Programmazione Java
Corso Pratico
09 – Connessione a un RDBMS

Maurizio Franco

Cenni su

Database, RDBMS, SQL

Un database o base di dati (DB) è una collezione di dati.

Il database è un insieme di informazioni che vengono suddivise per argomenti in ordine logico (tabelle) e poi tali argomenti vengono suddivisi per categorie (campi).

Connessione a un RDBMS

Un db è gestito tramite un DBMS.

Un DBMS (Data Base Management System) è un sistema software in grado di gestire efficientemente le informazioni necessarie a un sistema informativo, rappresentandone i dati in forma integrata, e garantendone la persistenza.

Connessione a un RDBMS

RDBMS stà per Relational DataBase Management System.

Un RDBMS è un DBMS che usa, come modello logico di rappresentazione dei dati, il modello relazionale.

Un requisito importante di una buona base dati consiste nel non duplicare inutilmente le informazioni in essa contenute: questo è reso possibile dai gestori di database relazionali che consentono di salvare i dati in tabelle che possono essere collegate.

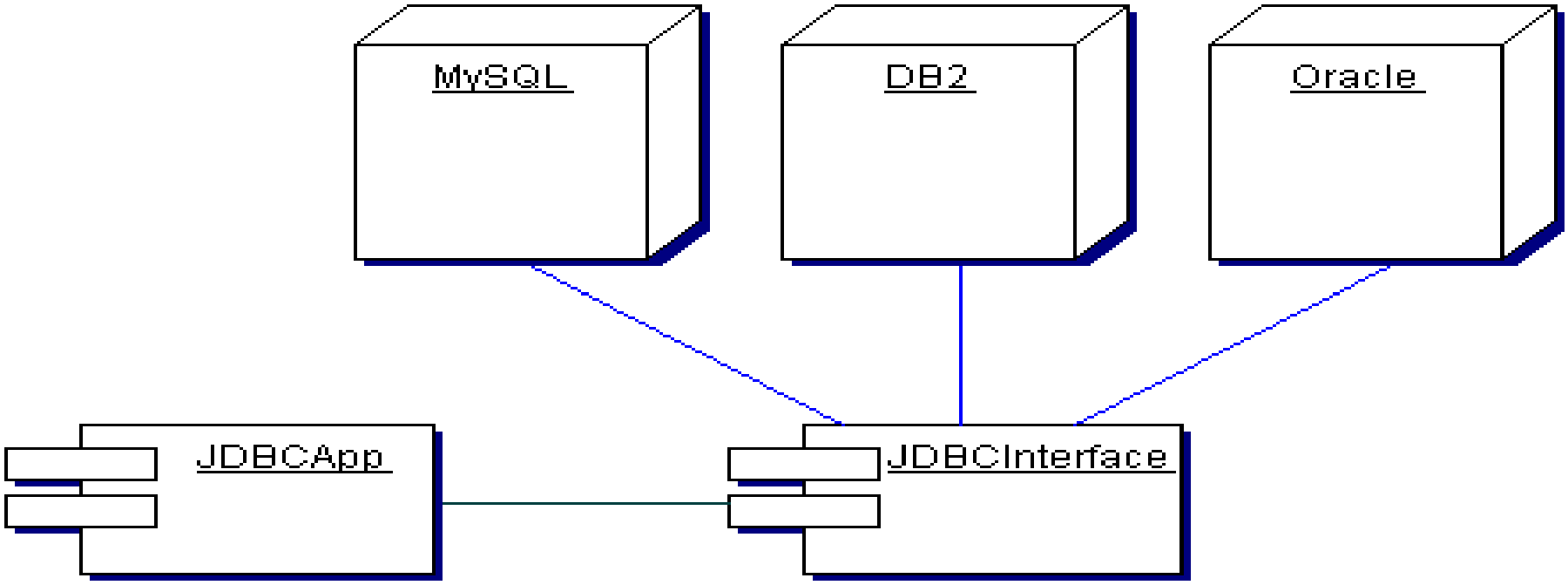
La struttura fondamentale del modello relazionale e' appunto la "relazione", cioe' una tabella bidimensionale costituita da righe (tuple) e colonne (attributi).

SQL: Structured Query Language

Linguaggio standard di gestione e manipolazione di database compatibili.
Con sql è possibile leggere, modificare e gestire dati memorizzati in un sistema basato sul modello relazionale, creare e modificare schema, ecc.

Introduzione a Java per scrivere applicazioni jdbc con accesso a database engine di tipo relazionale(rdbms)

Rappresentazione di applicazione java con accesso tramite jdbc



JDBC

Java Data Base Connectivity

Interfaccia che permette ad un'applicazione java di potersi connettere a diversi RDBMS, senza che venga fatta alcun tipo di modifica al codice.

