Laboratorio di Programmazione di Rete

Docente: Novella Bartolini

Lezione del 7 giugno 2010

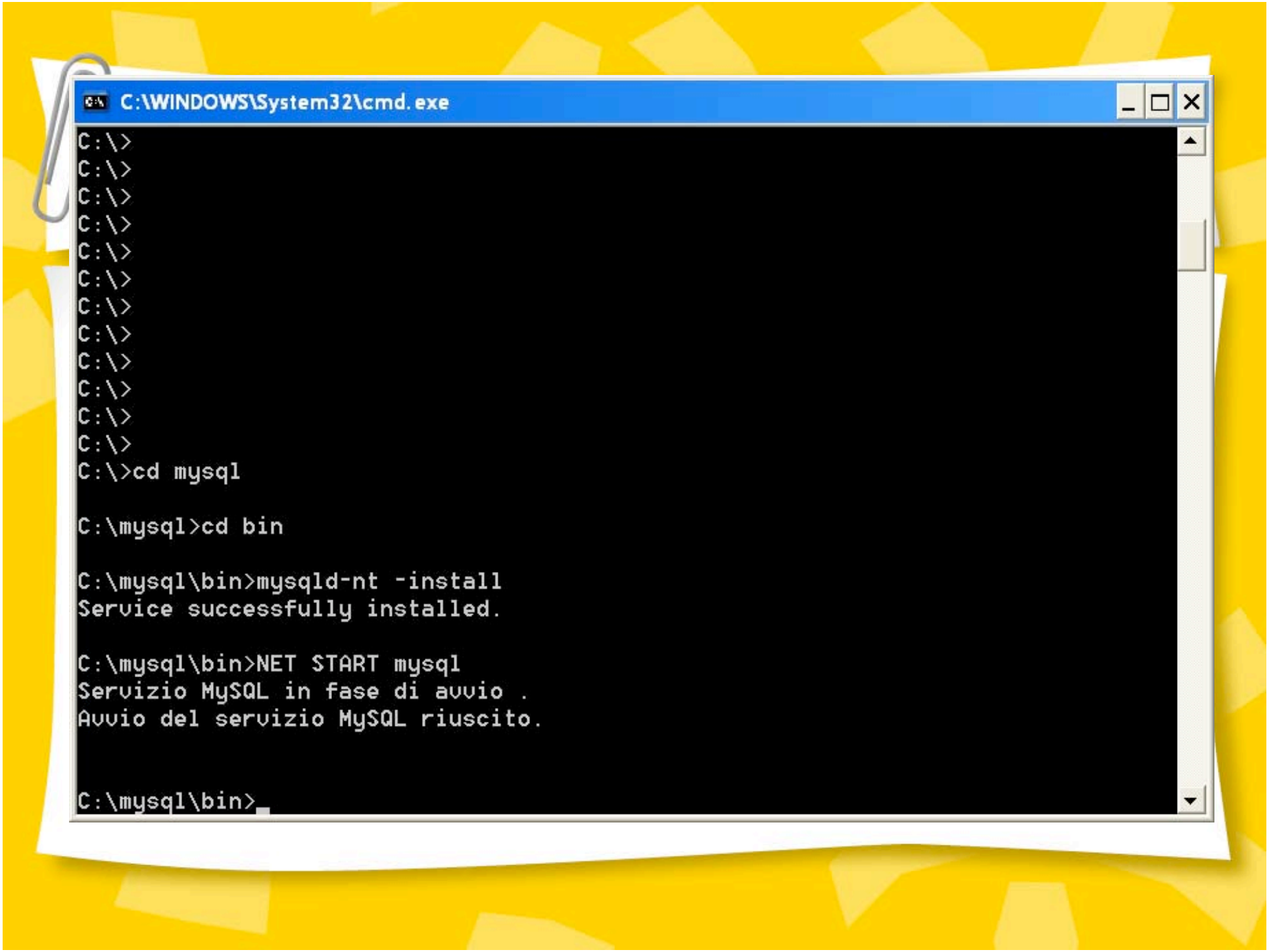
Utilizzo di basi di dati da servlet/JSP

Installazione di MySQL

- Installazione di MySQL
  - scaricare il file ***mysql-4.0.18-win.zip*** dalla pagina del corso
  - eseguire setup
  - configurare le variabili di ambiente aggiungendo il percorso **`...\mysql\bin`** al ***PATH***
- Avviare il server
  - eseguire **`...\mysql\bin\mysqld`**

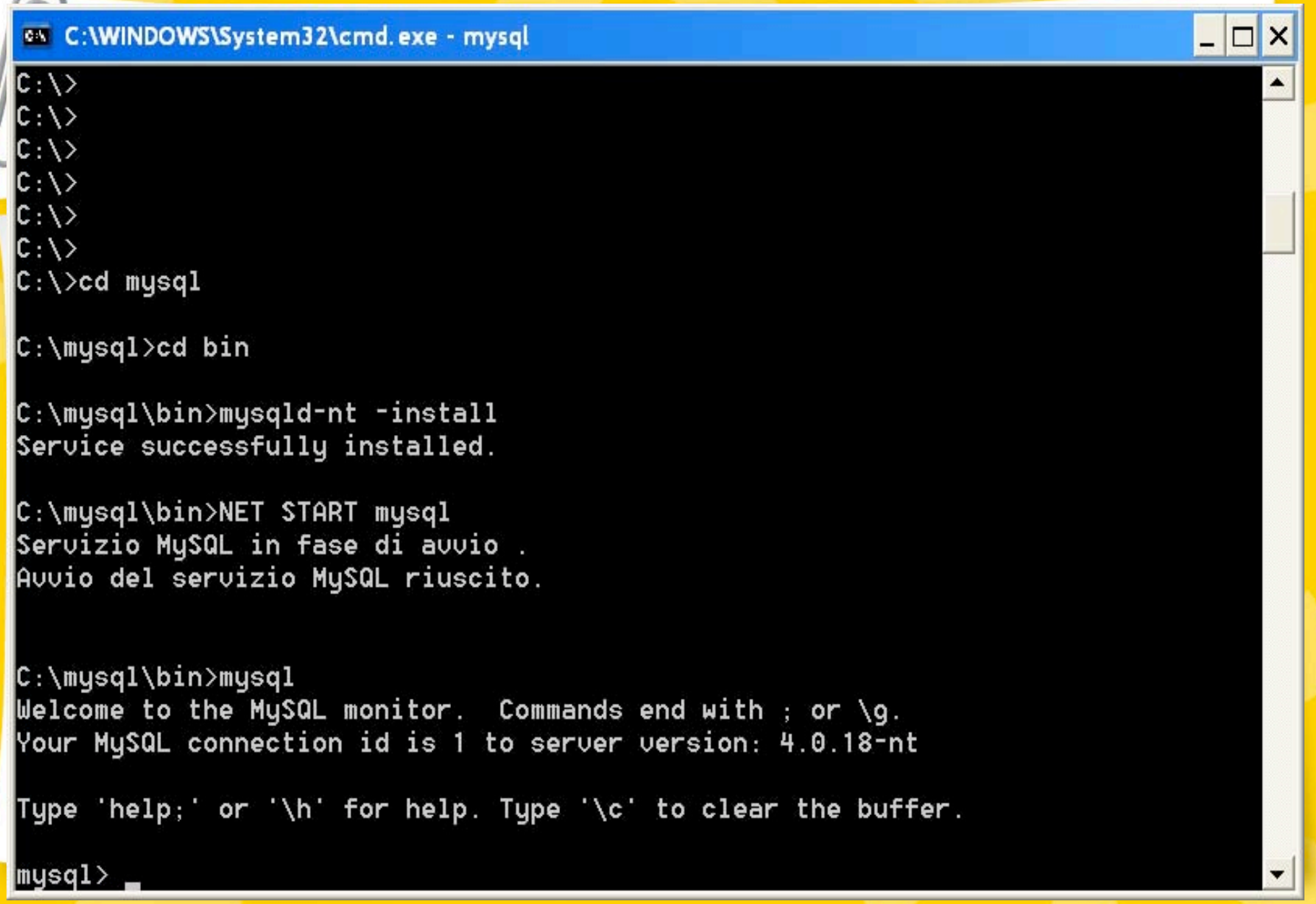
Il server MySQL (cont.)

