ACCESO A DATOS

CAPÍTULO 2

Manejo de conectores

José Eduardo Córcoles
Francisco Montero

Ra-Ma®

ÍNDICE DE CONTENIDOS

OBJETIVOS

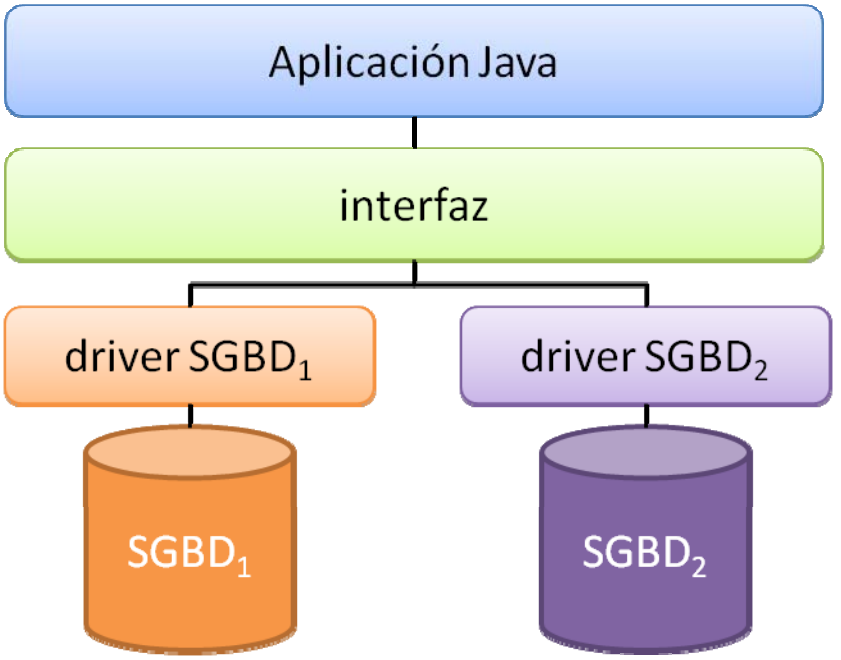
- Valorar las ventajas e inconvenientes de utilizar conectores.
- Utilizar gestores de bases de datos embebidos e indep.
- Utilizar el conector idóneo.
- Establecer la conexión con una base de datos.
- Definir la estructura de la base de datos (BBDD).
- Desarrollar aplicaciones que modifican la BBDD.
- Definir los objetos que almacenan el resultado de las consultas.
- Desarrollar aplicaciones que efectúan consultas.
- Eliminar los objetos una vez finalizada su función.
- Gestionar las transacciones.

2.1 EL DESFASE OBJETO-RELACIONAL.

- El problema del desfase objeto-relacional consiste en la diferencia de aspectos que existen entre la programación orientada a objetos, con la que se desarrollan aplicaciones, y la base de datos, con las que se almacena la información.
- Surgen cuando:
  - Se realizan actividades de programación.
  - Se especifican los tipos de datos.
  - En el proceso de elaboración del software se realiza una traducción del modelo orientado a objetos al modelo Entidad-Relación.

