12.A. Introducción.

Sitio: Formación Profesional a Distancia
Curso: Programación
Libro: 12.A. Introducción.
Imprimido por: Iván Jiménez Utiel
Día: domingo, 17 de mayo de 2020, 16:25

Tabla de contenidos

1. Introducción.

2. El desfase objeto-relacional.

3. JDBC.

4. Conectores o Drivers.

5. Instalación de la base de datos.

1. Introducción.

Hoy en día, la mayoría de aplicaciones informáticas necesitan almacenar y gestionar gran cantidad de datos.

Esos datos, se suelen guardar en bases de datos relacionales, ya que éstas son las más extendidas actualmente.

Las bases de datos relacionales permiten organizar los datos en tablas y esas tablas y datos se relacionan mediante campos clave. Además se trabaja con el lenguaje estándar conocido como SQL, para poder realizar las consultas que deseemos a la base de datos.

Una base de datos relacional se puede definir de una manera simple como aquella que presenta la información en tablas con filas y columnas.

idreg	Temporada	Lugar	equicasero	golescasero	equiforaneo	golesforaneo
1	1955-1956	París	Real Madrid	4	Stade Reims	3
2	1956-1957	Madrid	Real Madrid	2	Fiorentina	0
3	1957-1958	Bruselas	Real Madrid	3	A.C. Milán	2
4	1958-1959	Sttutgart	Real Madrid	2	Stade Reims	0
5	1959-1960	Glasgow	Real Madrid	7	Eintrach de Fra	3
6	1960-1961	Berna	Benfica	3	Barcelona	2
7	1961-1962	Amsterdam	Benfica	5	Real Madrid	3
8	1962-1963	Londres	A.C. Milán	2	Benfica	1
9	1963-1964	Viena	Inter de Milán	3	Real Madrid	1
10	1964-1965	Milán	Inter de Milán	1	Benfica	0
11	1965-1966	Bruselas	Real Madrid	2	Partizán de Bel	1
12	1966-1967	Lisboa	Celtic de Glasgow	2	Inter de Milán	1
13	1967-1968	Londres	Manchester United	4	Benfica	1
14	1968-1969	Madrid	A.C. Milán	4	Ajax de Amster	1
15	1969-1970	Milán	Feyenoord	4	Celtic de Glasg	1
16	1970-1971	Londres	Ajax de Amsterdam	2	Panathinaikos	1
17	1971-1972	Rotterdam	Ajax de Amsterdam	2	Inter de Milán	0
18	1972-1973	Belgrado	Ajax de Amsterdam	1	Juventus	0
19	1973-1974	Bruselas	Bayern de Munich	1	Atl. De Madrid	1
20				4		1
21	1974-1975	París	Bayern de Munich	2	Leeds United	0
22	1975-1976	Glasgow	Bayern de Munich	1	Saint Etienne	0
23	1976-1977	Roma	Liverpool	3	Borussia Moenc	1

Una **tabla es una serie de filas y columnas**, en la que **cada fila es un registro** y cada columna es un campo. Un **campo** representa un dato de los elementos almacenados en la tabla (NSS, nombre, etc.) Cada registro representa un elemento de la tabla (el equipo Real Madrid, el equipo Real Murcia, etc.)

No se permite que pueda aparecer dos o más veces el mismo registro, por lo que uno o más campos de la tabla forman lo que se conoce como clave primaria.

El sistema gestor de bases de datos, en inglés conocido como: **Database Management System (DBMS)**, gestiona el modo en que los datos se almacenan, mantienen y recuperan.

En el caso de una base de datos relacional, el sistema gestor de base de datos se denomina: **Relational Database Management System (RDBMS)**.

Tradicionalmente, la programación de bases de datos ha sido como una Torre de Babel: gran cantidad de productos de bases de datos en el mercado, y cada uno "hablando" en su lenguaje privado con las aplicaciones.