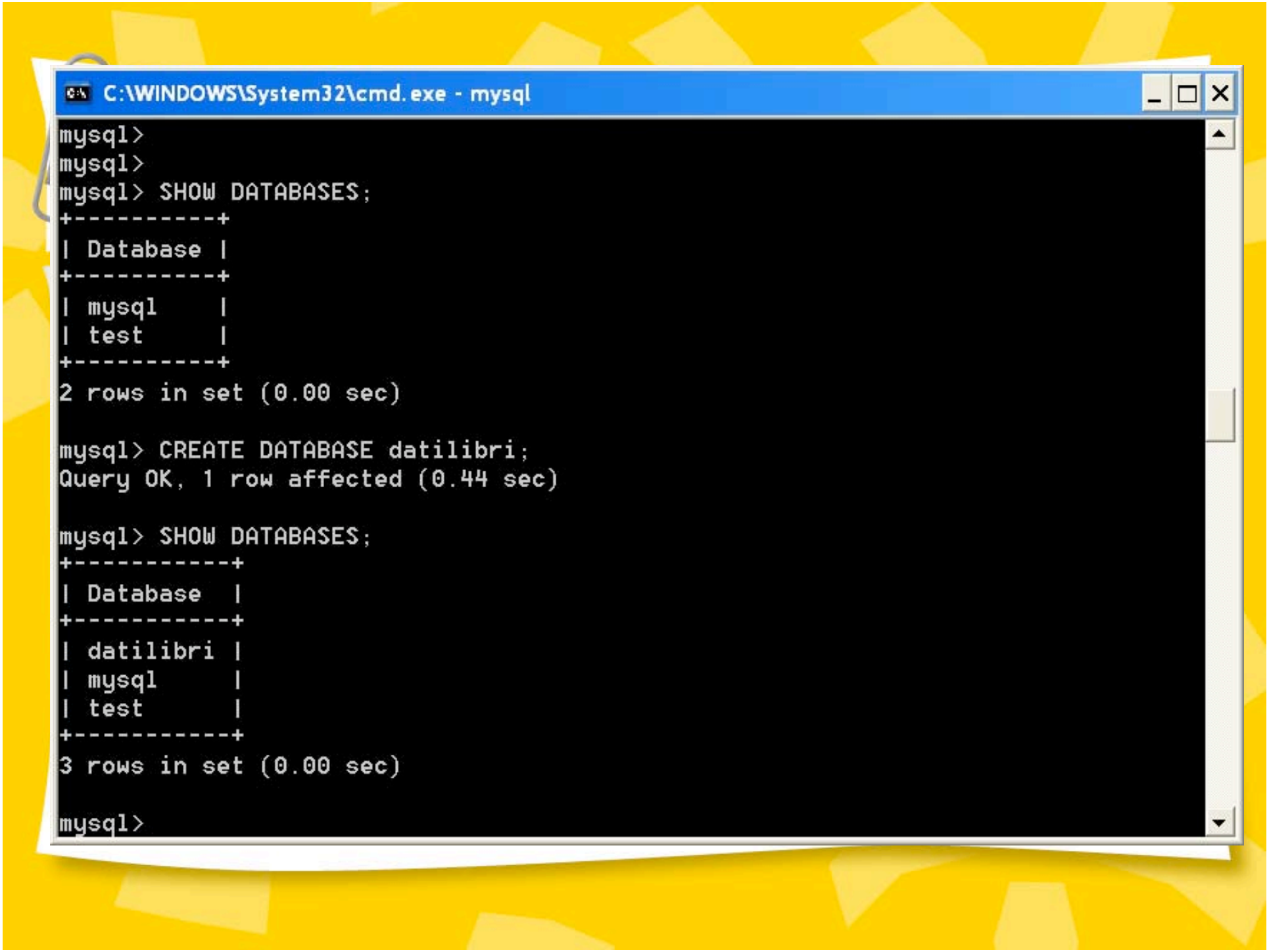
- Per installare il server come servizio di Windows NT/XP, eseguire
  ...\mysql\bin\mysql-nt -install
- Per avviare il servizio, eseguire il comando ***NET START mysql***
- Per arrestare il servizio, eseguire il comando
  NET STOP mysql
- Per rimuovere il server eseguire
  ...\mysql\bin\mysql-nt -remove


C:\WINDOWS\System32\cmd.exe
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>cd mysql
C:\mysql>cd bin
C:\mysql\bin>mysqld-nt -install
Service successfully installed.
C:\mysql\bin>NET START mysql
Servizio MySQL in fase di avvio .
Avvio del servizio MySQL riuscito.
C:\mysql\bin>

Il client MySQL

- Per eseguire il client MySQL digitare
  - ***mysql***
- Per avere l'elenco dei database ai quali ha accesso il server MySQL, eseguire
  - ***SHOW DATABASES***;
- Per creare un nuovo database, eseguire
  - ***CREATE DATABASE nome_nuovo_database***;


C:\WINDOWS\System32\cmd.exe - mysql
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>cd mysql
C:\mysql>cd bin
C:\mysql\bin>mysqld-nt -install
Service successfully installed.
C:\mysql\bin>NET START mysql
Servizio MySQL in fase di avvio .
Avvio del servizio MySQL riuscito.
C:\mysql\bin>mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 4.0.18-nt
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>


C:\WINDOWS\System32\cmd.exe - mysql
mysql>
mysql>
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql |
| test |
+----------+
2 rows in set (0.00 sec)
mysql> CREATE DATABASE datilibri;
Query OK, 1 row affected (0.44 sec)
mysql> SHOW DATABASES;
+-----------+
| Database |
+-----------+
| datilibri |
| mysql |
| test |
+-----------+
3 rows in set (0.00 sec)
mysql>

Utilizzare una delle basi di dati disponibili

- Appena si avvia il server non è possibile modificare alcun database perchè il programma non sa con quale database deve lavorare
  - Istruzione ***USE***

    USE nome_nuovo_database;
- Una volta selezionato il database da utilizzare è possibile eseguire tutte le operazioni di accesso e modifica ai dati
  - L'istruzione ***SHOW TABLES*** mostra le tabelle della base di dati