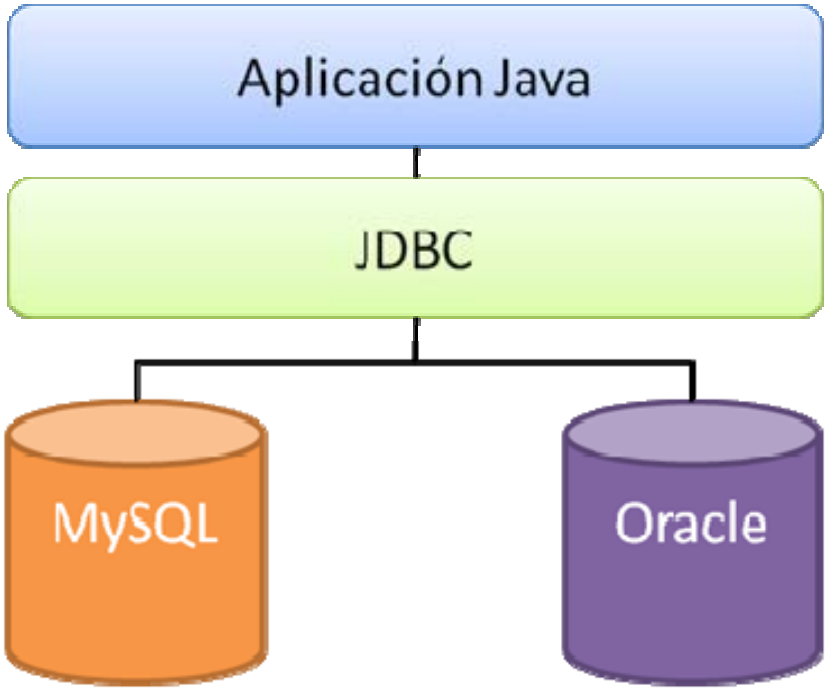
2.2 PROTOCOLOS DE ACCESO A BASES DE DATOS. CONECTORES

- Muchos servidores de bases de datos utilizan protocolos de comunicación específicos que facilitan el acceso a los mismos.
- Estas interfaces de alto nivel ofrecen facilidades para:
  - Establecer una conexión a una base de datos.
  - Ejecutar consultas sobre una base de datos.
  - Procesar los resultados de las consultas realizadas.



2.2 PROTOCOLOS DE ACCESO A BASES DE DATOS. CONECTORES

- Cuando se construye una aplicación de base de datos, el **conector** oculta los detalles específicos de cada base de datos.
- La mayoría de los fabricantes ofrecen conectores para acceder a sus bases de datos.
- Un ejemplo de conector muy extendido es el conector **JDBC**.

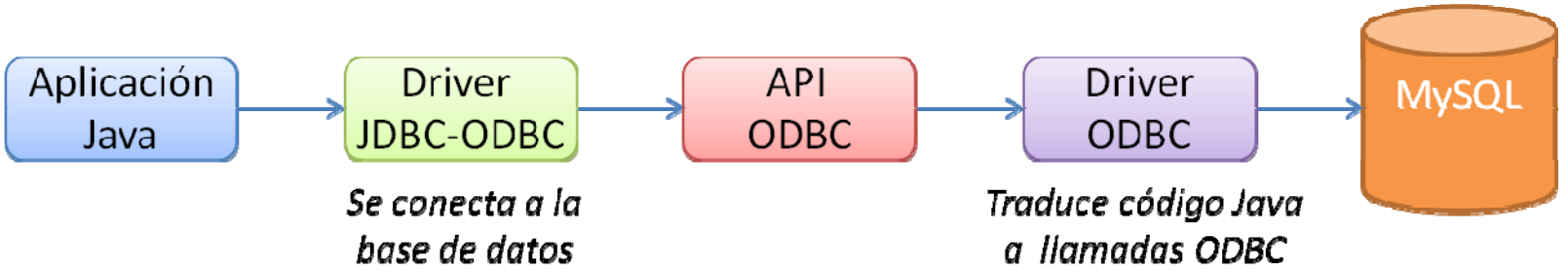


2.2.1 COMPONENTES JDBC

- El conector JDBC incluye cuatro componentes principales:
  - El propio API JDBC, que facilita el acceso desde el lenguaje de programación Java a bases de datos relacionales.
  - El gestor del conector JDBC (driver manager), que conecta una aplicación java con el driver correcto de JDBC.
  - La suite de pruebas JDBC, encargada de comprobar si un conector (driver) cumple con los requisitos JDBC.
  - El driver o puente JDBC-ODBC, que permite que se puedan utilizar los drivers ODBC como si fueran de tipo JDBC.

