UT10. BASES DE DATOS

Módulo: PROGRAMACIÓN

Curso 2022/2023. 1° DAM

Ruth Lospitao Ruiz

INTRODUCCIÓN

- Como se ha visto en la Unidad de Trabajo anterior, Java proporciona las herramientas suficientes para realizar una aplicación en la cual se utilice archivos para almacenar información que genera una aplicación.
- Sin embargo, la utilización de archivos conlleva diversos problemas como la redundancia, la dificultad de mantenimiento, la rigidez en las búsquedas, la seguridad, etc.
- Para paliar estos problemas surgieron las bases de datos, y la herramienta que Java nos proporciona para poder manipular la información que hay contenida en las BD es el API JDBC (Java DataBase Connectivity)
- Este API de Java nos permite ejecutar sentencias SQL

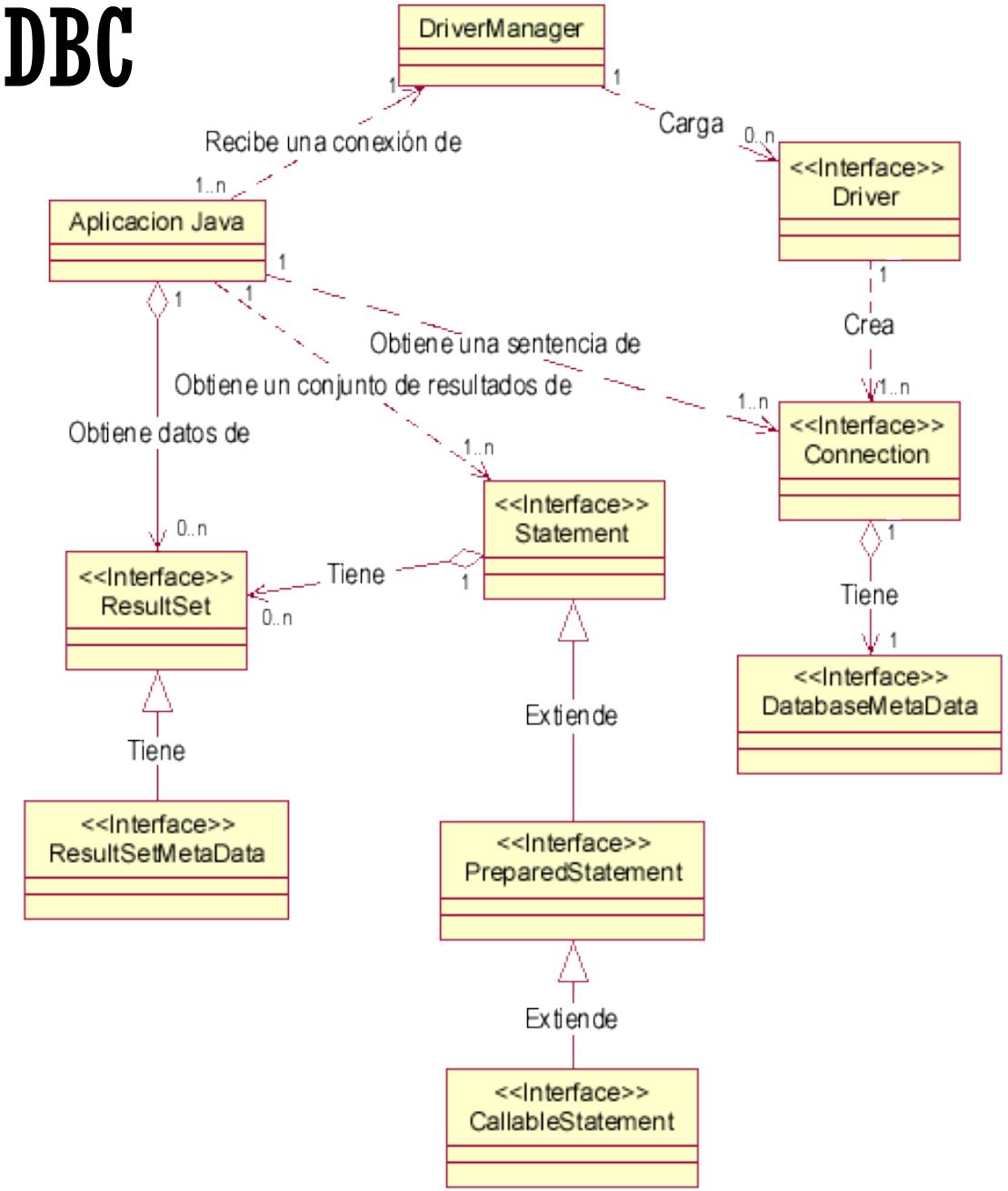
INTRODUCCIÓN

- Una aplicación podrá hacer uso de un BD si establece una conexión con la misma.
- Para ello, deberá usar un driver, que será el encargado de convertir el lenguaje de alto nivel SQL a sentencias entendibles por el sistema gestor de bases de datos.
- Una vez establecida la conexión con la BD podrá enviar sentencias SQL a esa BD y procesar los resultados obtenidos.

JDBC

- Es una herramienta muy potente para desarrollar aplicaciones que interactúan con la BD, en las que tenemos independencia del tipo de servidor de la BD que usemos.
- Para conseguir esta independencia del servidor, se ha construido una jerarquía de clases que hacen posibles las tareas necesarias para el trabajo con BD tales como: la conexión la BD, intercambio de información entre la aplicación y la BD, etc.