Definizione di tabelle

- Si può aggiungere una tabella ad una base di dati con l'istruzione

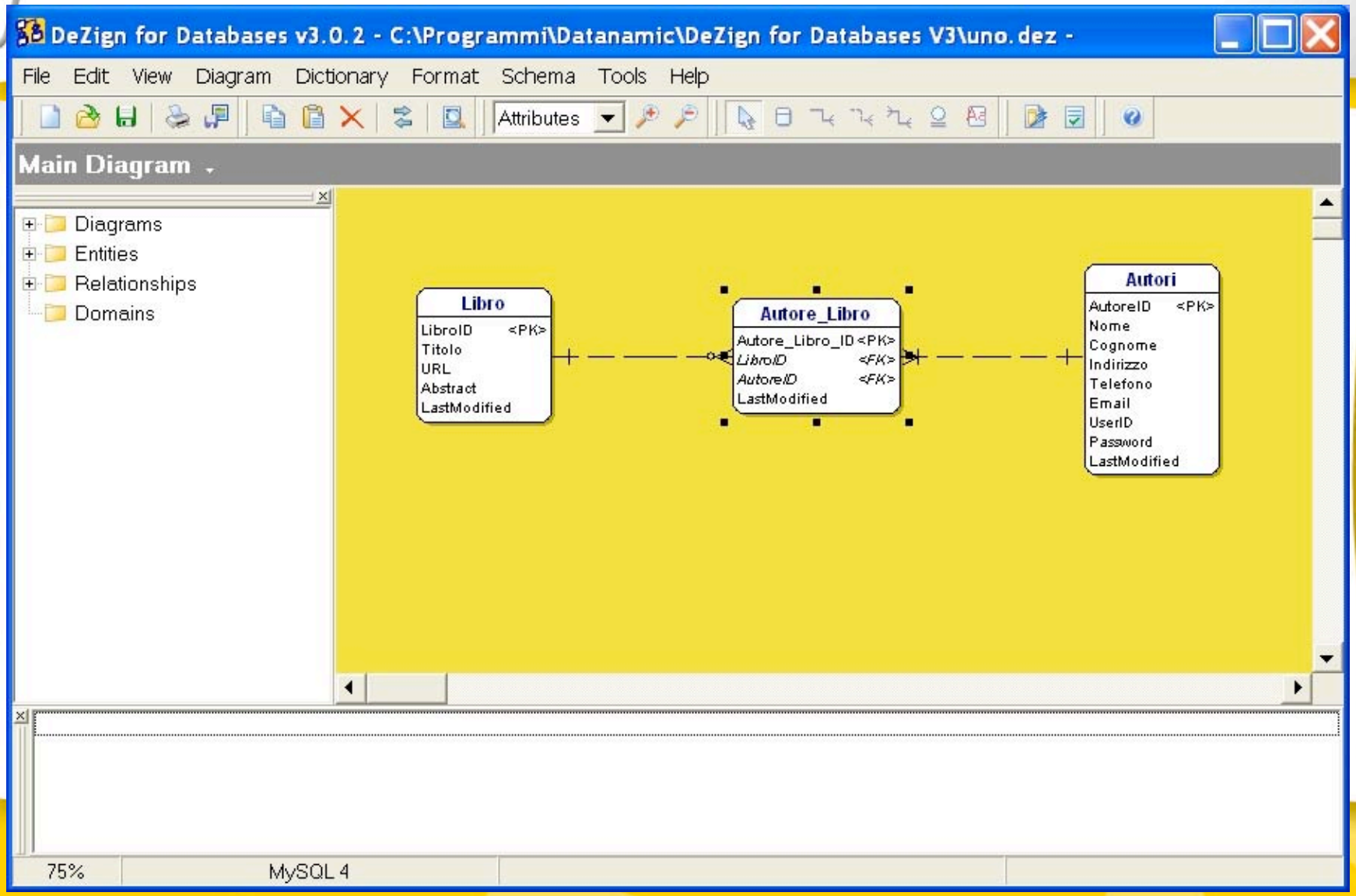
```
CREATE TABLE nome_tabella (
campo1 tipo1,
campo2 tipo2,
..., ...
);
```

Informazioni sulle tabelle

- Per ottenere una descrizione dettagliata dei campi previsti in una tabella utilizzare l'istruzione

```
DESCRIBE nome_tabella;
```

Progetto di una piccola base di dati



Creazione della base di dati da script

- Scrivere tutti i comandi SQL per la creazione delle tabelle e per l'inserimento dei dati dal prompt del client è un procedimento lungo e soggetto a numerosi errori -> si ricorre spesso a script.

- Per poter utilizzare un file di script file_script.sql, si esegue il comando

```
source file_script.sql
```

File di script creaDB.sql

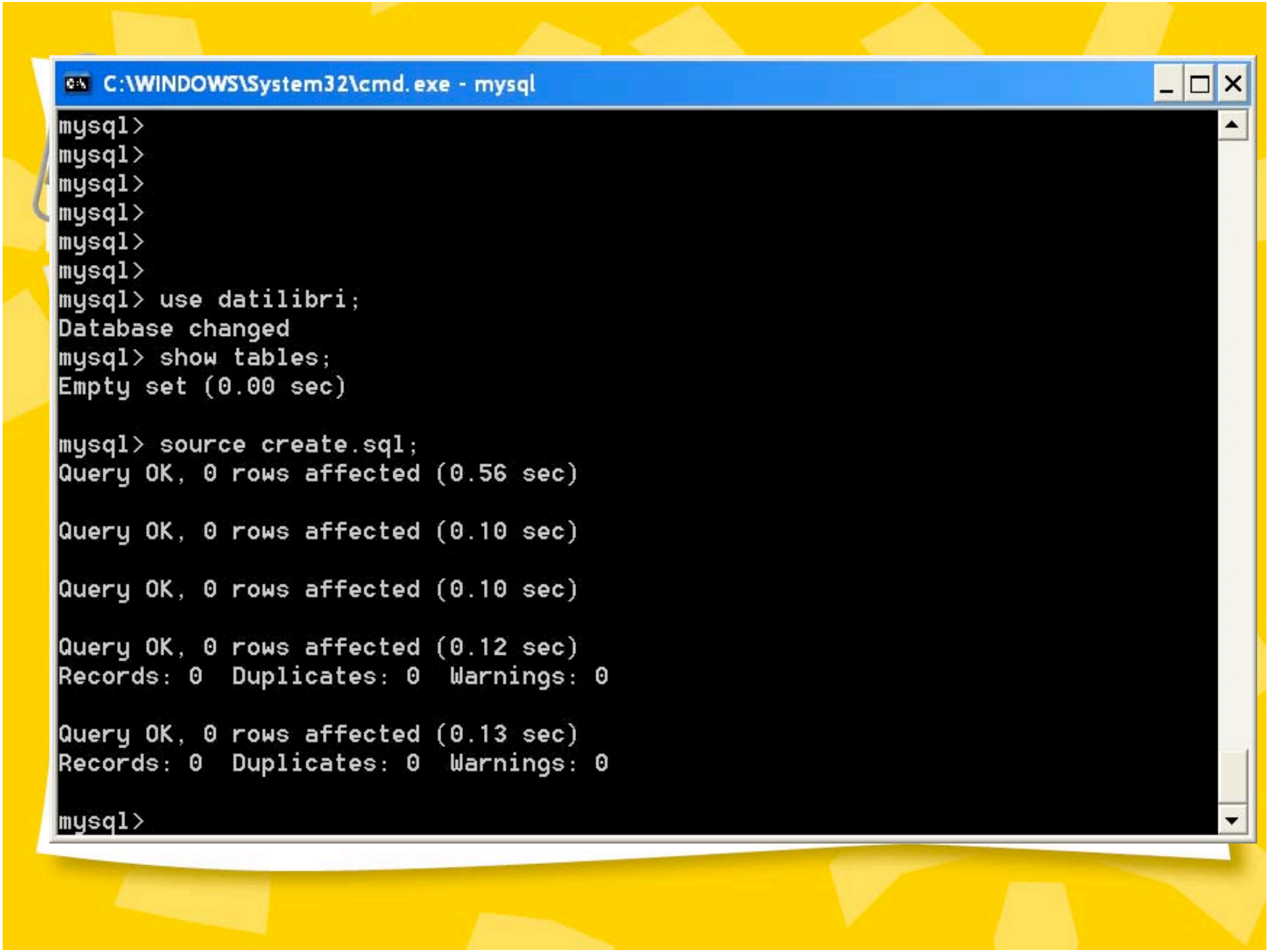
```
CREATE DATABASE esempio;
USE esempio;

CREATE TABLE Libro (
    LibroID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    Titolo VARCHAR(40) NOT NULL,
    URL VARCHAR(40),
    Abstract TEXT,
    LastModified TIMESTAMP,
    CONSTRAINT LibroID PRIMARY KEY (LibroID),
    UNIQUE KEY IDX_Libro_1(LibroID)
);

. . .
```

