realizamos siempre accediendo al servidor directamente a través de un entorno gráfico asociado al mismo (Microsoft Sql Server Management System).

Pero esta no es la única forma de acceder a los datos almacenados en una base de datos, ni todas las bases de datos tienen que estar almacenadas en un servidor de BD.

2. Diferentes formas de acceder a una base de datos

Para acceder a la información almacenada en una base de datos disponemos, la mayoría de las veces, de varias alternativas:

- En modo consola
- Desde el entorno gráfico del SGBD (Sistema Gestor de Base de Datos)
- Desde cualquier programa escrito en el lenguaje que sea.

2.1. En modo Consola.

Introduciendo comandos desde la línea de comandos del propio sistema operativo.

En este caso tenemos que:

- Arrancar el motor de la base de datos.
- Abrir la base de datos de la que queremos sacar información.
- Ejecutar las instrucciones SQL concretas para obtener lo deseado.
- Abandonar el motor de la base de datos.

Practicaremos este modo en el punto de las bases de datos embebidas.

2.2. Desde el entorno gráfico

Casi todos los sistemas gestores de bases de datos incluyen algún entorno gráfico desde el cual se pueda acceder de forma directa y más cómoda a la base de datos. En estos casos lo primero que tenemos que hacer es establecer la conexión con la base de datos a utilizar, una vez la conexión establecida solo nos queda utilizar las opciones de menú y herramientas gráficas del entorno.

Si has utilizado Access has utilizado su entorno gráfico, en este caso la conexión la estableces cuando abres una determinada base de datos o cuando se abre la que tiene Access por defecto (la última abierta).

El curso pasado utilizamos el entorno gráfico Microsoft SQL Server Management Studio de SQL Server, en este caso al iniciar MSSMS te pide establecer conexión con el servidor de base de datos y una vez en el servidor puedes conectarte a la base de datos que quieras dentro del servidor.

2.3. Desde cualquier programa

Esta es la tercera vía de acceso a los datos, la más utilizada y es la que practicaremos este curso. Cualquier aplicación maneja datos por lo que los programas que la conforman deben disponer de instrucciones para poder realizar ese acceso.

Esos datos se suelen almacenar la mayoría de las veces en bases de datos. Estas bases de datos pueden formar parte de la aplicación, en este caso hablamos de bases de datos embebidas, o estar alojados fuera de la aplicación en un servidor, en este caso hablaremos de arquitectura cliente/servidor. En este último caso pueden estar en el mismo ordenador, por lo que será una base de datos local, o estar en otro ordenador, en un servidor al cual se accederá mediante red/internet y hablaremos de acceso remoto. En los tres casos el tratamiento de la información será el mismo, lo único que cambiará será la fase inicial, la de conexión con la base de datos.

3. Bases de datos embebidas

Una base de datos embebida es una base de datos que carece de servidor, cuyo motor está incrustado en la propia aplicación, y suele estar almacenada en ficheros locales. Permite trabajar con una base de datos instalada directamente en el cliente junto con la aplicación que la utiliza. En este caso el motor se inicia cuando se inicia la aplicación y termina cuando se cierra la aplicación. Suelen ser bases de datos más ligeras, de respuesta rápida (suelen cargar gran parte de la información en memoria) pero con acceso monousuario o multiusuario muy limitado.

Muchas de estas bases de datos provienen del movimiento OpenSource aunque también hay algunas de origen propietario y la mayoría de los sistemas grandes tienen una versión embebida.

Por citar algunas tenemos:

- SQLite
- Derby
- HSQLDB
- ….

Para trabajar con ellas, el proceso será similar en cada una de ellas:

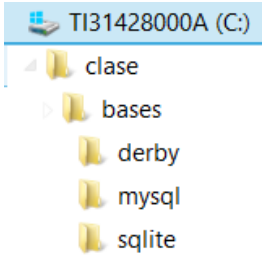
Descargar el programa (el motor de la base de datos).

Suele contener un fichero ejecutable que permite crear y acceder a la base de datos desde la línea de comandos.

