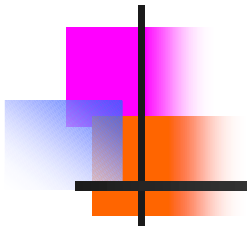


Esercitazione su JDBC

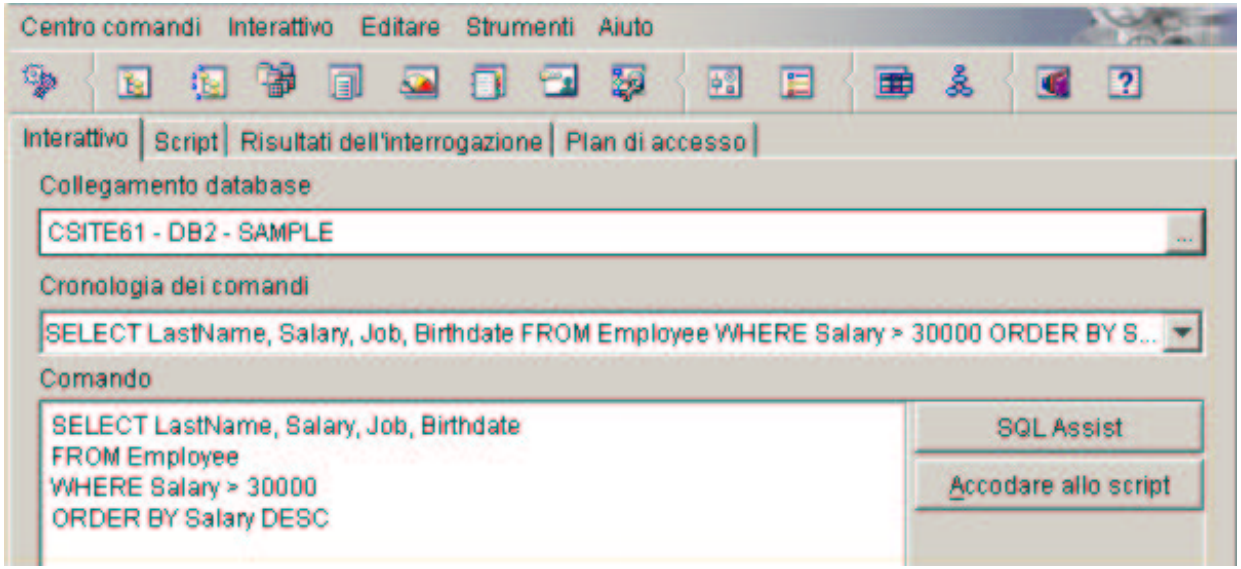
Basi di Dati L

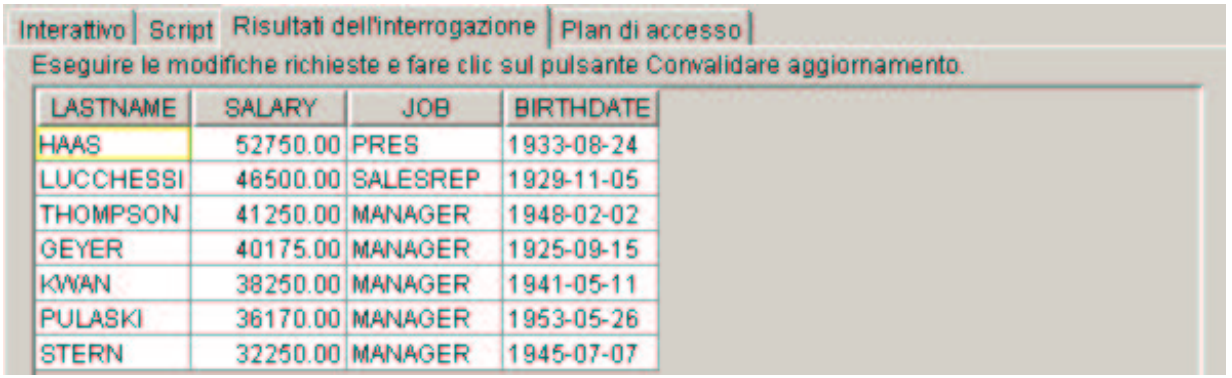
Ingegneria dei Processi Gestionali

(Ilaria Bartolini - Roberto Cabras)