File di script creaDB.sql (cont.)

```
. . .
CREATE TABLE Autori (
    AutoreID INTEGER NOT NULL AUTO_INCREMENT,
    Nome VARCHAR(40) NOT NULL,
    Cognome VARCHAR(40) NOT NULL,
    Indirizzo VARCHAR(40),
    Telefono VARCHAR(40),
    Email VARCHAR(40),
    UserID VARCHAR(40),
    Password VARCHAR(40),
    LastModified TIMESTAMP,
    PRIMARY KEY (AutoreID),
    UNIQUE KEY IDX_Autori_1(AutoreID)
);

. . .
```

File di script creaDB.sql (cont.)

```
. . .
CREATE TABLE Autore_Libro (
    Autore_Libro_ID INTEGER NOT NULL AUTO_INCREMENT,
    LibroID INTEGER NOT NULL,
    AutoreID INTEGER NOT NULL,
    LastModified TIMESTAMP,
    PRIMARY KEY (Autore_Libro_ID),
    KEY IDX_Autore_Libro_1(LibroID),
    KEY IDX_Autore_Libro_2(AutoreID)
);

ALTER TABLE Autore_Libro
    ADD FOREIGN KEY (LibroID) REFERENCES Libro (LibroID);

ALTER TABLE Autore_Libro
    ADD FOREIGN KEY (AutoreID) REFERENCES Autori (AutoreID);
```


C:\WINDOWS\System32\cmd.exe - mysql
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> use datilibri;
Database changed
mysql> show tables;
Empty set (0.00 sec)
mysql> source create.sql;
Query OK, 0 rows affected (0.56 sec)
Query OK, 0 rows affected (0.10 sec)
Query OK, 0 rows affected (0.10 sec)
Query OK, 0 rows affected (0.12 sec)
Records: 0 Duplicates: 0 Warnings: 0
Query OK, 0 rows affected (0.13 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql>

Precisazione sui privilegi di accesso

CREATE DATABASE esempio;

grant all privileges on esempio.* to 'admin'@'localhost' identified by 'admin';
flush privileges;

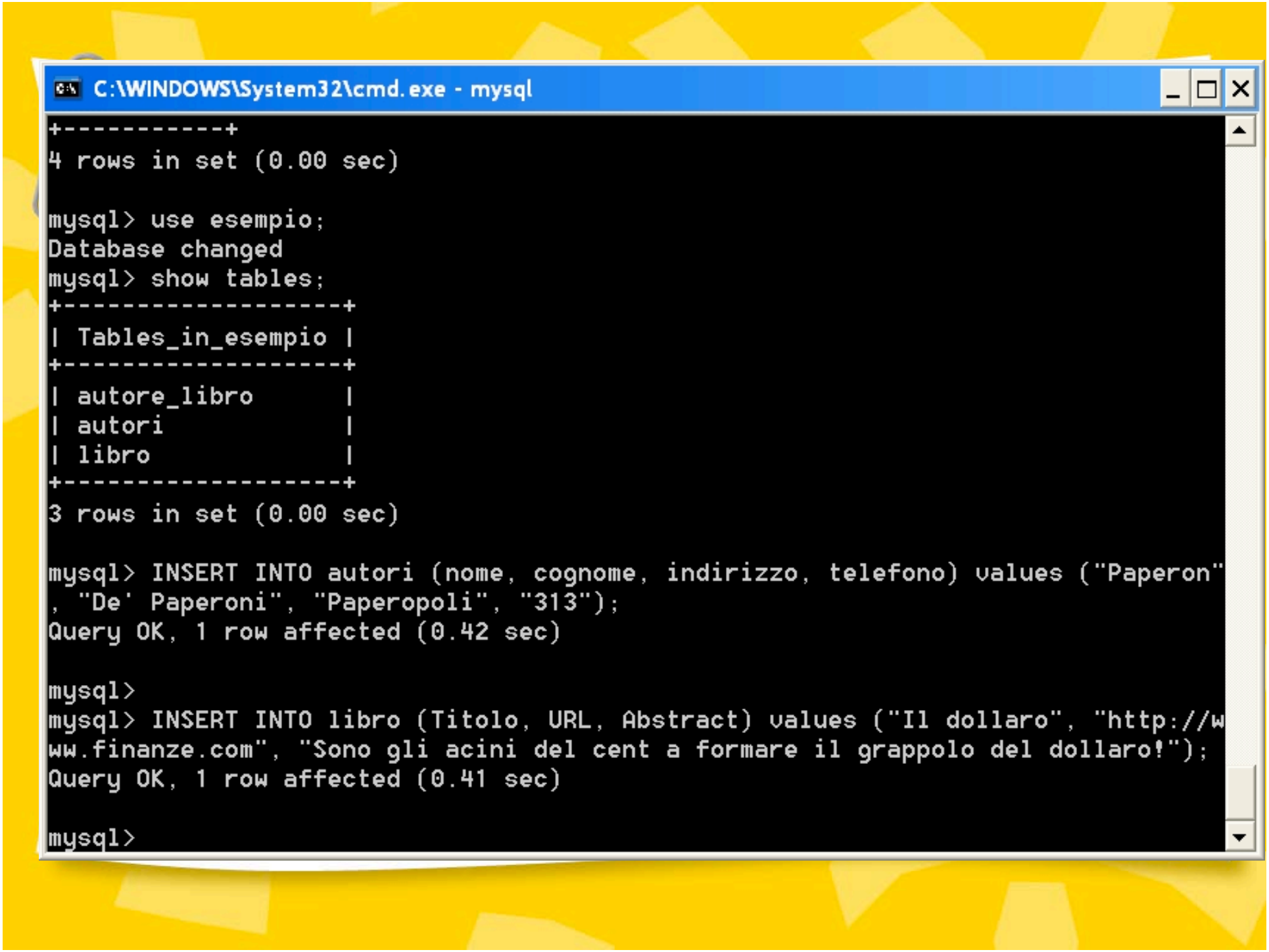
USE esempio;

Per accedere al db si usano user = ‘admin’ e password = ‘admin’

Inserimento di un record in una tabella

- Per inserire un nuovo record in una tabella si usa il comando insert

```
INSERT INTO nome_tabella
(campo_1, campo_2, ..., campo_n)
VALUES ("valore1", "valore2", ... "valoren");
```


C:\WINDOWS\System32\cmd.exe - mysql
+-----------+
4 rows in set (0.00 sec)
mysql> use esempio;
Database changed
mysql> show tables;
+-------------------+
| Tables_in_esempio |
+-------------------+
| autore_libro |
| autori |
| libro |
+-------------------+
3 rows in set (0.00 sec)
mysql> INSERT INTO autori (nome, cognome, indirizzo, telefono) values ("Paperon"
, "De' Paperoni", "Paperopoli", "313");
Query OK, 1 row affected (0.42 sec)
mysql>
mysql> INSERT INTO libro (Titolo, URL, Abstract) values ("Il dollaro", "http://w
ww.finanze.com", "Sono gli acini del cent a formare il grappolo del dollaro!");
Query OK, 1 row affected (0.41 sec)
mysql>

Modifica di record di una tabella

- Modificare dei dati in una tabella
  - **UPDATE** *tableName*
    SET *fieldName1 = value1, … , fieldNameN = valueN*
    WHERE *criteria*
  - Es: **UPDATE** *autori*
    SET *cognome = 'Jones'*
    WHERE *cognome = 'Smith'* **AND** *nome = 'Sue'*

