Protocols d'accés a bases de dades

Accés a bases de dades

Consorci d'Educació de Barcelona
Generalitat de Catalunya
Ajuntament de Barcelona

Introducció

En tecnologies de base de dades podem trobar-nos amb dues normes de connexió a una base de dades:

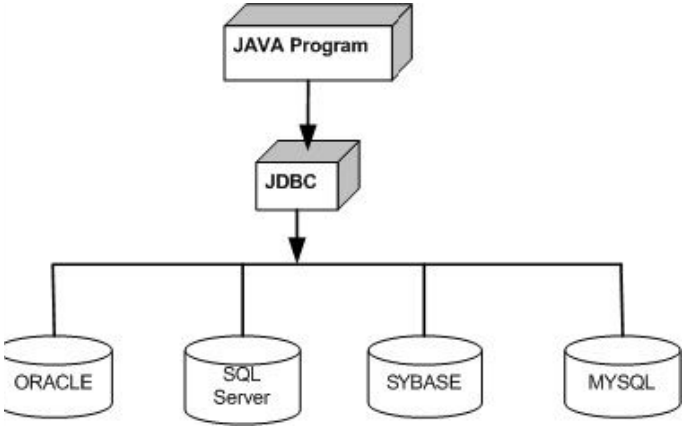
- ODBC: Open Database Connectivity. Defineix una API que poden usar les aplicacions per obrir una connexió amb una base de dades, enviar consultes, actualitzacions i obtenir resultats. Les aplicacions poden usar aquesta API sempre que el servidor de base de dades sigui compatible amb ODBC.
- JDBC. Java Database Connectivity. Defineix una API que poden usar els programes Java per a connectar-se als servidors de bases de dades relacionals.
- OLE-DB. Object Linking and Embedding for Databases. De Microsoft. És una API de C++ amb objectius semblants als de ODBC però per a orígens de dades que no són bases de dades. En ODBC les comandes sempre estan en SQL, en OLE-DB poden estar en qualsevol llenguatge suportat per l'origen de dades.

Accés a dades mitjançant JDBC

- DBC proporciona una llibreria estàndard per accedir a fonts de dades principalment orientades a bases de dades relacionals que usen SQL.
- No sols proveeix una interfície sinó que també defineix una arquitectura estàndard, perquè els fabricants puguin crear els drivers que permetin a les aplicacions Java l'accés a les dades.
- JDBC disposa d'una interfície diferent per a cada base de dades, és el que anomenem driver (controlador o connector). Això permet que les anomenades als mètodes Java de les classes JDBC es corresponguin amb el API de la base de dades

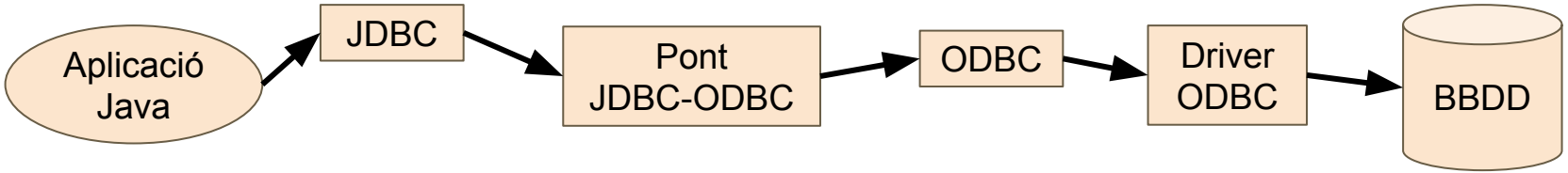
Accés a dades mitjançant JDBC

- JDBC consta d'un conjunt de classes i interfícies que ens permeten escriure aplicacions Java per gestionar les següents tasques amb una bbdd relacional:
  - Connectar-se a la base de dades.
  - Enviar consultes i instruccions DML a la bbdd.
  - Recuperar i processar els resultats rebuts.

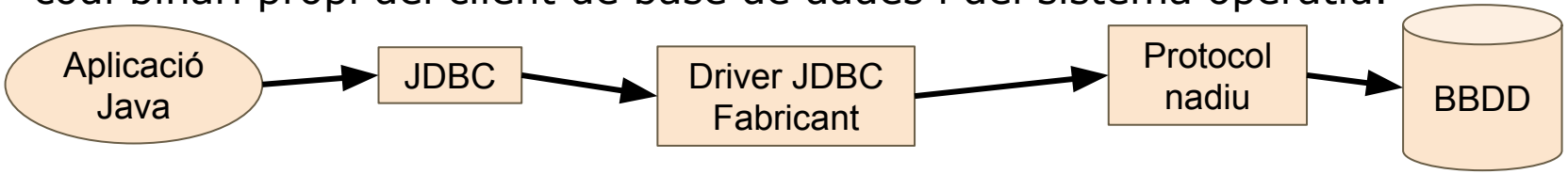


Tipus de drivers

- **Tipus 1 JDBC-ODBC Bridge**: permet l'accés a bases de dades JDBC mitjançant un driver ODBC. Converteix les anomenades al API de JDBC en ODBC. Exigeix la instal·lació i configuració de ODBC en la màquina client.