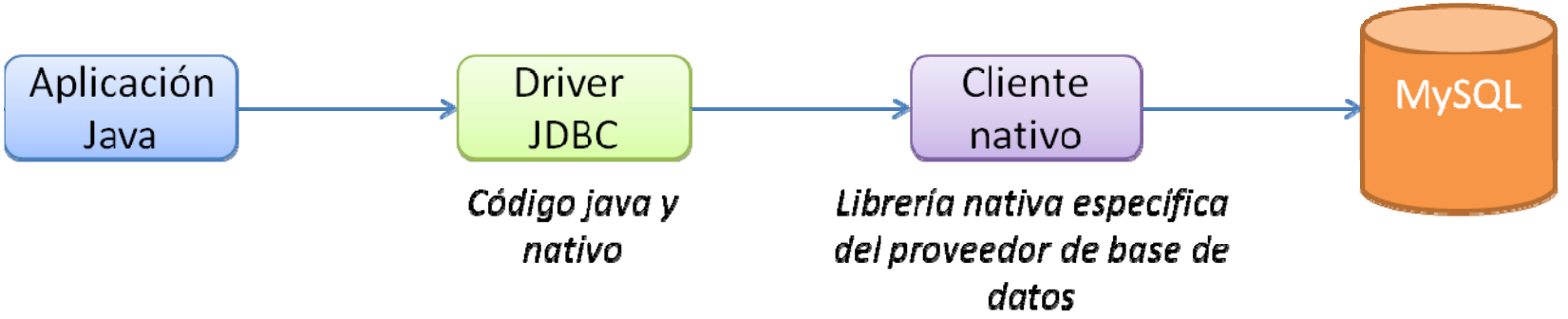
2.2.2 TIPOS DE CONECTORES JDBC

- En función de los componentes anteriores, en un conector JDBC existen cuatro tipos de controladores JDBC.
- **Driver tipo 1**: utilizan un API nativa estándar, donde se traducen las llamadas de JDBC a invocaciones ODBC a través de librerías ODBC del sistema operativo.



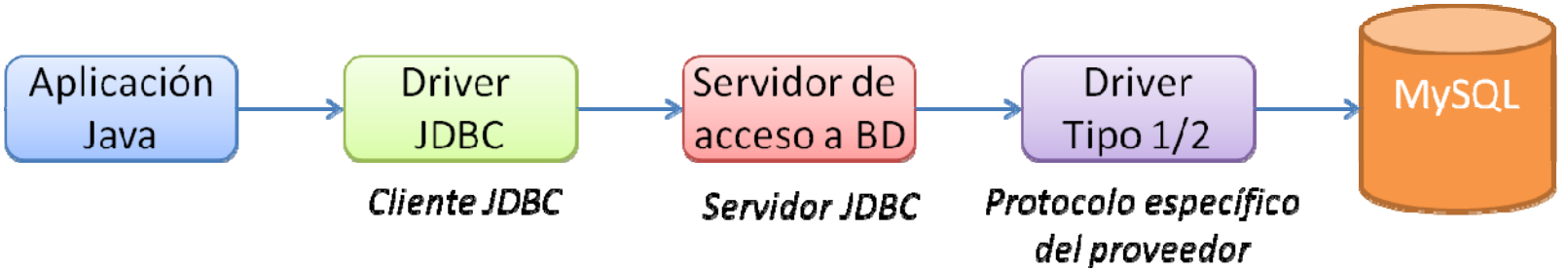
2.2.2 TIPOS DE CONECTORES JDBC

- **Driver tipo 2**: utilizan un API nativa de la base de datos, es decir son drivers escritos parte en Java y parte en código nativo. El driver usa una librería cliente nativa, específica de la base de datos con la que se desea conectar.



2.2.2 TIPOS DE CONECTORES JDBC

- **Driver tipo 3**: utilizan un servidor remoto con una API genérica, es decir son drivers que usan un cliente Java puro que se comunica con un middleware server usando un protocolo independiente de la base de datos (por ejemplo, TCP/IP).