Rimozione di record da una tabella

- Per rimuovere dei dati da una tabella si usa l'istruzione ***DELETE***
- ```
  DELETE FROM tableName WHERE criteria
  ```
- ```
  DELETE FROM autori WHERE cognome =
  'Jones' AND nome = 'Sue'
  ```

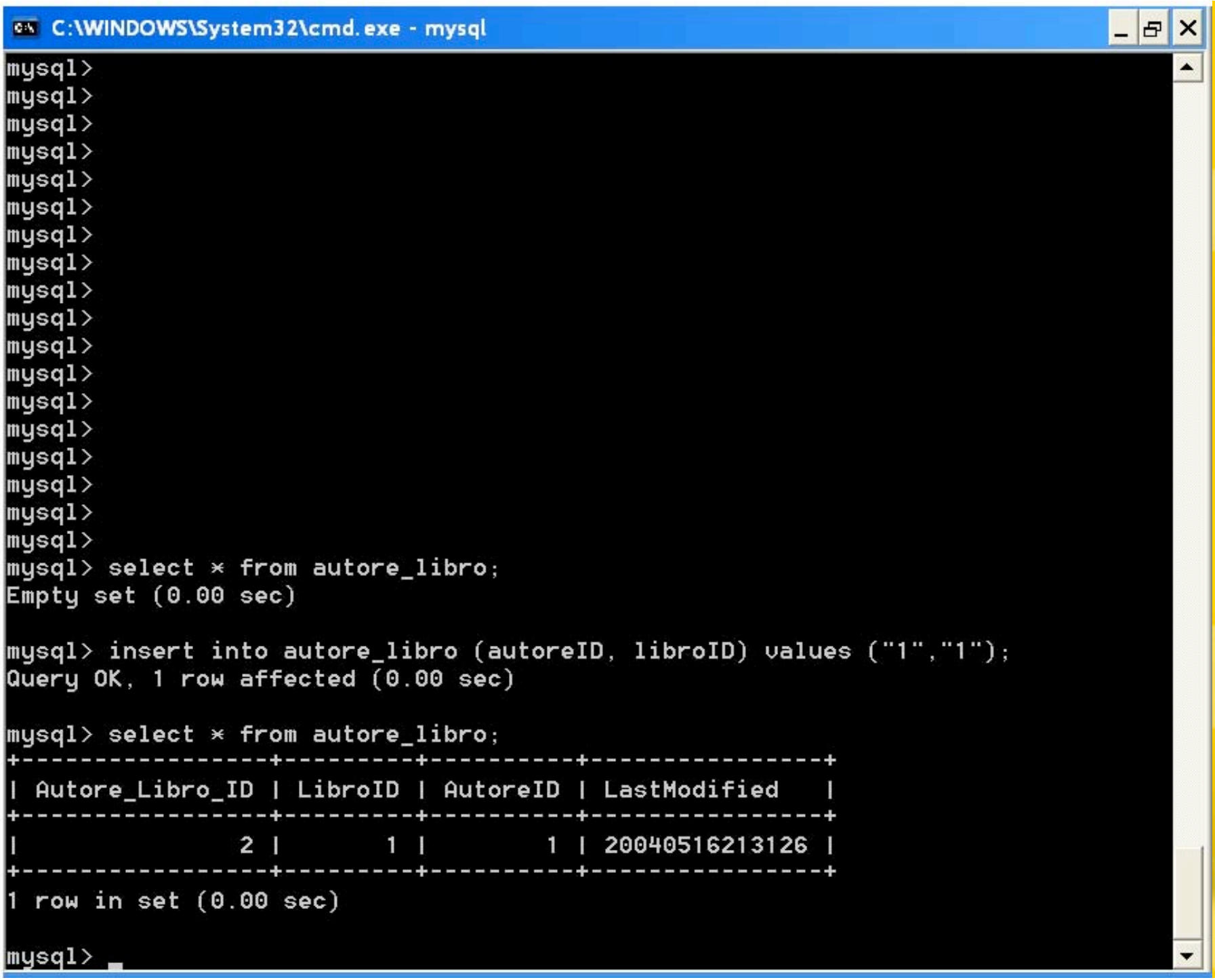
Consultazione di una tabella

- Istruzione

  SELECT ... FROM ... WHERE ...

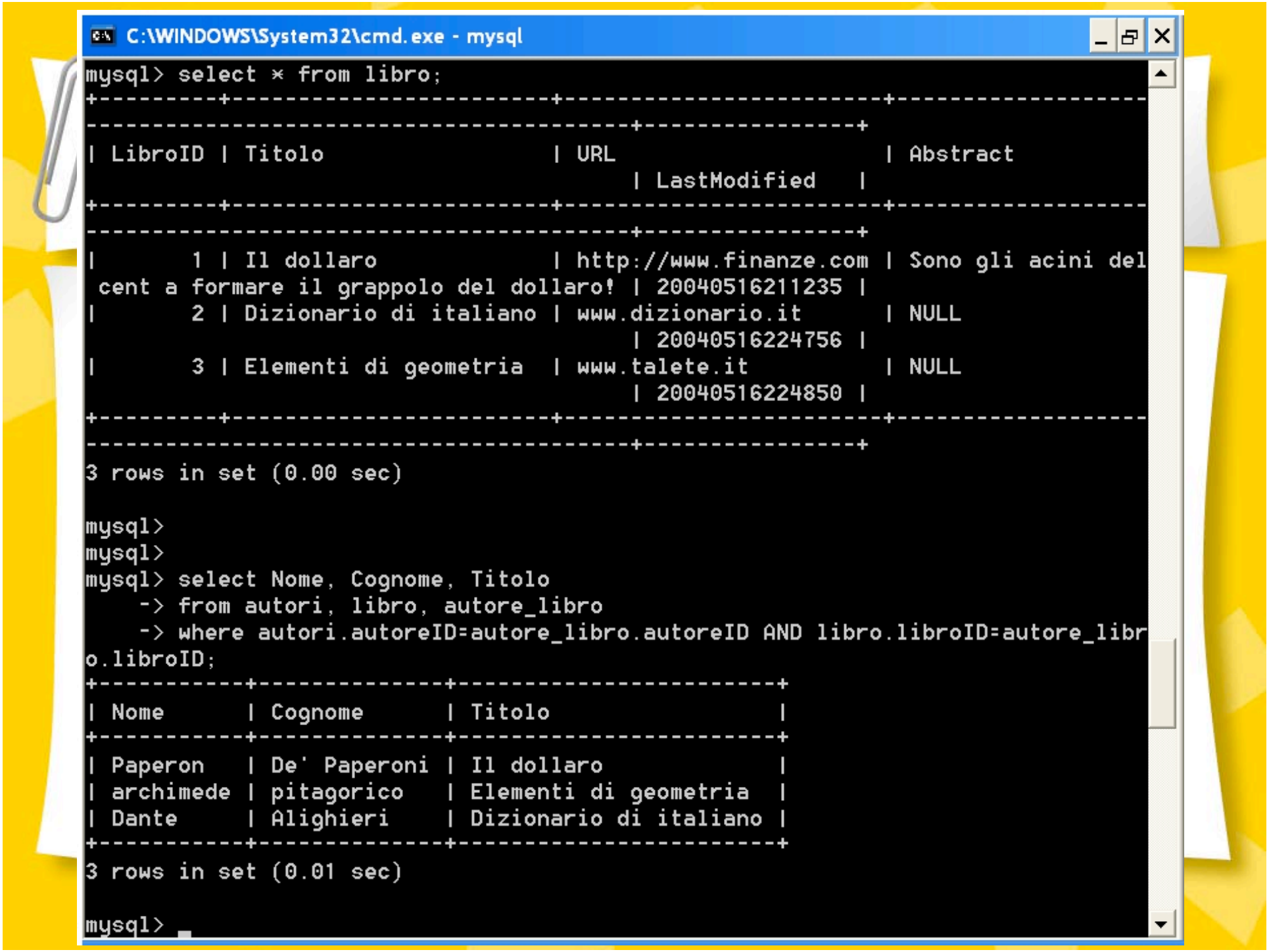
  Es:

```
SELECT campo1, campo2
FROM nome_tabella
WHERE campo3=valore_x;
```


