

Database

Danilo Croce
Maggio 2019-20

ODBC e JDBC

- **Open DataBase Connectivity(ODBC)** introdotto dalla Microsoft per interoperare con diversi DBMS. E' attualmente il più diffuso standard per l'utilizzo di DBMS relazionali.
- **Java DataBase Connectivity(JDBC)** della SUN, fornisce API standard per interrogazione di DBMS
- Il Driver traduce le richieste formulate in JDBC o ODBC in chiamate allo specifico DBMS. Il driver viene caricato dinamicamente a run-time quando viene richiesto ad un *gestore dei driver*, il **Drive Manager**
 - Definizione di un livello standard di astrazione **API (Application Programming Interface)** per accedere alle *capabilities* di DBMS eterogenei.
 - **Portabilità a livello di eseguibile**: permettono di definire un singolo eseguibile per accedere a diversi DBMS (senza dover ricompilare il programma)

ODBC e JDBC

- Un'applicazione che vuole interrogare un data source tramite JDBC o ODBC deve:
 - selezionare il ***data source*** da interrogare
 - caricare dinamicamente il **driver** corrispondente (tramite il Drive Manager)
 - stabilire una **connessione** con il data source
- NB: parliamo di *data source* e non di DBMS perché tramite JDBC e ODBC raggiungiamo un grado di astrazione che ci permette di ottenere i dati tramite query SQL indipendentemente dal tipo di DBMS (sorgente dati) sottostante.

JDBC Caratteristiche

- **API** (Application Programming Interface) Java
- **Uno standard**
 - Può essere utilizzata da diverse componenti (es: Applet, Applicazioni, EJB, Servlet)
- E' (in generale) **indipendente dal DBMS** (caricato a *run-time*)
- **FUNZIONALITA'**:
 - Esecuzione di comandi **SQL** (DML e DDL)
 - Manipolazione di **result set** (insieme di tuple) con cursori
 - Gestione dei **metadati** (ogg.DatabaseMetaData)
 - Gestione delle **transazioni**
 - Definizione di ***stored procedure***

Architettura JDBC

- **Applicazione:** Inizia e termina la connessione ad un data source
- **Driver Manager** :si occupa di caricare i driver JDBC e di passare le chiamate JDBC dell'applicazione al driver specifico
- **Driver** : Stabilisce la connessione con il data source
- **Data Source** : processa i comandi provenienti dal driver e restituisce i dati richiesti