MODELO DE CLASES DEL JDBC



MODELO DE CLASES DEL JDBC

Clase / Interface	Descripción
Driver	Permite conectarse a una BD: cada gestor de BD requiere de un driver distinto
DriverManager	Permite gestionar todos los drivers instalados en el sistema
DriverPropertyInfo	Proporciona diversa información acerca de un driver
Connection	Representa una conexión con una BD. Una aplicación puede tener más de una conexión a más de una BD
DatabaseMetadata	Proporciona información acerca de una BD (como las tablas que la componen)
Statement	Permite ejecutar sentencias SQL sin parámetros
PreparedStatement	Permite ejecutar sentencias SQL con parámetros de entrada
CallableStatement	Permite ejecutar sentencias SQL con parámetros de entrada y salida, típicamente procedimientos almacenados
ResultSet	Contiene los registros obtenidos al ejecutar un SELECT
RsultSetMetadata	Permite obtener información sobre un ResultSet (nombre columnas, número…)

PASOS

PASO 1: CONEXIÓN A LA BD

PASO 2: DIÁLOGO CON LA BD

PASO 3: CIERRE CONEXÓN

CONEXIÓN A LA BASE DE DATOS

- A la hora de conectarnos a una BD, debemos tener en cuenta que cada una tiene su propio lenguaje, es decir, necesitamos un traductor que convierta órdenes que le transmitimos en comandos entendibles a esa BD concreta.
- A este tipo de conectaros se les llama controladores.
- Existen gran cantidad de controladores JDBC que nos permiten conectarnos a bases de datos de distintos fabricantes
- Estos controladores se distribuyen como clases de Java. Por lo tanto, lo único que tendremos que hacer es cargarlas en nuestra aplicación con una sentencia similar a:

```
Class.forName(“controlador”);
```

EJEMPLOS DE DRIVERS

- Ejemplo de Driver para Oracle:

```
String driver = new String (“oracle.jdbc.driver.OracleDriver”);
Class.forName(driver);
```

- Ejemplo de Driver para MySQL:

```
String driver = new String (“com.mysql.jdbc.Driver”);
Class.forName(driver);
```

- Ejemplo de Driver para SQL Server:

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
```

- Ejemplo de Driver para PostgresSQL:

```
Class.forName("org.postgresql.Driver");
```

CONEXIÓN A LA BASE DE DATOS

- Una vez que hemos indicado y cargado el controlador es necesario abrir la conexión con la BD, indicando a que ordenador van dirigidas las ordenemos que lancemos.
- Para realizar esto es necesario crear un objeto tipo **`Connection`** utilizando le método **`getConnection()`** de la clase **`DriverManager`** y que tiene como argumento un URL JDBC.
- La sintaxis de la URL JDBC tiene la siguiente forma:

```
jdbc::subprotocolo://servidor:puerto/basedatos
```

- Ejemplo:

```
Connection conexión =
DriverManager.getConnection(“jdbc:mysql://172.29.2.70:1521/nominas”);
```

DIÁLOGO CON LA BD

- Una vez que se tiene abierta la conexión con la BD mediante el objeto de tipo **`Connection`**, se pueden hacer solicitudes de información mediante un objeto de la clase **`Statement`**
- Para crear las distintas órdenes utilizaremos el método **`createStatement`**() de la clase **`Connection`**. Este método crea un objeto Statement asociado a la conexión con la BD que estamos utilizando.

```
Statement sentencia = conexion.createStatement();
```

DIÁLOGO CON LA BD

- Con el objeto sentencia que nos devuelve el método **`createStatement`** podemos realizar cualquier tipo de operación (consultas, modificaciones, etc) sobre la BD.
- Para realizarlas se suele utilizar dos métodos de la clase **`Statement`**:
  - **`execute (String sentencia)`** para ejecutar una SQL que no devuelve datos
  - y **`executeQuery(String sentencia)`** para ejecutar una consulta SQL que sí devuelve datos
- Cada una de estas sentencias puede devolver cero o más resultados, JDBC utiliza los objetos tipo **`ResultSet`** para capturar la información que recibe

```
ResultSet resultado = sentencia.executeQuery ("SELECT * FROM TABLA");
```

- De esta forma tendremos el objeto **`resultado`** con el contenido de la consulta, el cual podremos recorrer con los métodos definidos en la clase **`ResultSet`**. Por ejemplo, con su método **`next()`**
- El objeto ResultSet devuelto por el método **`executeQuery()`**, permite recorrer las filas pero no proporciona información sobre la estructura. Para ello se utiliza **`ResultSetMetaData`**. Para obtener un objeto **`ResultSetMetaData`** basta con llamar al método **`getMetadaData()`** del objeto **`ResultSet`**

CIERRE DE LA CONEXIÓN

- Cuando terminamos de operar con la Base de Datos se debe cerrar la conexión establecida.
- Para ello, se utilziará el método `close()` que se encuentra definido en `Connection, ResultSet y Statement.`

```
resultado.close();
sentencia.close();
conexión.close();
```

UT10. BASES DE DATOS

Módulo: PROGRAMACIÓN

Curso 2022/2023. 1° DAM

Ruth Lospitao Ruiz