Java, mediante JDBC (***Java Database Connectivity***), permite **simplificar el acceso a base de datos**, proporcionando un lenguaje mediante el cual las aplicaciones pueden comunicarse con motores de bases de datos. Sun desarrolló este API para el acceso a bases de datos, con tres objetivos principales en mente:

- Ser un API con soporte de SQL: poder construir sentencias SQL e insertarlas dentro de llamadas al API de Java,
- Aprovechar la experiencia de los APIs de bases de datos existentes,
- Ser sencillo.

Autoevaluación

JDBC permite acceder a bases de datos relacionales cuando programamos con Java, pudiendo así utilizar SQL.

- Verdadero.
- Falso.

Para saber más

Si necesitas refrescar o simplemente aprender el concepto de clave primaria, en la wikipedia puedes consultarlo.

Clave primaria.

2. El desfase objeto-relacional.

El desfase objeto-relacional, también conocido como impedancia objeto-relacional, consiste en la diferencia de aspectos que existen entre la programación orientada a objetos y la base de datos. Estos aspectos se puede presentar en cuestiones como:

- **Lenguaje de programación**. El programador debe conocer el lenguaje de programación orientada a objetos (POO) y el lenguaje de acceso a datos.

- **Tipos de datos**: en las bases de datos relacionales siempre hay restricciones en cuanto a los tipos de datos que se pueden usar, que suelen ser sencillos, mientras que la programación orientada a objetos utiliza tipos de datos más complejos.

- **Paradigma de programación**. En el proceso de diseño y construcción del software se tiene que hacer una traducción del modelo orientado a objetos de clases al modelo Entidad-Relación (E/R) puesto que el primero maneja objetos y el segundo maneja tablas y tuplas o filas, lo que implica que se tengan que diseñar dos diagramas diferentes para el diseño de la aplicación.

El modelo relacional trata con relaciones y conjuntos debido a su **naturaleza matemática**. Sin embargo, el modelo de Programación Orientada a Objetos trata con objetos y las asociaciones entre ellos. Por esta razón, el problema entre estos dos modelos surge en el momento de querer persistir los objetos de negocio

La escritura (y de manera similar la lectura) mediante JDBC implica: abrir una conexión, crear una sentencia en SQL y copiar todos los valores de las propiedades de un objeto en la sentencia, ejecutarla y así almacenar el objeto. Esto es sencillo para un caso simple, pero trabajoso si el objeto posee muchas propiedades, o bien se necesita almacenar un objeto que a su vez posee una colección de otros elementos. Se necesita crear mucho más código, además del tedioso trabajo de creación de sentencias SQL.

Este problema es lo que denominábamos **impedancia Objeto-Relacional**, o sea, el conjunto de dificultades técnicas que surgen cuando una base de datos relacional se usa en conjunto con un programa escrito en con u lenguajes de Programación Orientada a Objetos.

Podemos poner como ejemplo de desfase objeto-relacional, un Equipo de fútbol, que tenga un atributo que sea una colección de objetos de la clase Jugador. Cada jugador tiene un atributo "teléfono". Al transformar éste caso a relacional se ocuparía más de una tabla para almacenar la información, implicando varias sentencias SQL y bastante código.

Debes conocer

Si no has estudiado nunca bases de datos, ni tienes idea de qué es SQL o el modelo relacional, sería conveniente que te familiarizaras con él. A continuación te indicamos un tutorial bastante ameno sobre SQL y en donde describe brevemente el modelo relacional.

Tutorial SQL.

3. JDBC.

JDBC es un API Java que hace posible ejecutar sentencias SQL.

De JDBC podemos decir que:

- Consta de un **conjunto de clases e interfaces** escritas en Java.

- Proporciona un API estándar para desarrollar aplicaciones de bases de datos con un API Java pura.

Con JDBC, no hay que escribir un programa para acceder a una base de datos Access, otro programa distinto para acceder a una base de datos Oracle, etc., sino que **podemos escribir un único programa** con el API JDBC y el programa se encargará de enviar las sentencias SQL a la base de datos apropiada. Además, y como ya sabemos, una aplicación en Java puede ejecutarse en plataformas distintas.

En el desarrollo de JDBC, y debido a la confusión que hubo por la proliferación de API's propietarios de acceso a datos, Sun buscó los aspectos de éxito de un API de este tipo, ODBC (Open Database Connectivity).