C:\WINDOWS\System32\cmd.exe - mysql
mysql> select * from autore_libro;
Empty set (0.00 sec)
mysql> insert into autore_libro (autoreID, libroID) values ("1","1");
Query OK, 1 row affected (0.00 sec)
mysql> select * from autore_libro;
| Autore_Libro_ID | LibroID | AutoreID | LastModified |
| 2 | 1 | 1 | 20040516213126 |
1 row in set (0.00 sec)
mysql>

Fusione dei dati di più tabelle (join)

- Per combinare i dati di più tabelle
  - ```
    SELECT fieldName1, fieldName2, …
        FROM table1, table2
        WHERE table1.fieldName = table2.fieldName
    ```
  - ```
    SELECT Nome, Cognome, LibroID
        FROM autori, autore_libro
        WHERE autori.autoreID =
              autore_libro.autoreID
        ORDER BY Cognome, Nome
    ```


C:\WINDOWS\System32\cmd.exe - mysql
mysql> select * from libro;
+---------+-----------------------+-----------------------+------------------
--+----------------+
| LibroID | Titolo | URL | Abstract
                                        | LastModified |
+---------+-----------------------+-----------------------+------------------
--+----------------+
| 1 | Il dollaro | http://www.finanze.com | Sono gli acini del
 cent a formare il grappolo del dollaro! | 20040516211235 |
| 2 | Dizionario di italiano | www.dizionario.it | NULL
                                        | 20040516224756 |
| 3 | Elementi di geometria | www.talete.it | NULL
                                        | 20040516224850 |
+---------+-----------------------+-----------------------+------------------
--+----------------+
3 rows in set (0.00 sec)
mysql>
mysql>
mysql> select Nome, Cognome, Titolo
    -> from autori, libro, autore_libro
    -> where autori.autoreID=autore_libro.autoreID AND libro.libroID=autore_libr
o.libroID;
+-----------+-------------+-----------------------+
| Nome | Cognome | Titolo |
+-----------+-------------+-----------------------+
Paperon	De' Paperoni	Il dollaro
archimede	pitagorico	Elementi di geometria
Dante	Alighieri	Dizionario di italiano
+-----------+-------------+-----------------------+
3 rows in set (0.01 sec)
mysql> _

Ordinamento dei risultati di una query

- Clausola ***ORDER BY*** opzionale
  - ***SELECT fieldName1, fieldName2, …***
    FROM tableName ORDER BY field ASC
  - ***SELECT fieldName1, fieldName2, …***
    FROM tableName ORDER BY field DESC
- L'ordinamento può anche essere richiesto su più campi
  - ***ORDER BY field1 sortingOrder, field2***
    sortingOrder, …

Utilizzare MySQL da servlet/JSP

- Creare una connessione ad un database
- Interrogare/modificare il database
- Mostrare il risultato di un'interrogazione

…. iniziamo con degli esempi pratici che utilizzeremo per comprendere i dettagli teorici…

Utilizzare MySQL da servlet/JSP

- Si utilizzano delle API (driver) che forniscono i metodi per l'apertura di una connessione con il database, il recupero e l'aggiornamento dei dati
- Uno dei driver maggiormente consigliati per prestazioni e affidabilità è il

  MySQL Connector/J che trovate alla pagina del corso
  - Il file ***mysql-connector-java-3.0.11-stable-bin.jar*** va incluso in una delle directory del ***CLASSPATH*** (esempio ***/context-root/WEB-INF/lib***, oppure in ***<CATALINA_HOME>/common/lib***)

Esercizio: servlet che visualizza i dati del database "esempio" (1/4)

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
    public class MySQLServlet extends javax.servlet.http.HttpServlet {
        public void doGet(HttpServletRequest request,
                          HttpServletResponse response)
                throws IOException, ServletException {
            response.setContentType("text/html");
            PrintWriter out=response.getWriter();
            out.println("<html><head><title> Test di MySQL
                                        </title></head><body>");
            out.println("<H1> Questa pagina è prodotta da <br>"+
                        "una servlet che esegue una query al DB "</H1><br>");
            out.println("<table border=\"1\" cellpadding=\"5\""
                + " cellspacing=\"0\" width=\"400\">");

. . .
```

Esercizio: servlet che visualizza i dati del database "esempio" (2/4)

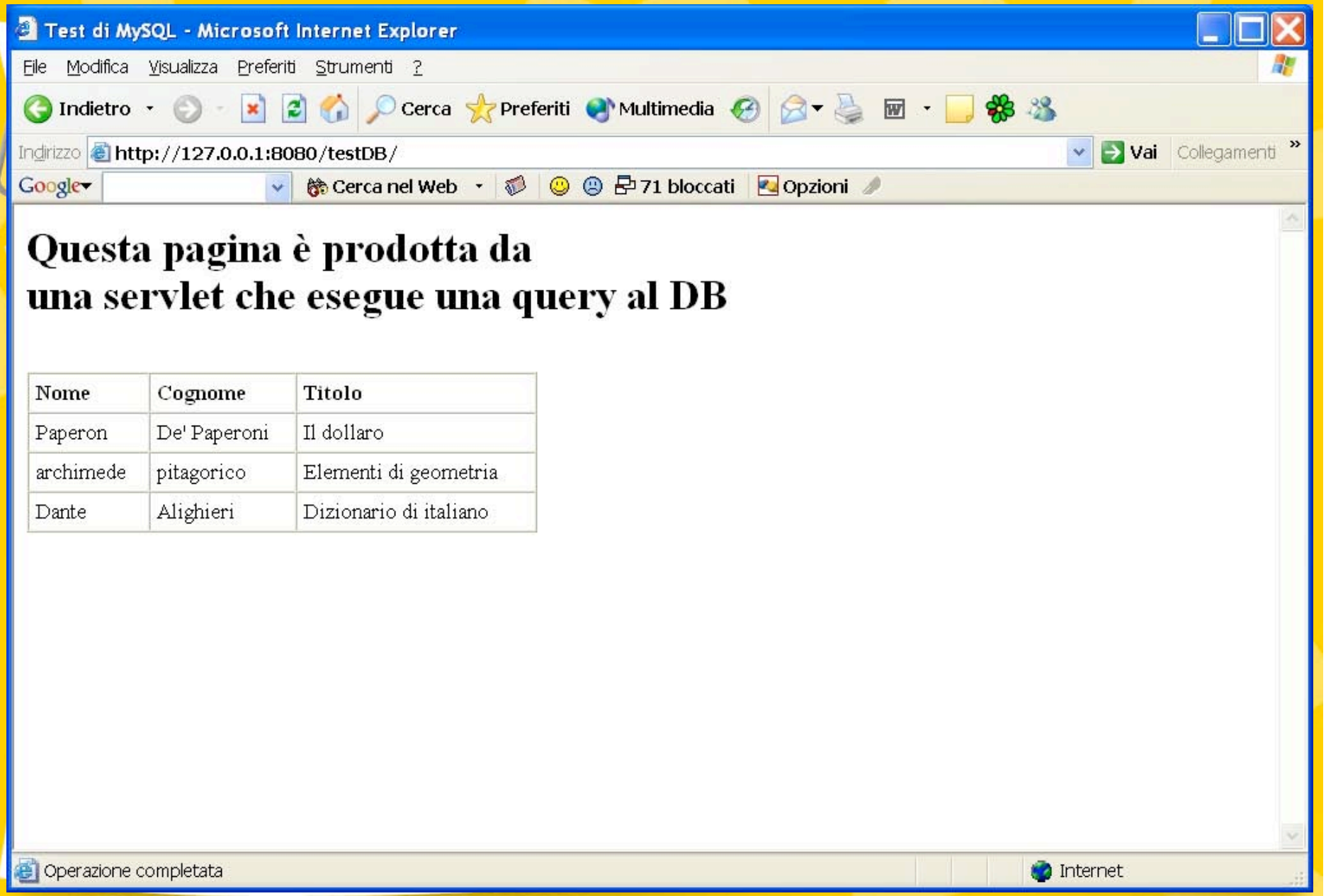
```
String connectionURL="jdbc:mysql://localhost:3306/esempio";
      Connection connection = null;
      Statement statement=null;
      ResultSet resultSet=null;
      try {
              //caricamento dinamico della classe e
              //registrazione del driver presso il driver manager
          Class.forName( "com.mysql.jdbc.Driver");
              // connect to database
          connection = DriverManager.getConnection(connectionURL);
              // create Statement to query database
          statement = connection.createStatement();