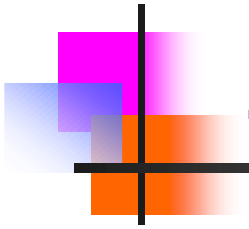
…come usare SQL (1)

- Le istruzioni SQL possono essere eseguite interattivamente…





LASTNAME	SALARY	JOB	BIRTHDATE
HAAS	52750.00	PRES	1933-08-24
LUCCHESSI	46500.00	SALESREP	1929-11-05
THOMPSON	41250.00	MANAGER	1948-02-02
GEYER	40175.00	MANAGER	1925-09-15
KWAN	38250.00	MANAGER	1941-05-11
PULASKI	36170.00	MANAGER	1953-05-26
STERN	32250.00	MANAGER	1945-07-07



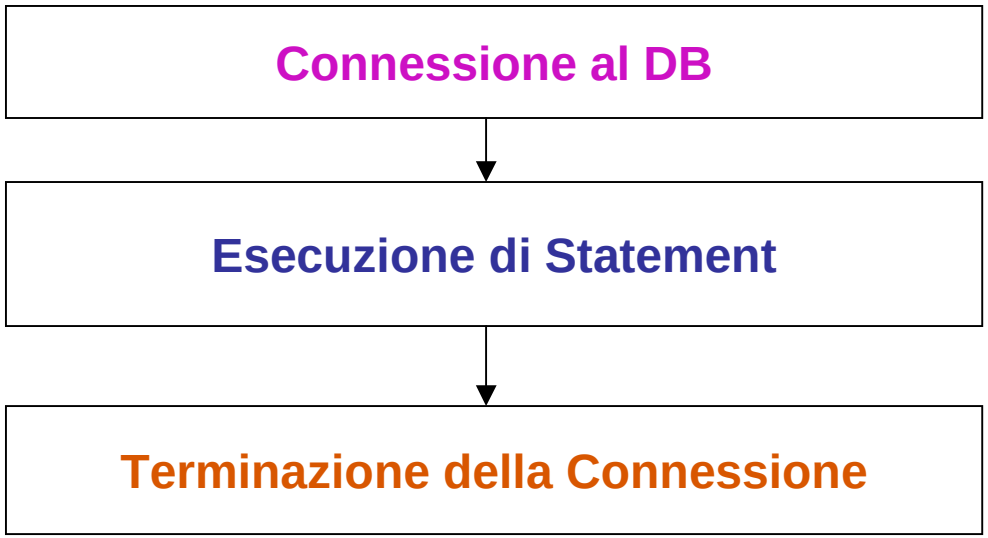
…come usare SQL (2)

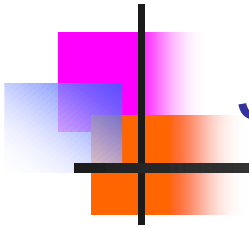
- … o inserendole nel codice di un'applicazione scritta in un linguaggio di programmazione "ospite" (ad es. Java)

```
System.out.println("Retrieve some data from the database...");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM employee");
// display the result set
while (rs.next()) {
            String a = rs.getString(1);
            String str = rs.getString(2);
            System.out.print(" empno= " + a);
            System.out.print(" firstname= " + str);
            System.out.print("\n");
          }
          rs.close();
          stmt.close();
```

DB2 e sviluppo software di base

- DB2 mette a disposizione interfacce di programmazione (API) per i principali linguaggi di programmazione, quali C/C++, Java, VB…
- Un esempio d'uso di base di DB2 può essere quello di costruire applicazioni client che, conoscendo la struttura delle tabelle del database (DB) residente sul server e utilizzando un API per connettersi al DB stesso, interrogano e/o aggiornano il contenuto delle tabelle mediante statement SQL