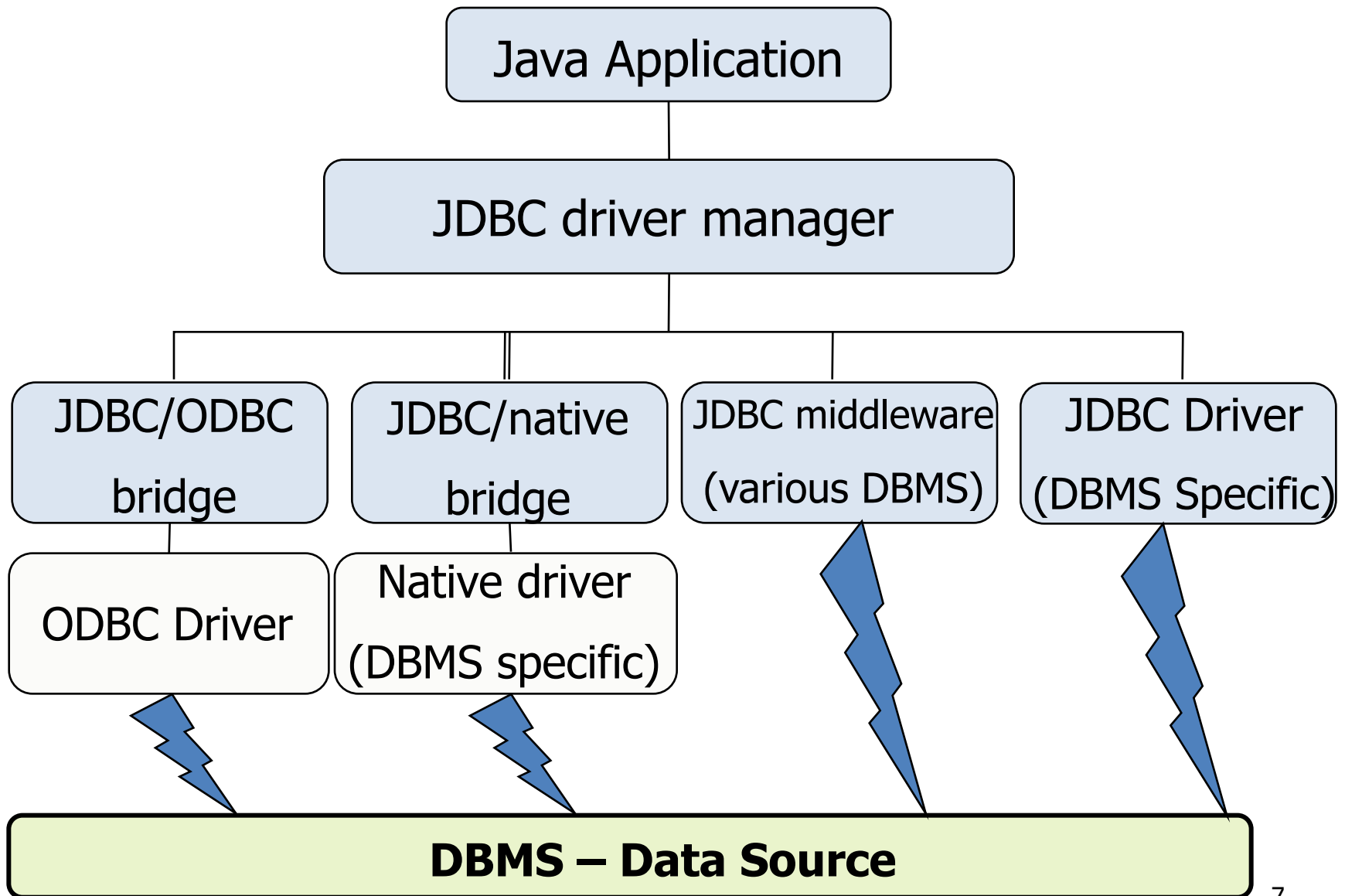
Classi di driver JDBC

<http://industry.java.sun.com/products/jdbc/drivers>

Vengono distinte quattro diverse tipologie di driver JDBC:

- **Classe1-Bridges:** traduce comandi SQL in API non-native (es: JDBC-ODBC Bridge)
- **Classe2-Traduzione diretta ad API native (no Java)**
 - Traduce le query SQL in API native del data source (richiede installazione di sw su client)
- **Classe3-Network Bridges:**
 - spedisce comandi in rete ad un middleware server che dialoga con il data source (non richiede installazione di sw su client)
- **Classe4-Traduzione diretta ad API native (Java based):**
 - converte le chiamate JDBC direttamente nel protocollo di rete utilizzato dal DBMS. Richiede un driver java su ogni client

Architettura JDBC



Applicazione Java connDB (1)

```
import java.sql.*;
```

1 Caricare il driver JDBC:

- `Class.forName("oracle/jdbc.driver.OracleDriver");`
- `Class.forName("com.mysql.jdbc.Driver").newInstance();`

2 Definire l'URL della connessione al Data Base:

- `url="jdbc:mysql://localhost/musicians";`

"jdbc:connectionType://host:port/database"

3 Stabilire la connessione:

```
String user = "nomeutente"; password = "password";
Connection con = DriverManager.getConnection(url, user, pwd);
```

4 Creare un oggetto statement.

```
Statement statement =connection.createStatement();
```


Applicazione Java connDB (2)

5

Eseguire una query : (ad es. INSERT,SELECT,DELETE)

```
String query = "SELECT col1, col2, col3 FROM
table";
```

6

ResultSet results =

```
statement.executeQuery(query);
```

Analizzare/Calcolare i risultati:analisi del risultato contenuto nella classe ResultSet

```
while (results.next()) {
    String a = results.getString(1);
    Integer eta = results.getInt(2);
    System.out.print("NOME= " + a);
    System.out.print("ETA'= " + eta.toString());
    System.out.print("\n");}
```

7

Chiudere/Rilasciare la connessione e lo statement

```
con.close();
statement.close();
```

Java e SQL Data Type Matching

<u>SQL Type</u>	<u>Java Classes</u>	<u>ResultSet get method</u>
BIT	Boolean	getBoolean()
CHAR	String	getString()
VARCHAR	String	getString()
DOUBLE	Double	getDouble()
FLOAT	Double	getDouble()
INTEGER	Integer	getInt()
REAL	Double	getFloat()
DATE	java.sql.Date	getDate()
TIME	java.sql.Time	getTime()
TIMESTAMP	java.sql.TimeStamp	getTimestamp()

- <https://dev.mysql.com/downloads/connector/j/>