              // query database
          String query_autori_libri;
          query_autori_libri="SELECT Nome, Cognome, Titolo ";
          query_autori_libri+="from autori, libro, autore_libro ";
          query_autori_libri+="where autori.autoreID=autore_libro.autoreID ";
          query_autori_libri+="AND libro.libroID=autore_libro.libroID;";
          resultSet =
             statement.executeQuery(query_autori_libri);
```

Esercizio: servlet che visualizza i dati del database "esempio" (3/4)

```
// process query results
            StringBuffer results = new StringBuffer();
            ResultSetMetaData metaData = resultSet.getMetaData();
            results.append("<tr>");
            for ( int i = 1; i <= 3; i++ ) {
               results.append( "<td><b>" + metaData.getColumnName( i )
               + "</b></td>" );
            }
            results.append( "</tr>" );

            while ( resultSet.next() ) {
                results.append("<tr>");
                for ( int i = 1; i <= 3; i++ ) {
                    results.append("<td>"+ resultSet.getObject( i )
                    + "</td>" );
                  }
                  results.append( "</tr>" );
                }
           out.println(results.toString());
           } //fine del try
```

Esercizio: servlet che visualizza i dati del database "esempio" (4/4)

```
        // detect problems interacting with the database
             catch ( SQLException e ) {
                 System.err.println("SQL Problem: "+e.getMessage());
                 System.err.println("SQL State: "+e.getSQLState());
                 System.err.println("Error: "+e.getErrorCode());
                 System.exit( 1 );
             }

             // detect problems loading database driver
             catch ( ClassNotFoundException e ) {
                System.err.println("Non trovo il driver"+ e.getMessage());
             }

             finally {  //eseguita sempre a meno che non venga chiamato exit()
                   try  {
                        if (connection!=null) connection.close();
                   }
                   catch (SQLException e) {
                        System.err.println(e.getMessage());
                   }
             }
             out.println("</table></body></html>");
        }
}
```

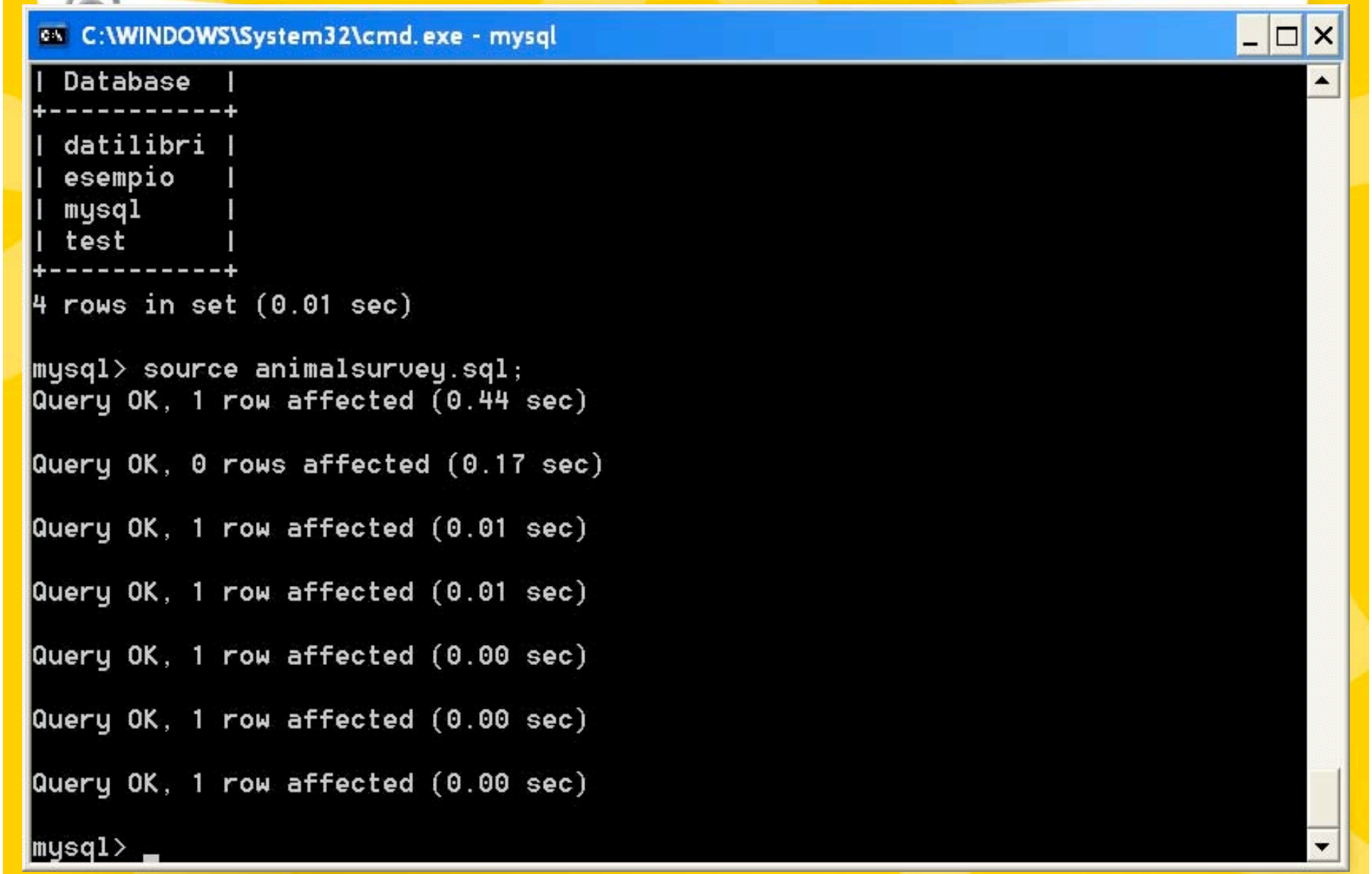

Test di MySQL - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro
Cerca
Preferiti
Multimedia
Indirizzo
http://127.0.0.1:8080/testDB/
Vai
Collegamenti
Google
Cerca nel Web
71 bloccati
Opzioni
Questa pagina è prodotta da
una servlet che esegue una query al DB
Nome
Cognome
Titolo
Paperon
De' Paperoni
Il dollaro
archimede
pitagorico
Elementi di geometria
Dante
Alighieri
Dizionario di italiano
Operazione completata
Internet

Esercizio: sondaggio online

- In una base di dati si vogliono rappresentare 4 categorie di animali. All'utente dell'applicazione viene sottoposto un form in cui selezionare il proprio animale preferito. Al termine della votazione viene visualizzata una pagina con le percentuali di voti riscosse da ciascun animale.

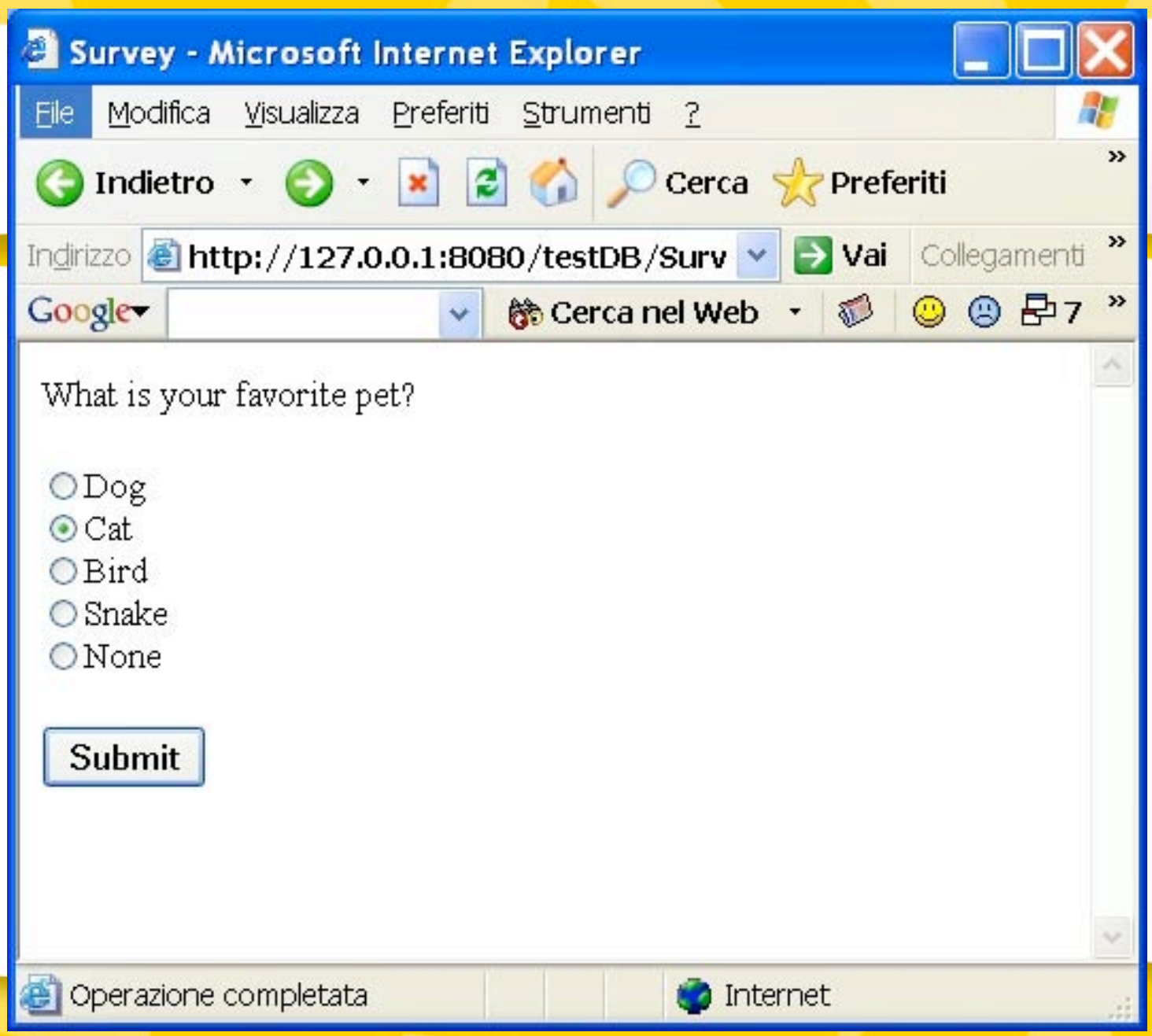
Creazione della base di dati: animalsurvey.sql