Las instrucciones SQL se escriben y se ejecutan automáticamente desde la línea de comando del sistema operativo, y el punto y coma; es interpretado como fin de instrucción y orden de ejecución.

Algunas incluyen herramientas gráficas.

Antes de empezar a estudiar los conectores, veamos algunas de ellas, las instalaremos, lo que nos permitirá utilizarlas en las prácticas posteriores, y de paso practicaremos con el acceso desde consola a una base de datos. Para practicar las explicaciones crea en tu disco la siguiente estructura de carpetas:



3.1. Ejemplos de Bases de datos embebidas.

3.1.1. SQLite

SQLite es un sistema gestor de base de datos relacional multiplataforma escrito en C que proporciona un motor muy ligero pero a la vez potente. Las bases de datos se guardan en forma de ficheros por lo que es fácil trasladar la base de datos con la aplicación que la usa. Cuenta con una utilidad que nos permitirá ejecutar comandos SQL contra la bd en modo consola. SQLite es un proyecto de dominio público.

Su biblioteca implementa la mayor parte del estándar SQL-92, incluyendo transacciones de bases de datos atómicas, consistencia de la bd, aislamiento y durabilidad, triggers y la mayor parte de las consultas complejas.

Los programas que utilizan la funcionalidad de SQLite lo hacen a través de llamadas simples a subrutinas y funciones. SQLite se puede utilizar desde programas en C, C++, Php, Visual Basic, Perl, Delphi, Java, etc..

Su instalación es sencilla. Desde la página http://www.sqlite.org/download.html se puede descargar.

El archivo a descargar dependerá del sistema operativo sobre el que lo vayas a instalar, busca siempre un archivo que sea command-line shell, por ejemplo

sqlite-shell-win32-x86-3081101.zip.

Al descomprimirlo se obtiene un único archivo ejecutable (sqlite3.exe).

Al ejecutar sqlite3 desde la línea de comandos escribimos el nombre del fichero que contendrá la base de datos, si el fichero no existe lo creará, si existe cargará la base de datos.

El siguiente ejemplo crea la base de datos ejemplo.db en la carpeta c:/clase/bases/sqlite

```
c:\>Sqlite3 c:/clase/bases/sqlite/ejemplo.db
SQLite version 3.8.11.1 2015-07-29 20:00:57
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> BEGIN TRANSACTION;
sqlite> CREATE TABLE departamentos (
...>dept_noTINYINT(2) NOT NULL PRIMARY KEY,
...>dnombreVARCHAR(20),
  ...>locVARCHAR(20));
sqlite>COMMIT;
sqlite> INSERT INTO departamentos VALUES (10, 'INFORMATICA', 'DESPA6');
sqlite> SELECT * FROM DEPARTAMENTOS;
10|INFORMATICA|DESPA6
sqlite> INSERT INTO departamentos VALUES (20, 'COMERCIO', 'DESPA7');
sqlite> INSERT INTO departamentos VALUES (30, 'ADMINISTRATIVO', 'DESPA8');
sqlite> INSERT INTO departamentos VALUES (40, 'FOL', 'DESPA5');
sqlite> SELECT * FROM DEPARTAMENTOS;
10|INFORMATICA|DESPA6
20|COMERCIO|DESPA7
30|ADMINISTRATIVO|DESPA8
40|FOL|DESPA5
 sqlite>.quit
```

También podemos abrir la base de datos con el comando .open:

```
sqlite> .open c:/clase/bases/sqlite/ejemplo.db      <-- Ojo con las barras /
```

3.1.2. Apache Derby

Es otra base de datos relacional de código abierto, implementado en su totalidad en Java que forma parte del Apache DB subproject y está disponible bajo la licencia Apache versión 2.0.

Podemos citar algunas de sus ventajas: su tamaño reducido, está basada en Java y soporta los estándares SQL, ofrece un controlador integrado JDBC que permite incrustar Derby en cualquier solución basada en Java, soporta cliente-servidor utilizando Derby Network Server, es fácil de instalar, implementar y utilizar.

Para su instalación nos bajamos la última versión desde la web: http://db.apache.org/derby/derby_downloads.html.

