Tecnologie di Sviluppo per il Web

Programmazione su Basi di Dati: JDBC
Dettagli e Approfondimenti

versione 3.2

G. Mecca – mecca@unibas.it – Università della Basilicata

Sommario

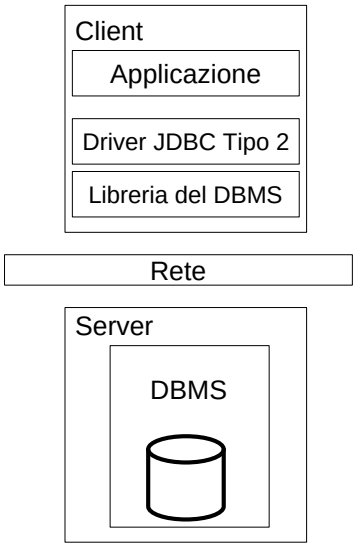
- Tipi di Driver
- Statement
- Stored Procedures
- Tipi di ResultSet
- Tipi di dato
- Metadati

Tipi di Driver

- Esistono vari tipi di driver
  - ⇨Tipo 1 - “JDBC-ODBC Bridge”
  - ⇨Tipo 2 - Parte Java, Parte API Nativa
  - ⇨Tipo 3 - Driver di Rete Tutto in Java
  - ⇨Tipo 4 - Driver Nativo Tutto in Java
- Nei DBMS moderni
  - ⇨tipicamente driver di tipo 4, tutti scritti in Java
  - ⇨per comodità faremo riferimento anche al driver di tipo 1

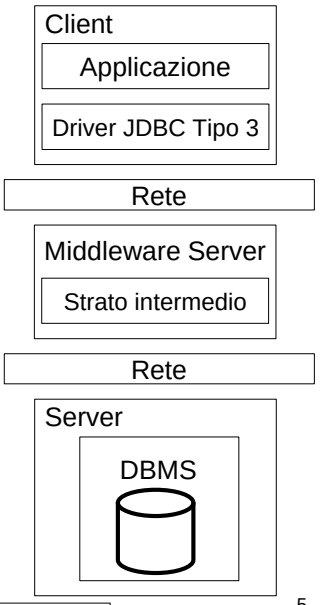
Driver di Tipo 2

- Parte Java, Parte API Nativa
  - ⇨sfrutta una libreria del DBMS
  - ⇨traduce le chiamate in chiamate alla libreria
  - ⇨migliori prestazioni
  - ⇨non indipendente dalla piattaforma



Driver di Tipo 3

- Driver di Rete Tutto Java
  - sfrutta un server intermedio
  - driver JDBC leggero
  - client indipendente dalla piattaforma
  - servizi avanzati (caching)
  - architettura complessa
  - tempi di trasferimento



Tipi di Driver

- Tipo 1 fornito a corredo di J2SDK
  - classe sun.jdbc.odbc.JdbcOdbcDriver
- Tipo 2
  - poco usato
- Tipo 3
  - applicazioni di fascia alta
  - servizi avanzati
- Tipo 4
  - maggiore semplicità
  - architettura semplificata

Statement

- Finora
  - ⇨due tipologie di statement
- Statement ordinari
  - ⇨costruiti specificando il testo della query SQL
- Statement preparati
  - ⇨statement precompilati di cui è necessario specificare successivamente i valori per i parametri

Statement

- Statement ordinari
  - ⇨metodo createStatement()
- Esecuzione
  - ⇨int executeUpdate(): int serve a restituire un codice di risultato
  - ⇨ResultSet executeQuery()
- Esiste anche un metodo execute()
  - ⇨restituisce più di un resultSet; poco usato

Statement

- E' possibile
  - limitare il numero di ennuple del ResultSet restituito da uno statement
  - void setMaxRows(int numeroRighe)
  - int getMaxRows()
- Esempio
  - st.setMaxRows(10);
  - i ResultSet generati conterranno al più 10 ennuple

Stored Procedures

- Una terza categoria di statement
  - utilizzati per la chiamata delle stored procedures
  - Interfaccia CallableStatement, estende PreparedStatement
- Sintassi per la chiamata della procedura
  - {call <nomeProc>[(<parametri>)]}
  - i parametri sono indicati da "?"