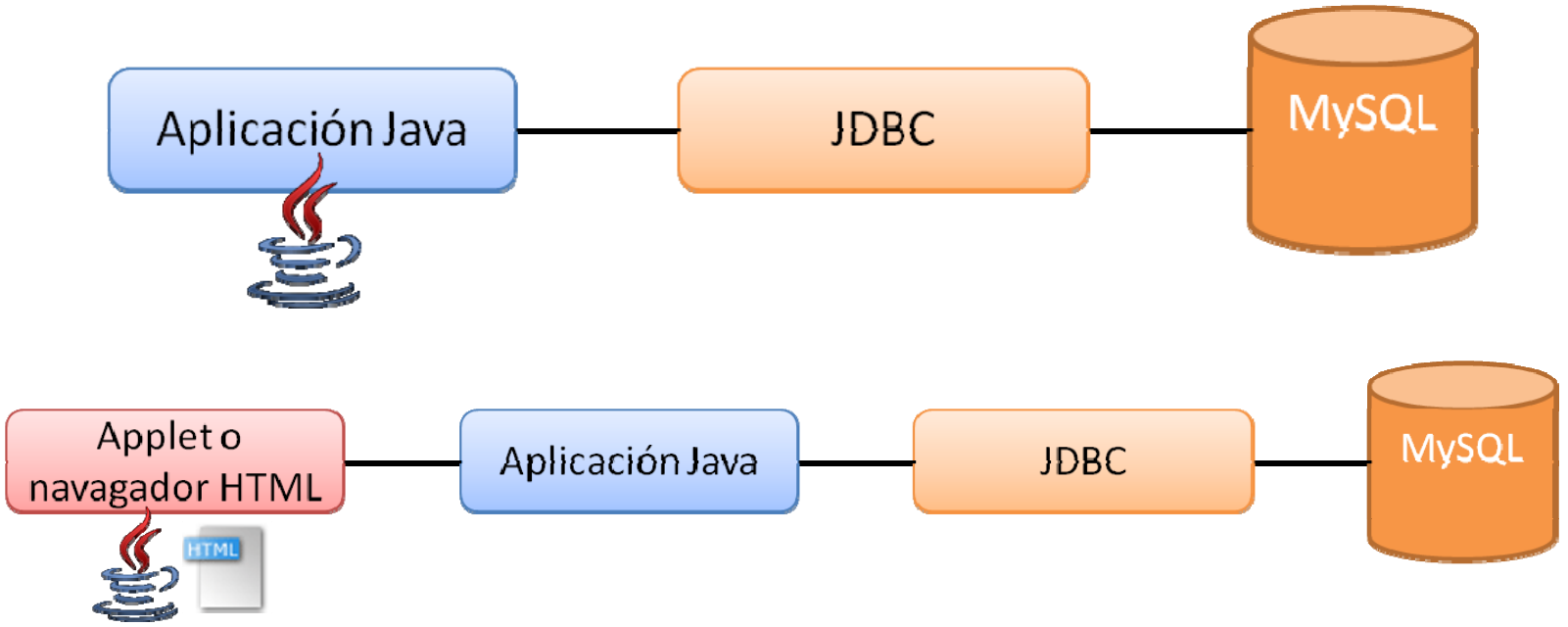


2.2.2 TIPOS DE CONECTORES JDBC

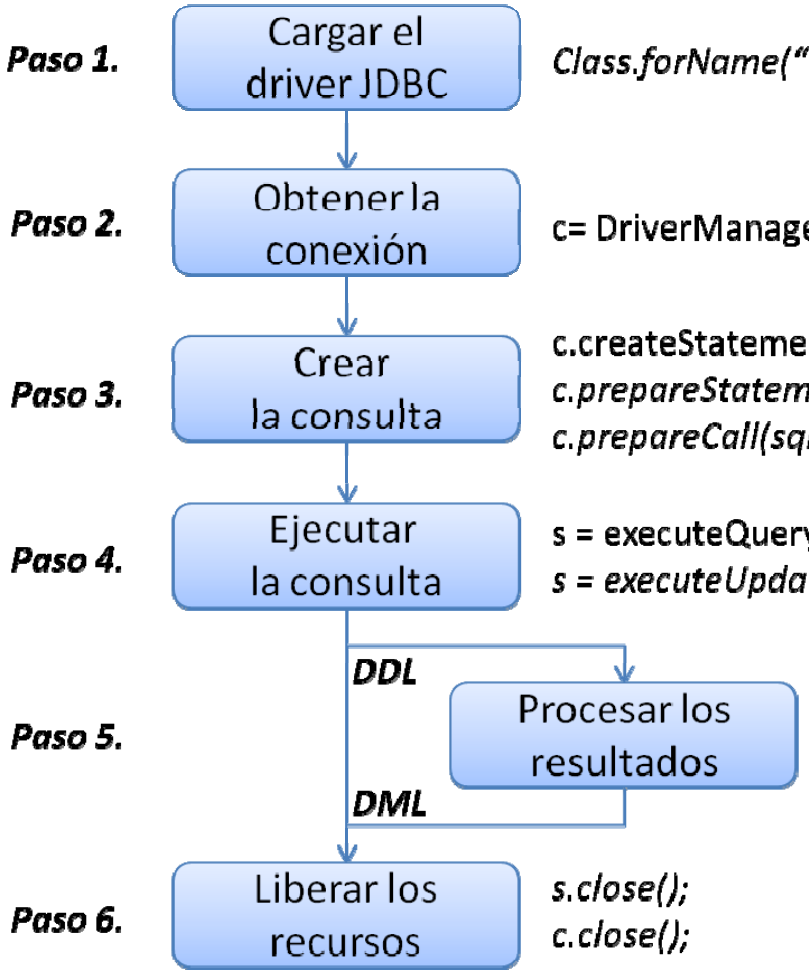
- **Driver tipo 4**: es el método más eficiente de acceso a base de datos. Este tipo de drivers son suministrados por el fabricante de la base de datos y su finalidad es convertir llamadas JDBC en un protocolo de red comprendido por la base de datos.