JDBC pt.2

In tal modo un'applicazione java che utilizza l'interfaccia jdbc, ha la possibilità di essere database engine indipendente

JDBC
è uno strato di astrazione software tra un'applicazione Java ed un database

Connessione a un RDBMS

Per far si che questa indipendenza da db-engine possa esserci, ciascuno vendor di db-engine deve fornire un driver specifico.

Il driver fornirà l'implementazione di alcune interfacce definite all'interno dei packages java.sql e javax.sql: Driver, Connection, Statement, PreparedStatement, CallableStatement, ResultSet, DatabaseMetaData, ResultSetMetaData

Connessione a un RDBMS

Per scrivere un'applicazione che acceda ad un db è necessario quindi

- caricare un driver per la connessione
- aprire una connessione
- creare un oggetto Statement per interrogare il database
- eseguire le operazioni di interazione con il database
- gestire i risultati ottenuti
- gestire la connessione

CARICARE UN DRIVER PER LA CONNESSIONE

```
[.....]
//driver di tipo "bridge jdbc-odbc
String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
Class.forName(driver);
[.....]
```

APRIRE UNA CONNESSIONE

```
 [.....]
//stringa di connessione
String url =
   "jdbc:mysql://localhost:3306/nomedatabase";
//connessione con username e password
Connection con =
DriverManager.getConnection (url, "userName",
   "userPassword");
 [.....]
```

CREARE UN OGGETTO STATEMENT

```
Statement cmd = con.createStatement ();
```

INTERROGARE/ESEGUIRE OPERAZIONI SU DB

```
String query = "SELECT * FROM nome_tabella";

ResultSet res = cmd.executeQuery(query);
```

GESTIONE RISULTATI E CONNESSIONI

```
while (res.next()) {

   System.out.println(res.getString("columnName1"));

   System.out.println(res.getString("columnName2"));
}
res.close();
cmd.close();
con.close();
```

Connessione a un RDBMS

Cenno su classe java.sql.ResultSet

Rappresenta la tabella generata
come risultato dell'esecuzione di una query

Ci si può muovere tra le righe del ResultSet
con un "cursore" manipolato dai metodi

boolean next() – si passa alla successiva
boolean previous() – alla precedente
boolean absolute(int n) – alla riga n
boolean relative(int n) – n righe più avanti
getString(String attr)
getString(int index)
getInt(String attr)
getInt(int index)

Query di comando
(inserimento/modifica/cancellazione)

Simile alla query di selezione

Si invoca il metodo executeUpdate(String query) sullo statement, passando la stringa query come argomento; restituisce il numero di righe coinvolte dall'esecuzione del comando.

Esempio:

```
String comando = "UPDATE MiaTabella where name=\"Maurizio\"; ";
int r = st.executeUpdate(comando);
```

QUERY PARAMETRICHE

Per effettuare query parametriche si utilizza la classe più specializzata java.sql.PreparedStatement
Nella dichiarazione della stringa con il codice sql sono presenti dei punti interrogativi come attributi parametrici
Specificare il valore dei parametri

Esempio:

```
[.....]
String query = "SELECT name FROM MiaTabella WHERE age > ? "
PreparedStatement pStatement = con.prepareStatement(query);
int xxx=10;
pStatement.setInt(1,xxx);
ResultSet rs = ps.executeQuery();
[.....]
```

APACHE DERBY DB

INSTALLIAMO UN DATABASE IN LOCALE

Connessione a un RDBMS

Al momento del download della jdk, abbiamo scaricato in locale anche un database, Java db, basato al 100% su tecnologia java, altro non è che Apache Derby database, un database open-source.

Apache Derby DB, binari, sorgenti e documentazione possiamo trovarlo al seguente url:

https://db.apache.org/derby

Tutta la documentazione relativamente a Java DB la toviamo qui:

http://docs.oracle.com/javadb

Procediamo con il settaggio delle variabili di ambiente:

DERBY_HOME

ed aggiungiamo
%DERBY_HOME%\bin in PATH

e

%DERBY_HOME%\lib in CLASSPATH

Connessione a un RDBMS

Proviamo a far partire il server aprendo un prompt ed eseguendo il comando startNetworkServer

nelle sue varianti:

startNetworkServer -h 0.0.0.0
per avviare il server(standalone) permettendo connessioni da qualsiasi host(ip).

Procediamo con il settaggio delle variabili di ambiente:

DERBY_HOME

ed aggiungiamo
%DERBY_HOME%\bin in PATH

e

%DERBY_HOME%\lib in CLASSPATH