Stored Procedures

- Creazione di un CallableStatement
  - ⇨metodo prepareCall di Connection
  - ⇨CallableStatement prepareCall(String chiamata);
- Esempio

  CallableStatement cs =
  conn.prepareCall("{call contaCitta(?,?,?)}");
- Passi successivi
  - ⇨impostare i parametri
  - ⇨effettuare la chiamata e prelevare i risultati

Stored Procedures

- Impostare i parametri di ingresso
  - ⇨metodi setXXX
  - ⇨es: cs.setString(1, citta);
- Impostare i parametri di uscita
  - ⇨è necessario specificarne il tipo JDBC
  - ⇨metodo registerOutParameter
  - ⇨es: cs.registerOutParameter(2,
    java.sql.Types.INTEGER)

Stored Procedures

- Chiamare la procedura
  - ⇨metodo execute()
- Prelevare i parametri di uscita
  - ⇨metodo getXXX
  - ⇨si applicano all'oggetto statement e non al resultSet
  - ⇨es: int numProprietari = cs.getInt(2);
  - ⇨es: int numAutomobili = cs.getInt(3);

Stored Procedures

- Il metodo stampaStatistiche
  - ⇨supponiamo che sia disponibile la stored procedure contaCitta
  - ⇨il metodo legge un nome di città (Stringa)
  - ⇨calcola e stampa il numero di auto di proprietari residenti nella città

Stored Procedures: Esempio

```
public void stampaStatistiche() {
   String citta = Console.leggiStringa();
   try {
      Connection conn = DataSource.getConnection();
      CallableStatement cs = conn.prepareCall("{call contaCitta(?,?,?)}");
      cs.setString(1, citta);
      cs.registerOutParameter(2, java.sql.Types.INTEGER);
      cs.registerOutParameter(3, java.sql.Types.INTEGER);
      cs.execute();
      int numProprietari = cs.getInt(2);
      int numAuto = cs.getInt(3);
      System.out.println(numProprietari + "" + numAuto);
    } catch(SQLException se) {
      System.out.println("ERRORE:"+ se);
    }
}
```


Tipi di ResultSet

- ResultSet
  - ⇨è una collezione di ennuple che può essere scandita con un cursore
  - ⇨metodo boolean next()
- Ci sono vari tipi di ResultSet
  - ⇨scandibili solo in avanti
  - ⇨scandibili anche indietro
  - ⇨sensibili alle modifiche della base di dati
  - ⇨aggiornabili

Tipi di ResultSet

- Tipo standard
  - ⇨in avanti, non sensibile, non aggiornabile
- Altri tipi
  - ⇨variante del metodo createStatement()
  - ⇨Statement createStatement(int tipoDiScansione, int tipoAggiornamento)
  - ⇨i valori che è possibile utilizzare corrispondono a costanti definite in ResultSet

Tipo di Scansione

- Scandibile solo in avanti
  - ⇨ResultSet.TYPE_FORWARD_ONLY
- Scandibile avanti e indietro, insensibile ai cambiamenti nella base di dati
  - ⇨ResultSet.TYPE_SCROLL_INSENSITIVE
- Scandibile avanti e indietro, sensibile ai cambiamenti nella base di dati
  - ⇨ResultSet.TYPE_SCROLL_SENSITIVE

Tipo di Aggiornamento

- Non aggiornabile
  - ⇨ResultSet.CONCUR_READ_ONLY
- Aggiornabile
  - ⇨ResultSet.CONCUR_UPDATABLE
- Attenzione: non sempre implementati
- Es: avanti e indietro, sensibile, non agg.

```
Statement st = conn.createStatement(
  ResultSet.TYPE_SCROLL_SENSITIVE,
  ResultSet.CONCUR_READ_ONLY);
```


Metodi per la Scansione

- Scansione avanti e indietro
  - ⇨boolean next()
  - ⇨boolean previous()
  - ⇨boolean first()
  - ⇨boolean last()
  - ⇨boolean absolute(int n)
  - ⇨boolean relative(int n)

Tipi di Dato

- Trasferimento di dati da e verso il DBMS
  - è necessario mettere in corrispondenza tipi Java e tipi del DBMS
  - metodi getXXX() sui ResultSet
  - metodi setXXX() sui PreparedStatement
- Problema
  - servono regole di corrispondenza
  - come avere regole di corrispondenza valide per tutti i DBMS e i loro tipi ?