2.2.3 MODELOS DE ACCESO A BASES DE DATOS

- A la hora de establecer el canal de comunicación entre una aplicación Java y una base de datos se pueden identificar dos modelos distintos de acceso. Estos modelos dependen del número capas que se contemplen.



2.2.4 ACCESO A BASES DE DATOS MEDIANTE UN CONECTOR JDBC



Actividad 2.1

2.2.5 CLASES BÁSICAS DEL API JDBC

- En la interfaz hay distintos tipos de objetos que se deben tener presentes, por ejemplo: *Connection*, *Statement* y *ResultSet*.
  - Los objetos de la clase *Connection* ofrecen un enlace activo a una base de datos a través del cual un programa en Java puede leer y escribir datos, así como explorar la estructura de la base de datos y sus capacidades.
  - La interfaz *DriverManager*, complementaria de la clase *Connection*, con ella se registran los controladores JDBC y se proporcionan las conexiones que permiten manejar las URL específicas de JDBC.
  - La clase *Statement* proporciona los métodos para que las sentencias, utilizando el lenguaje de consulta estructurado (SQL), sean realizadas sobre la base de datos .

Actividad 2.2

2.2.6 CLASES ADICIONALES DEL API JDBC

- Además de las clases básicas anteriores, el API JDBC también ofrece la posibilidad de acceder a los **metadatos** de una base de datos.
- Con ellos se puede obtener información sobre la estructura de la base de datos y, gracias a ello, se pueden desarrollar aplicaciones independientemente del esquema que tenga la base de datos.
- Las principales clases asociadas a metadatos de una base de datos son las siguientes: ***DatabaseMetaData*** y ***ResultSetMetaData***.

Actividad 2.3

2.3 EJECUCIÓN DE SENTENCIAS DE DEFINICIÓN DE DATOS

- Las principales sentencias asociadas con el lenguaje DDL son CREATE, ALTER y DROP. Dichas sentencias permiten:
  - CREATE sirve para crear una base de datos o un objeto.
  - ALTER sirve para modificar la estructura de una base de datos o de un objeto.
  - DROP permite eliminar una base de datos o un objeto.
- Para enviar comandos SQL a la base de datos con JDBC se usa un objeto de la clase **Statement**.

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

- Statement tiene muchos métodos, pero hay dos interesantes: **executeUpdate**() y **executeQuery**().

2.4 EJECUCIÓN DE SENTENCIAS DE MANIPULACIÓN DE DATOS

- Las sentencias de manipulación de datos (DML, Data Manipulation Language) son las utilizadas para insertar, borrar, modificar y consultar los datos que hay en una base de datos. Las sentencias DML son las siguientes:
  - La sentencia **SELECT** sirve para recuperar información de una base de datos y permite la selección de una o más filas y columnas de una o muchas tablas.
  - La sentencia **INSERT** se utilizan para agregar registros a una tabla.
  - La sentencia **UPDATE** permite modificar la información de las tablas.
  - La sentencia **DELETE** permite eliminar una o más filas de una tabla.

Actividad 2.4

2.5 EJECUCIÓN DE CONSULTAS

- La ejecución de consultas sobre bases de datos desde aplicaciones Java descansa en dos tipos de clases disponibles en el API JDBC y en dos métodos. Las clases son: Statement y ResultSet y los métodos son: executeQuery y executeUpdate.