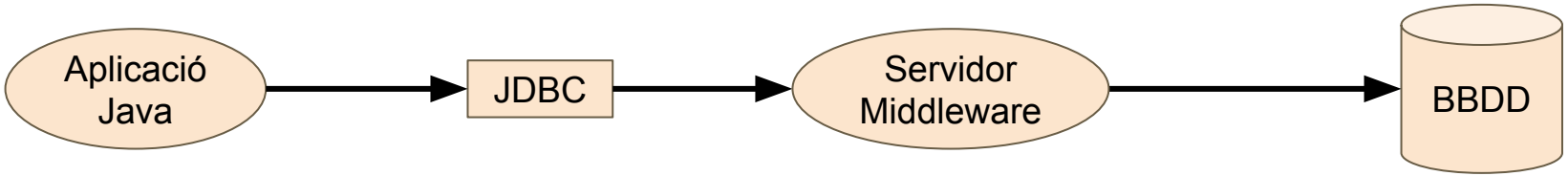


- **Tipus 2 Native**: controlador escrit parcialment a Java i en codi natiu de la base de dades. Tradueix les anomenades al API de JDBC Java en crides pròpies del motor de base de dades. Exigeix instal·lar en la màquina client codi binari propi del client de base de dades i del sistema operatiu.



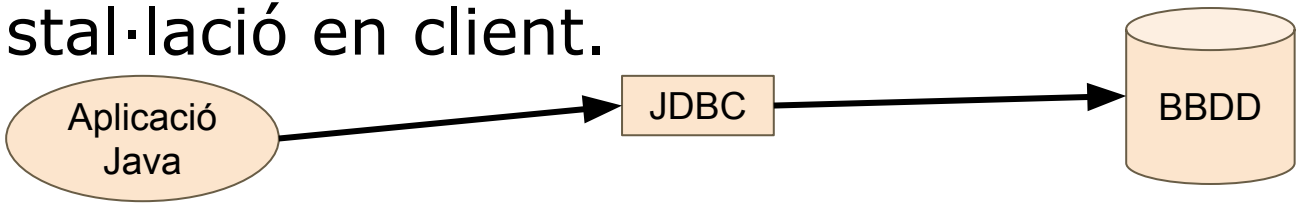
Tipus de drivers

- **Tipus 3 Network**: controlador de Java pur que utilitza un protocol de xarxa (per exemple HTTP) per a comunicar-se amb el servidor de base de dades. Tradueix les anomenades al API de JDBC Java en crides pròpies del protocol de xarxa i a continuació són traduïdes per un programari intermedi (Middleware) al protocol usat per la bbdd. No exigeix instal·lació en client.



Tipus de drivers

- **Tipus 4 Thin**: controlador de Java pur amb protocol natiu. Tradueix les anomenades al API de JDBC Java en crides pròpies del protocol de xarxa usat pel motor de base de dades. No exigeix instal·lació en client.



Els tipus 3 i 4 són la millor manera d'accedir. Els tipus 1 i 2 s'usen normalment quan no queda un altre remei. En la majoria dels casos l'opció més adequada serà el tipus 4.

Com funciona JDBC

- JDBC defineix diverses interfícies que permeten realitzar operacions amb bases de dades; a partir d'elles es deriven les classes corresponents.
- Aquestes classes estan definides en el paquet java.sql.
- El funcionament d'un programa amb JDBC requereix els següents passos:
  - Importar les classes necessàries
  - Carregar el driver JDBC
  - Identificar l'origen de dades
  - Crear un objecte Connection
  - Crear un objecte Statement
  - Executar una consulta amb l'objecte Statement
  - Recuperar les dades de l'objecte ResultSet
  - Alliberar successivament ResultSet, Statement, Connection

Exemple d'accés a MySQL

Preparació màquina

- Descarrega una MV amb MySQL, o un container amb MySQL instal·la'l el MySQL al teu equip o alguna altra alternativa.