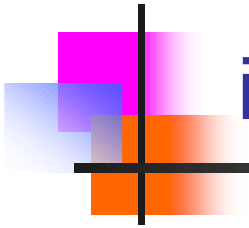




JDBC (Java Database Connectivity)

- Cos'è JDBC?
  - È un API Java per programmi (scritti in Java) di connessione a dati residenti in DB relazionali
  - Consiste di un insieme di classi e interfacce scritte nel linguaggio di programmazione Java (`package java.sql`)
  - Fornisce un API standard per sviluppatori di tool/DB e rende possibile scrivere applicazioni DB usando un API Java pura

- Cosa fa JDBC?
  - Stabilisce una connessione a un DB
  - Invia istruzioni SQL
  - Processa i risultati



interfacce e classi JDBC (1)

- Interfaccia Driver: si occupa (per un DBMS specifico) della connessione, e dell'autenticazione. Ogni driver ha una stringa di connessione che riconosce nella forma: **`jdbc:product_name:database_alias`**

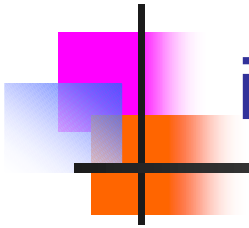
  in cui **`database_alias`** specifica a quale database del DBMS connettersi

  (…nel caso del laboratorio CCIB: **`jdbc:db2:sample/studenti`**)

- Classe DriverManager: in essa vengono registrati i driver dei diversi DBMS e ad essa si rivolgono le richieste di connessione

- Interfaccia Connection: rappresenta una connessione a un DB

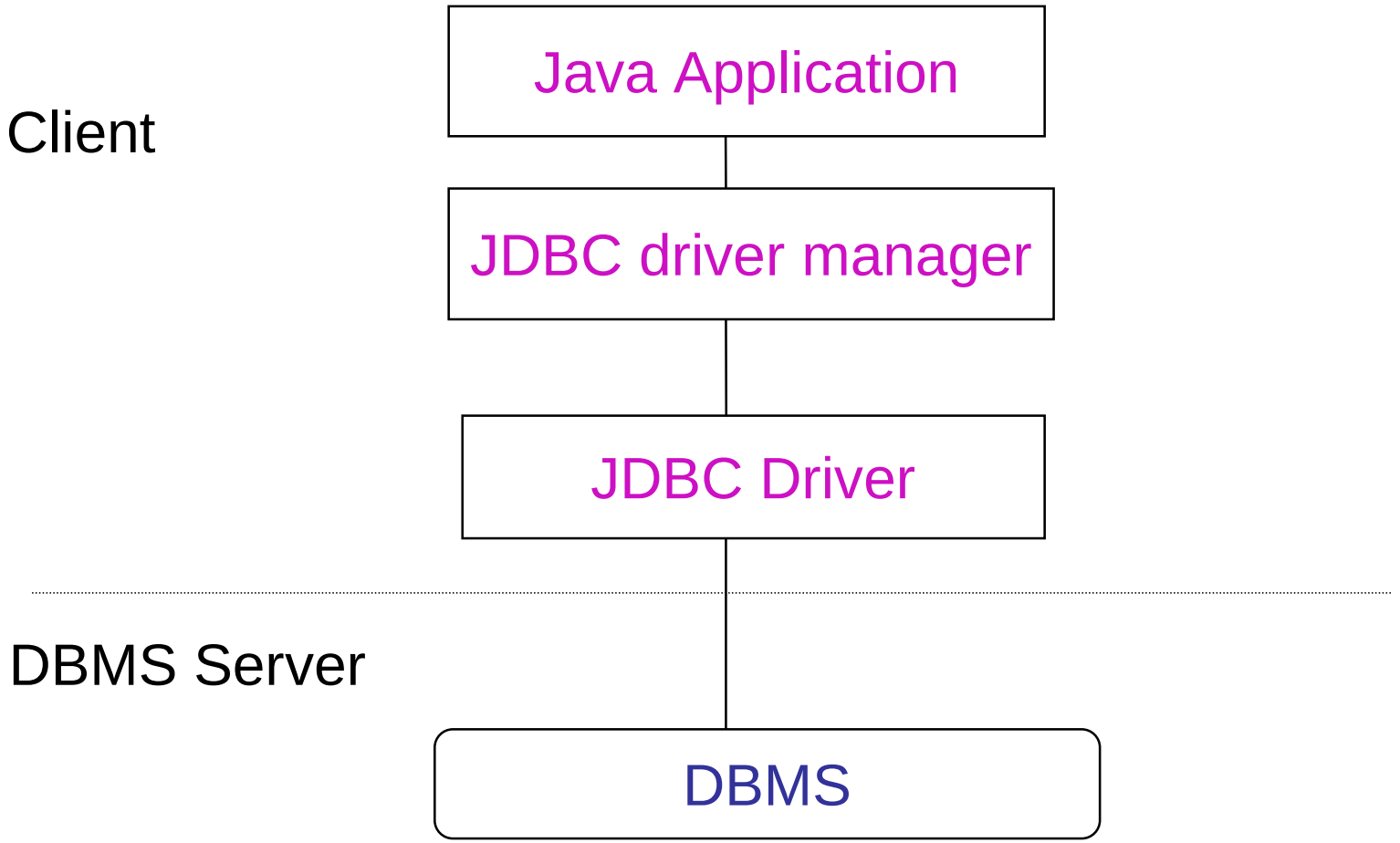
- Interfaccia Statement: si occupa degli statement di interrogazione e/o aggiornamento