2.5.1 CLASE STATEMENT

```
public void Consulta_Statement(){
    try {
     Statement stmt = conn1.createStatement();
     String query = "SELECT * FROM album WHERE
      titulo like 'B%'";
     ResultSet rs = stmt.executeQuery(query);
     while (rs.next()) {
        System.out.println("ID - " + rs.getInt("id") +
           ", Título " + rs.getString("titulo") +
           ", Autor " + rs.getString("autor") );
     }
     rs.close();
     stmt.close();
    } catch (SQLException ex) {
        System.out.println("ERROR:al hacer un
      Insert");
        ex.printStackTrace();
    }
  }
```

- Como se ha comentado las sentencias Statement son las encargadas de ejecutar las sentencias SQL estáticas con **Connection.createStatement().**
- El método executeQuery() de Statement está diseñado para sentencias que producen como resultado un único resultado (ResultSet) , como es el caso de las sentencias SELECT.

2.5.2 CLASE PREPAREDSTATEMENT

```
public void Consulta_preparedStatement(){
    try {
      String query = "SELECT * FROM album WHERE titulo like ?";
      PreparedStatement pst = conn1.prepareStatement(query);
      pst.setString(1, "B%");
      ResultSet rs = pst.executeQuery();
      while (rs.next()) {
         System.out.println("ID - " + rs.getInt("id") +
            ", Título " + rs.getString("titulo") +
            ", Autor " + rs.getString("autor") );
      }
      rs.close();
      pst.close();
    } catch (SQLException ex) {
         System.out.println("ERROR:al consultar");
         ex.printStackTrace();
    }
  }
```

- Una variante a las sentencia *Statement* es la sentencia *PreparedStatement*.
- Se utiliza para ejecutar las sentencias SQL precompiladas.
- Permite que los parámetros de entrada sean establecidos de forma dinámica, ganando eficiencia.

2.5.3 CLASE CALLABLESTATEMENT

- Otro tipo de sentencias son las asociadas a los objetos de la clase CallableStatement.
- Son sentencias preparedStatement que llaman a un procedimiento almacenado, es decir, métodos incluidos en la propia base de datos.
- No todos los gestores de bases de datos admiten este tipo de procedimientos.

```
CallableStatement cs = conn1.prepareCall("CALL DameAlbumes(?,?)");
// Se proporcionan valores de entrada al procedimiento
cs.setString(1, “Black%”);
cs.setString(2, “Metallica”);
```

Actividad 2.5

2.6 GESTIÓN DE TRANSACCIONES

- Una transacción en un Sistema de Gestión de Bases de Datos (SGBD), es un conjunto de órdenes que se ejecutan como una unidad de trabajo.
- Una transacción se inicia cuando se encuentra una primera sentencia DML y finaliza cuando se ejecuta alguna de las siguientes sentencias:
  - un COMMIT o un ROLLBACK.
  - una sentencia DDL, por ejemplo CREATE.
  - una sentencia DCL, por ejemplo GRANT o REVOKE.

Actividad 2.6

2.7 CONCLUSIONES Y PROPUESTAS PARA AMPLIAR

- En este capítulo se han mostrado JDBC como alternativa para conectar aplicaciones Java con bases de datos.
- El lector interesado en profundizar en la tecnología expuestas en el capítulo puede hacerlo en las líneas siguientes:
  - Optimización de conexiones (pool de conexiones en aplicaciones multihilo) y de acceso a datos relacionales con SQL.
  - Otras alternativas para el acceso con conectores, por ejemplo Java Blend y SQLJ.
- Para profundizar en estas líneas de trabajo el título *SQL y Java - guía para sqlj, jdbc y tecnologias relacionadas* – de Jim Melton y Andrew Eisenberg, de la editorial RA-MA es una buena referencia.

2.8 RESUMEN DEL CAPÍTULO

- En este capítulo se ha abordado el acceso a datos con conectores. En concreto, siendo coherentes con el resto de contenidos de este libro, se ha trabajado con JDBC, la alternativa más extendida para el acceso a datos almacenados en sistemas gestores relacionales desde Java.
- La primera parte se ha centrado en una introducción a JDBC y sus posibilidades.
- La segunda parte muestra con ejemplo de código el uso de las principales interfaces Java que permiten ejecutar operaciones de modificación y consulta con SQL.

ACCESO A DATOS

CAPÍTULO 2

Manejo de conectores

José Eduardo Córcoles
Francisco Montero

Ra-Ma®