En Windows descargamos deb-derby-10.11.1.1-bin.zip (o la última actualización) y lo descomprimimos en la carpeta que queramos (para el resto de la explicación llamaremos *rutacompleta* la ruta completa de acceso a dicha carpeta).

Dentro de la carpeta descomprimida tenemos las siguientes carpetas:

En la carpeta *demo* tenemos algunos programas de demostración.

En la carpeta *bin* tenemos los scripts para ejecutar algunas utilidades y configurar el programa.

En la carpeta *doc* tenemos la documentación api generada con el programa de comentarios de Java.

En la carpeta *doc* tenemos la documentación del Derby.

En la carpeta *lib* tenemos los archivos jar del programa.

En la carpeta *test* tenemos los archivos de testeo del programa.

En el subdirectorio *frameworks* tenemos los antiguos scripts para ejecutar algunas utilidades y configurar el programa. Estos son incluidos para tener una compatibilidad con versiones anteriores.

A partir de ahora, para poder utilizar Derby en nuestros programas Java, solo será necesario tener accesible la librería derby.jar en el CLASSPATH de nuestro programa.

Ejemplo para configurar la variable CLASSPATH en Windows, para la sesión abierta en la ventana Símbolo del sistema:

```
c:\> SET DERBY_INSTALL= rutacompleta\db-derby-10.11.1.1-bin

c\:> SET CLASSPATH = DERBY_INSTALL\lib\derby.jar;
DERBY_INSTALL\lib\derbytools.jar;%CLASSPATH%;
```

Si queremos visualizar la variable CLASSPATH:

```
c:\> echo %CLASSPATH%
```

Apache Derby trae una serie de ficheros .bat que nos permiten ejecutar desde consola órdenes para crear nuestras bases de datos y ejecutar sentencias DDL y DML, practicaremos con ij.bat que se encuentra en la carpeta *rutacompleta*\db-derby-10.11.1.1-bin\bin.

Desde la línea de comandos (Símbolo del sistema/ventana DOS) posiciónate en *rutacompleta*\db-derby-10.11.1.1-bin\bin y ejecútalo:

```
rutacompleta\db-derby-10.11.1.1-bin\bin>ij
```

A partir de este momento puedes ejecutar los comandos que incluye ij para trabajar con una base de datos Derby.

El primer comando que usaremos será para conectar con la base de datos, esta vez como no existe la tiene que crear previamente (create=true).

```
ij>connect 'jdbc:derby:c:\clase\bases\derby\ejemplo; create=true';
```

Ahora creamos la tabla Departamentos:

```
ij> CREATE TABLE departamentos (
  ...>dept_no INT NOT NULL PRIMARY KEY,
...>dnombre VARCHAR(20),
  ...>loc VARCHAR(20));
```

A continuación insertamos los departamentos:

```
ij> INSERT INTO departamentos VALUES (10, 'INFORMATICA', 'DESPA6');
ij> INSERT INTO departamentos VALUES (20, 'COMERCIO', 'DESPA7');
ij> INSERT INTO departamentos VALUES (30, 'ADMINISTRATIVO', 'DESPA8');
ij> INSERT INTO departamentos VALUES (40, 'FOL', 'DESPA5');
```

Comprobamos que se han insertado las filas:

```
ij> SELECT * FROM DEPARTAMENTOS;
```

También se pueden incluir todas las instrucciones en un archivo de texto (por ejemplo insertadptos.sql) y luego ejecutar el archivo:

```
ij> run 'insertadptos.sql';
```

Para cerrar:

```
ij>exit;
```

La siguiente vez, como la base de datos estará creada la conexión se realizará así:

```
ij>connect 'jdbc:derby:c\clase\bases\derby\ejemplo';
```

3.2. Otras bases de datos embebidas

Aquí tienes una lista de más bases de datos diseñadas para ser utilizadas como bases de datos embebidas:

HSQLDB
H2
FireBird
SQL Server Mobile
Oracle embedded

y otras no relacionales como...

DB4o
NeoDatis
Oracle Berkeley DB
MongoDB

Puedes encontrar infinidad de información sobre ellas, y algunas las utilizaremos en temas posteriores.