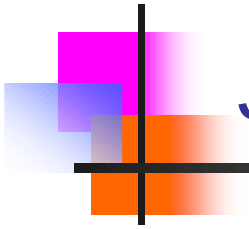


interfacce e classi JDBC (2)

- Interfaccia PreparedStatement: si occupa dell'esecuzione di query compilate
- Interfaccia CallableStatement: si occupa dell'esecuzione stored procedure
- Interfaccia ResultSet: rappresenta i dati restituiti dalle interrogazioni

Struttura di un'applicazione JDBC



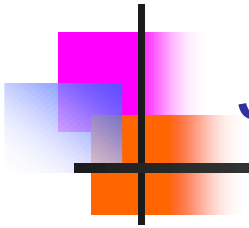


JDBC e DB2

- DB2 prevede 2 driver JDBC:
  1. driver network–enabled **`COM.ibm.db2.jdbc.net.DB2Driver`**
  2. driver nativo **`COM.ibm.db2.jdbc.app.DB2Driver`**

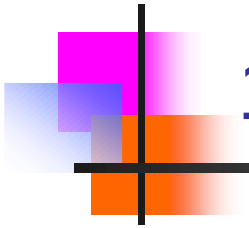
  entrambi sono contenuti nel file **`db2java.zip`** (direttorio **`\sqllib\java`**)

- driver network–enabled serve per connettersi attraverso la rete (Es. DBASE2) a server remoti che abbiano in esecuzione il servizio DB2 Jdbc Applet Server
- driver nativo permette di connettersi ad istanze DB2 residenti sulla macchina locale o catalogate localmente



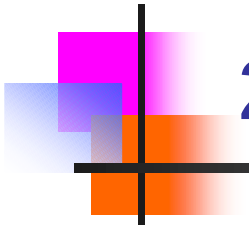
JDBC: passi principali

1. Importazione dei package
2. Registrazione dei driver JDBC
3. Apertura di una connessione al DB (Connection)
4. Creazione di un oggetto Statement
5. Esecuzione di una query e restituzione di un oggetto ResultSet
6. Utilizzazione dell'oggetto ResultSet
7. Chiusura degli oggetti ResultSet e Statement
8. Chiusura della Connection