- Afegeix la bbdd "**exemple**". Pots utilitzar el següent script sql.
- Descàrrega el connector JDBC per a MySql des d'aquí.
- Afegeix en eclipse una llibreria externa a un nou projecte (properties → libraries → *Add external jar*) que apunte al ".jar" acabat de descarregar
- Còpia el següent codi i executa'l.

```
import java.sql.*;
public class Main {
   public static void main(String[] args) {
       try{
           Class.forName("com.mysql.cj.jdbc.Driver ");
           Connection connexion=DriverManager.getConnection
           ("jdbc:mysql://localhost/exemple ","austria","austria");
           Statement sentencia =connexion.createStatement();
           String sql = "SELECT * from depart ";
           ResultSet result = sentencia.executeQuery(sql);
           while (result.next()){
               System.out.printf("%d, %s, %s, %n",
               result.getInt(1),
               result.getString(2),
               result.getString(3));
           }
           result.close();
           sentencia.close();
           connexion.close();
       } catch (ClassNotFoundException cn) { cn.printStackTrace();
       } catch (SQLException e) {e.printStackTrace();
       }
   }
}
```

Anàlisi de l'anterior exemple

Carregar el Driver:

En primer lloc es carrega el driver amb el mètode forName() de la classe Class (java.lang). Per a tal propòsit se li passa un objecte String amb el nom de la classe del driver com a argument. En l'exemple, com s'accedeix a una base de dades MySql, necessitem carregar el driver com.mysql.jdbc.Driver:

```
Class.forName("com.mysql.cj.jdbc.Driver ");
```

Establir la connexió:

A continuació s'estableix la connexió amb la base de dades (el servidor MySql ha d'estar engegat). Usem la classe DriverManager amb el mètode getConnection() de la següent manera:

```
Connection connexion=DriverManager.getConnection ("jdbc:mysql://localhost/exemple","austria","austria");
```

Anàlisi de l'anterior exemple

La sintaxis del mètode getConnetion() és la següent:

```
public static Connection getConnection (String url, String user, String password)
    throws SQLException
```

El primer paràmetre del mètode getConnection() representa la URL de connexió a la bbdd. Té el següent format per a connectar-se a MySql:

```
jdbc:mysql://nom_host:port/nom_basedades
```

- **jdbc:mysql** → indica que estem utilitzant un driver JDBC per a MySql.
- **nom_host** → indica el nom del servidor on està la base de dades. Aquí pot posar-se una IP, el nom de màquina, o localhost.
- **port** → port on està escoltant el servidor MySql. Per defecte és el 3306. Si no s'indica s'assumeix que és aquest valor.
- **nom_basedades** → nom de la base de dades a la qual volem connectar-nos. Ha d'existir prèviament en MySql.

Executar sentències SQL

A continuació es realitza la consulta, per a això recorrem a la interfície Statement per a crear una sentència. Per a obtenir un objecte Statement es crida al mètode createStatement() d'un objecte Connection vàlid. La sentència obtinguda (o objecte obtingut) té el mètode executeQuery() que serveix per a realitzar una consulta en la base de dades. Se li passa un String que consisteix en la consulta SQL:

```
"SELECT * from depart"
```

El resultat ens el retorna com un ResultSet, que és un objecte similar a una llista en la qual està el resultat de la consulta. Cada element de la llista és un dels registres de la taula "departaments". Recorrem la llista mitjançant el mètode next():

Executar sentències SQL

```
while (result.next()){
                    System.out.printf("%d, %s, %s, %n",
                    result.getInt(1),
                    result.getString(2),
                    result.getString(3));
```

- Els mètodes getInt() i getString() ens van retornant els valors dels camps de cada registre. Entre parèntesi es posa la posició de la columna en la taula. També es pot posar una cadena que indique el nom de la columna:

```
while (result.next()){
       System.out.printf("%d, %s, %s, %n",
       result.getInt("dept_no"),
       result.getString("dnombre"),
       result.getString("loc"));
       }
```

Executar sentències SQL

- ResultSet disposa de diversos mètodes per a moure el punter que apunta a cadascun dels registres retornats per la consulta:
  - boolean next() → Mou el punter una fila cap endavant a partir de la posició actual. Retorna true si el punter es posiciona correctament i false si no hi ha registres en ResultSet.
  - boolean first() → Mou el punter al primer registre de la llista.
  - boolean last() → Mou el punter a l'últim registre de la llista.
  - boolean previous() → Mou el punter al registre anterior de la posició actual.
  - void beforeFirst() → Mou el punter just abans del primer registre.
  - int getRow() → Retorna el número de registre actual. Per al primer registre retorna 1, per al segon 2 i així successivament.
- Finalment, s'alliberen tots els recursos i es tanca la connexió:

```
result.close();
sentencia.close();
conexion.close();
```

Activitat

A1.- Prenent com a base el programa que il·lustra els passos del funcionament de JDBC, obtenir el COGNOM, OFICI i SALARI dels empleats del departament 10 a través d'una aplicació Java.

A2.- Realitza un altre programa Java utilitzant la base de dades "exemple" que visualitzi el COGNOM de l'empleat amb màxim salari, visualitza també el seu SALARI i el nom del departament.

A3.- Realitza un programa que busqui els departaments d'una localitat. El programa sol·licitarà el nom d'una localitat a l'usuari i retornarà els departaments associats a aquesta localitat i els empleats d'aquest departaments.