ODBC se desarrolló con la idea de tener un estándar para el acceso a bases de datos en entorno Windows.

Aunque la industria ha aceptado ODBC como medio principal para acceso a bases de datos en Windows, ODBC no se introduce bien en el mundo Java, debido a la complejidad que presenta ODBC, y que entre otras cosas ha impedido su transición fuera del entorno Windows.

El nivel de abstracción al que trabaja JDBC es alto en comparación con ODBC, la intención de Sun fue que supusiera la base de partida para crear librerías de más alto nivel.

JDBC intenta ser tan simple como sea posible, pero proporcionando a los desarrolladores la máxima flexibilidad.

Autoevaluación

JDBC es la versión de ODBC para Linux.

- Verdadero.
- Falso.

4. Conectores o Drivers.

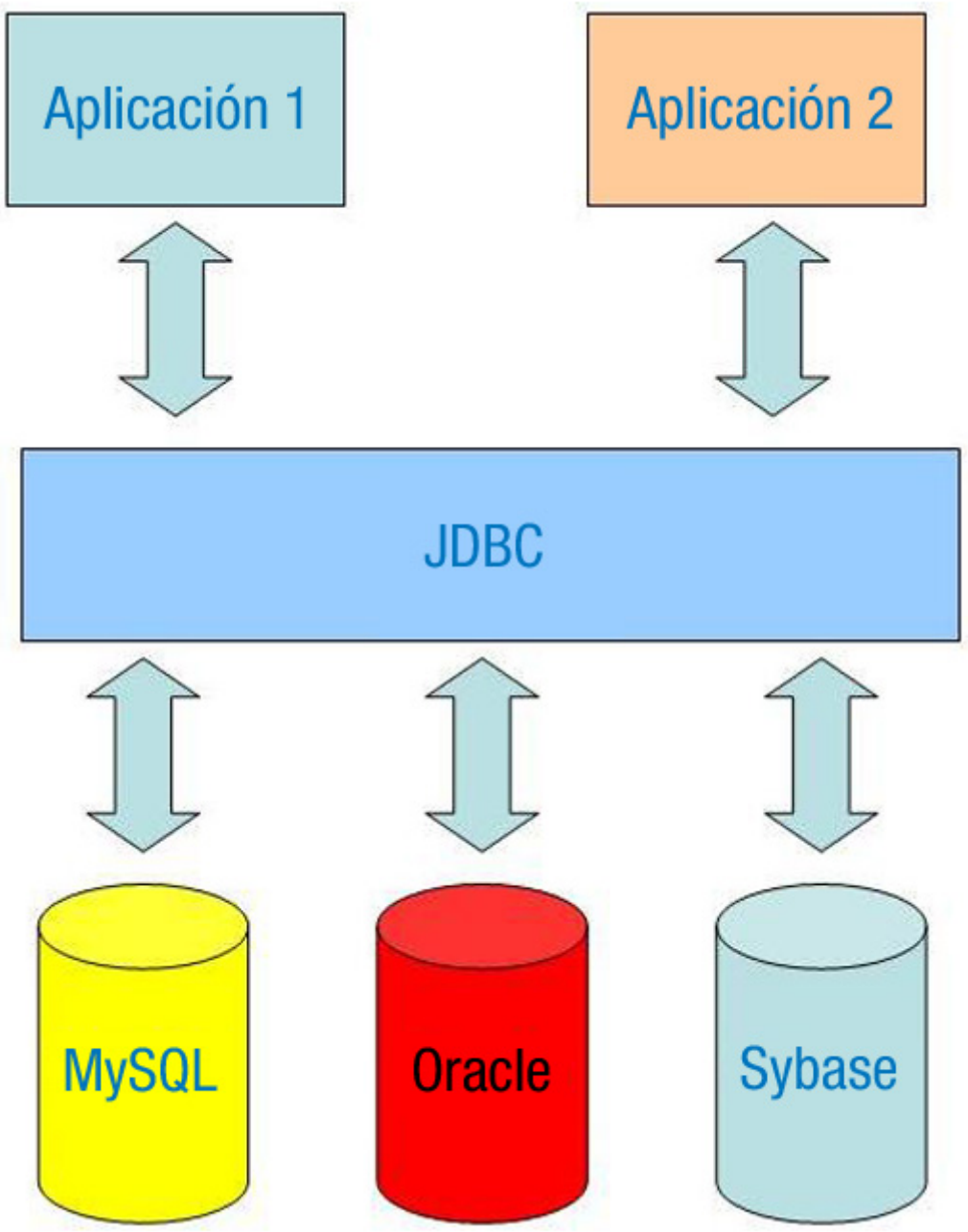
El API JDBC viene distribuido en dos paquetes:

- `java.sql`, dentro de J2SE
- `javax.sql`, extensión dentro de J2EE

Un conector o driver es un conjunto de clases encargadas de implementar las interfaces del API y acceder a la base de datos.

Para poder conectarse a una base de datos y lanzar consultas, una aplicación necesita tener un conector adecuado. Un conector suele ser un fichero .jar que contiene una implementación de todas las interfaces del API JDBC.

Cuando se construye una aplicación de base de datos, **JDBC oculta los detalles específicos de cada base de datos**, de modo que le programador se ocupe sólo de su aplicación.



El conector lo proporciona el fabricante de la base de datos o bien un tercero.

El código de nuestra aplicación no depende del driver, puesto que trabajamos contra los paquetes`java.sql` y `javax.sql`.

JDBC ofrece las clases e interfaces para:

- Establecer una conexión a una base de datos.

- Ejecutar una consulta.

- Procesar los resultados.

Ejemplo:

```
// Establece la conexión
Connection con = DriverManager.getConnection ("jdbc:odbc:miBD",
        "miLogin", "miPassword");

// Ejecuta la consulta
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT nombre, edad FROM Jugadores");

// Procesa los resultados
while (rs.next()) {
      String nombre = rs.getString("nombre");
      int edad = rs.getInt("edad");
}
```

El mismo código copiable:

```
// Establece la conexión
Connection con = DriverManager.getConnection (
"jdbc:odbc:miBD", "miLogin", "miPassword");

// Ejecuta la consulta
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT nombre, edad FROM Jugadores");

// Procesa los resultados
while (rs.next()) {
String nombre = rs.getString("nombre");
int edad = rs.getInt("edad");
}
```

En principio, todos los conectores deben ser compatibles con ANSI SQL-2 Entry Level (ANSI SQL-2 se refiere a los estándares adoptados por el **American National Standards Institute** (ANSI) en 1992. Entry Level se refiere a una lista específica de capacidades de SQL). Los desarrolladores de conectores pueden establecer que sus conectores conocen estos estándares.

Para saber más

Lista sobre los conectores JDBC para acceder a muchas las bases de datos listadas.

Conectores JDBC

5. Instalación de la base de datos.

Lo primero que tenemos que hacer, para poder realizar consultas en una base de datos, es obviamente, instalar la base de datos. Dada la cantidad de productos de este tipo que hay en el mercado, es imposible explicar la instalación de todas. Así que vamos a optar por una, en concreto por MySQL ya que es un sistema gestor gratuito y que funciona en varias plataformas.

Para instalar MySQL en Windows puedes seguir los pasos que te detallamos en la siguiente presentación:

Nota: Si no quieres complicarte mucho con la instalación, configuración y pruebas de funcionamiento, una alternativa muy aconsejable será instalar XAMPP o WAMP, y lanzar el servicio MySQL. Además luego podremos gestionar nuestras bases de datos a través de una interfaz gráfica y amigable, concretamente a través de phpMyAdmin, aplicación que viene incluida con XAMPP o con WAMP.

Si utilizas Linux, la instalación la puedes hacer con los pasos que te indican en el enlace:

Instalar MySQL en Linux.

Autoevaluación

Para programar accesos a MySQL hay que tener el driver JDBC para MySQL en nuestra aplicación.

- Verdadero.
- Falso.

Para saber más

Para probar con otras bases de datos, puedes instalar también Oracle. Aquí puedes ver como se instala.

Instalar Oracle.