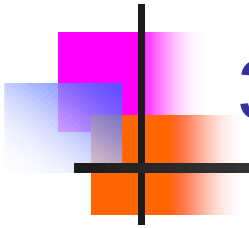
1: Importazione package

```
// Questo programma mostra un semplice esempio di
// applicazione Java (Esempio.java) in grado di eseguire
// interrogazioni/aggiornamenti sul database DB2 SAMPLE
// utilizzando JDBC


   //importazione package
   import java.sql.*;        //package JDBC
```



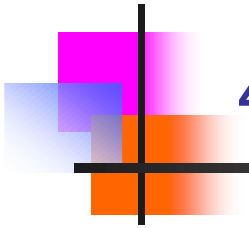
2: Registrazione driver JDBC

```
class Esempio {

 static {
   try {
   // caricamento e registrazione driver
    Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();
   } catch (Exception e) {
         e.printStackTrace();
 }
}
```



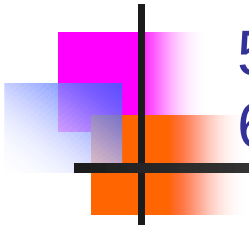
3: Apertura connessione database

```
public static void main(String argv[]) {
      Connection con = null;

      // URL jdbc:db2:database_alias
      String url = "jdbc:db2:sample";
      try {
         if (argv.length == 2) {
            String userid = argv[0];
            String passwd = argv[1];

            // connessione con id/password forniti dall'utente
            con = DriverManager.getConnection(url, userid, passwd);
            }
         else {
            System.out.println("\nUsage: java Esempio username password\n");
            System.exit(0);
         }
```



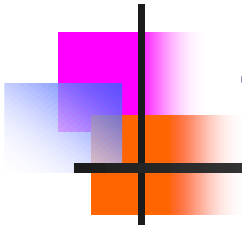
4. Creazione oggetto Statement

```
// interrogazione table EMPLOYEE
 System.out.println("Retrieve some data from the database...");

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



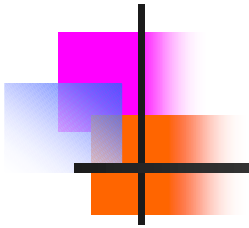
5. Esecuzione query, restituzione oggetto ResultSet & 6. Utilizzazione oggetto ResultSet

```
// esegue la query
ResultSet rs = stmt.executeQuery
                              ("SELECT * FROM DB2ADMIN.EMPLOYEE");
System.out.println("Received results:");
// mostra i risultati
// rs.next() = “false” se non ci sono più righe risultato
     while (rs.next()) {
         String a = rs.getString(1);
         String str = rs.getString(2);

         System.out.print(" empno= " + a);
         System.out.print(" firstname= " + str);
         System.out.print("\n");
      }
```



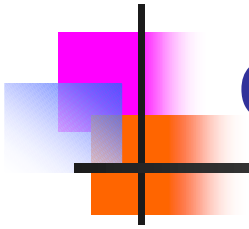
7. Chiusura oggetti ResultSet e Statement

```
// chiude ResultSet e Statement
   rs.close();
   stmt.close();
```



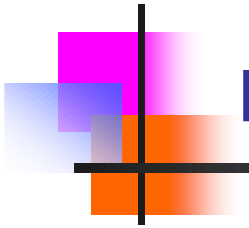
… Esecuzione di altre istruzioni SQL
8. Chiusura Connection