Tipi di Dato

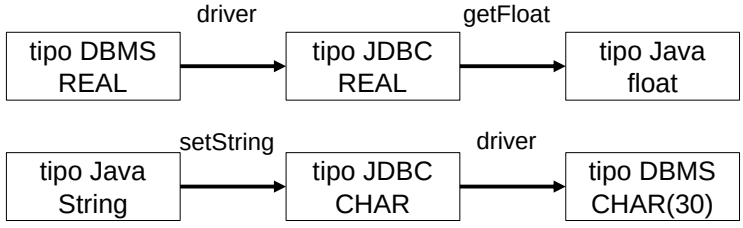
- Soluzione
  - JDBC stabilisce un insieme nutrito ma limitato di tipi utilizzabili
  - elencati nella classe java.sql.Types
  - le regole di corrispondenza sono fissate rispetto ai tipi JDBC
  - ciascun driver mette in corrispondenza ciascun tipo del DBMS con un tipo di JDBC e viceversa

Tabella di Corrispondenza per getXXX

SQL Type	Java Method	SQL Type	Java Method
BIGINT	getLong()	LONGVARCHAR	getString()
BINARY	getBytes()	NUMERIC	getBigDecimal()
BIT	getBoolean()	OTHER	getObject()
CHAR	getString()	REAL	getFloat()
DATE	getDate()	SMALLINT	getShort()
DECIMAL	getBigDecimal()	TIME	getTime()
DOUBLE	getDouble()	TIMESTAMP	getTimestamp()
FLOAT	getDouble()	TINYINT	getByte()
INTEGER	getInt()	VARBINARY	getBytes()
LONGVARBINARY	getBytes()	VARCHAR	getString()

Tipi di Dato

- Metodi getXXX e setXXX
  - ⇨vengono scelti secondo la tabella di corrispondenza con i tipi JDBC
  - ⇨conversione a due passi della rappresentaz.



Tipi di Dato

- Un caso particolare
  - ⇨istruzioni CREATE TABLE da JDBC
  - ⇨quali tipi specificare per gli attributi ?
- Soluzione
  - ⇨per essere portabili, le applicazioni devono essere scritte con riferimento ai tipi JDBC (e non ai tipi di uno specifico DBMS)
  - ⇨il driver si occupa di tradurre verso i tipi del DBMS

Tipi di Dato

- Esempio: testi lunghi
  - ⇨PostgreSQL: TEXT
  - ⇨Oracle: LONG VARCHAR
  - ⇨Access: LONGTEXT
- Per rendere il codice portabile:
  - ⇨nell'istruzione CREATE TABLE uso il tipo JDBC corrispondente: LONGVARCHAR
  - ⇨i driver effettueranno la conversione

Metadati

- Finora
  - abbiamo discusso applicazioni client-server tradizionali (sistemi informativi)
  - utilizzano una base di dati nota
- Un'altra categoria di applicazioni
  - client generici
  - es: un'applicazione che consente di connettersi ad un DBMS ed esplorarne le basi di dati, eventualmente modificandole

Metadati

- Problema
  - in questo secondo caso è necessario lavorare con più basi di dati, non necessariamente note
- Soluzione
  - interfacce per l'accesso a "metadati" (dati che descrivono i dati)
- E' possibile usarli per
  - esplorare il catalogo del DBMS
  - esplorare le funzionalità del DBMS (es: tipi supportati)

Metadati

- I metadati di JDBC
  - ⇨DatabaseMetaData – collezione di metodi per esplorare il catalogo della bd
  - ⇨ResultSetMetaData – collezione di metodi per esplorare un ResultSet
  - ⇨sono interfacce complesse

>> DatabaseMetadata
>> ResultSetMetadata

Riassumendo

- Tipi di Driver
- Statement
- Stored Procedures
- Tipi di ResultSet
- Tipi di dato
- Metadati

Termini della Licenza