```
// aggiorna il database… prova ad aggiornarlo! ☺
   System.out.println("\n\nUpdate the database... ");
   stmt = con.createStatement();
   int rowsUpdated = stmt.executeUpdate("UPDATE DB2ADMIN.EMPLOYEE
                        SET firstnme = 'SHILI' WHERE empno = '000010'");
   System.out.print("Changed "+rowsUpdated);
   if (1 == rowsUpdated)
       System.out.println(" row.");
   else
       System.out.println(" rows.");
   stmt.close(); // chiude Statement
   con.close(); // chiude Connection
} catch( Exception e ) {
         e.printStackTrace();
}
}
}
```



Corrispondenze tipi di dato SQL - Java

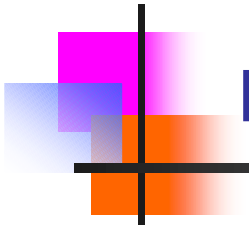
- Alcuni tipi di dato specifici di SQL devono essere mappati in corrispondenti tipi di dato Java per poter essere utilizzati
- La conversione riguarda tre categorie:
  - Alcuni tipi di dato SQL hanno i diretti equivalenti in Java e possono essere letti direttamente nei tipi Java (esempio: il tipo **`INTEGER`** SQL è equivalente al tipi **`int`** di Java)
  - Alcuni tipi di dato SQL possono essere convertiti negli equivalenti tipi di Java (esempio: il tipo SQL **`CHAR`** e **`VARCHAR`** possono essere convertiti nel tipo **`String`** di Java)
  - Una minoranza di tipi di dato SQL sono unici e necessitano della creazione di uno speciale oggetto Java, relativo a una classe dato, per ottenere l'equivalente SQL (esempio: il tipo SQL **`DATE`** si converte nell'oggetto **`Date`** definito dall'omonima classe Java)

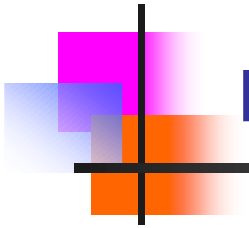


Esempio con Access e ODBC Prima Parte

```
// Registrazione del Package
import java.net.URL;
import java.sql.*;
public class ProvaJDBC
// Registrazione del driver ODBC
public static void main(String args[]) {
  String url = "jdbc:odbc:prova";   \\ prova = nome
  della sorgente dati ODBC
```

Esempio con Access e ODBC Seconda Parte

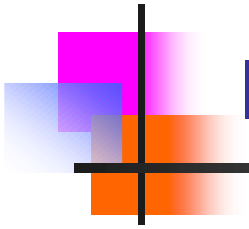
- **// Driver – Connessione - Statement**
- ```
  Connection con;
  String queryString;
  queryString = "SELECT * FROM Contatti “;
  Statement stmt;
  try {
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

  } catch(java.lang.ClassNotFoundException e) {
  System.err.print("ClassNotFoundException: ");
  System.err.println(e.getMessage());
  }
  try {
  con = DriverManager.getConnection(url, "LOGIN", "PASSWD");

  stmt = con.createStatement();
   ResultSet rs stmt.executeQuery(queryString);
  ```
  ```



Esempio con Access e ODBC Terza Parte

```
// mostra i risultati
 // Come con DB2. piccola variante per getString
      while (rs.next()) {
          String sNome    = rs.getString(“Nome”);
          String sCognome = rs.getString(“Cognome”);

          System.out.print(" nome= " + sNome);
          System.out.print(" cognome= " + sCognome);
          System.out.print("\n");
       }
```



Esempio con Access e ODBC Conclusione

- **// Nessun cambiamento rispetto a JDBC**
- ```
  stmt.close();
  con.close();

  } catch(SQLException ex) {
  System.err.println("SQLException: " +
  ex.getMessage());
  }
  }
  }
  ```
  ```

Esempio in Asp con Jet Engine e ADO

```
‘ Il principio non cambia : cambia solo la sintassi
Dim con, rs
        sSource = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" &         Server.mappath(“agenda.mdb") & ";"

    set con = server.createobject("adodb.connection")
‘connessione
    con.open sSource
    set rs = server.createobject("adodb.recordset")
    SqlStat="SELECT * FROM Contatti;"
    rs.Open SqlStat, con
    while not rs.eof
           response.write rs.fields(“Nome”)
           response.write rs.fields(“Cognome”)
        loop
        rs.close()  ‘chiusura
        set rs = nothing
```



Informazioni Utili

- Per la documentazione relativa al pakage java.sql seguire il collegamento Java2Docs che si trova sul desktop

- Per saperne di più:
  - **`http://java.sun.com/products/jdbc/`**
  - **`http://java.sun.com/j2se/1.3/docs/guide/jdbc/index.html`**