```
CREATE DATABASE animalsurvey;
USE animalsurvey;
create table surveyresults (
          id int NOT NULL ,
          surveyoption varchar (20) NOT NULL ,
          votes int NOT NULL ,
          constraint surveyresults_id primary key (id)
);

insert into surveyresults (id,surveyoption,votes) values (1, 'Dog', 0);
insert into surveyresults (id,surveyoption,votes) values (2, 'Cat', 0);
insert into surveyresults (id,surveyoption,votes) values (3, 'Bird', 0);
insert into surveyresults (id,surveyoption,votes) values (4, 'Snake', 0);
insert into surveyresults (id,surveyoption,votes) values (5, 'None', 0);
```


C:\WINDOWS\System32\cmd.exe - mysql
| Database |
+----------+
| datilibri |
| esempio |
| mysql |
| test |
+----------+
4 rows in set (0.01 sec)
mysql> source animalsurvey.sql;
Query OK, 1 row affected (0.44 sec)
Query OK, 0 rows affected (0.17 sec)
Query OK, 1 row affected (0.01 sec)
Query OK, 1 row affected (0.01 sec)
Query OK, 1 row affected (0.00 sec)
Query OK, 1 row affected (0.00 sec)
Query OK, 1 row affected (0.00 sec)
mysql>

Form di realizzazione del sondaggio: survey.html

```
<html><head><title>Survey</title></head>
<body>
<form method = "post" action = "/testDB/Survey">
   <p>What is your favorite pet?</p>
   <p>
      <input type = "radio" name = "animal"
         value = "1" />Dog<br />
      <input type = "radio" name = "animal"
         value = "2" />Cat<br />
      <input type = "radio" name = "animal"
         value = "3" />Bird<br />
      <input type = "radio" name = "animal"
         value = "4" />Snake<br />
      <input type = "radio" name = "animal"
         value = "5" checked = "checked" />None
   </p>
   <p><input type = "submit" value = "Submit" /></p>
</form>
</body> </html>
```


Survey - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro
Cerca
Preferiti
Indirizzo
http://127.0.0.1:8080/testDB/Surv
Vai
Collegamenti
Google
Cerca nel Web
What is your favorite pet?
Dog
Cat
Bird
Snake
None
Submit
Operazione completata
Internet

Realizzazione di una query attraverso l'interfaccia PreparedStatement

- Interfaccia ***PreparedStatements***
  - Più flessibile
  - Più efficiente
- Definire una query con un'oggetto che implementa l'interfaccia ***PreparedStatement***
  - ```
    PreparedStatement stringa_da_configurare =
    connection.prepareStatement(
            "SELECT field1, field2, field3" +
            "FROM tableA, tableB" +
            "WHERE tableA.field_X = tableB.field_Y "  +
            "AND fieldJ = ? AND fieldK = ?" );
    ```
    ```

Realizzazione di una query attraverso l'interfaccia PreparedStatement (cont.)

- Configurare i parametri in ***PreparedStatement***
  - `stringa_da_configurare.setString( 1, "Rossi" );`
  - `stringa_da_configurare.setString( 2, "Paolo" );`
- ***PreparedStatement*** con i parametri configurati
  - ```
    SELECT field1, field2, field3
       FROM tableA, tableB
       WHERE tableA.field_X = tableB.field_Y AND
                fieldJ = 'Rossi' AND fieldK = 'Paolo'
    ```
    ```

Servlet che gestisce la base di dati del sondaggio: SurveyServlet.java (1/5)

```
import java.io.*;
import java.text.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SurveyServlet extends HttpServlet {
   private Connection connection;
   private PreparedStatement updateVotes, totalVotes, results;

   // set up database connection and prepare SQL statements
   public void init( ServletConfig config )
      throws ServletException
   {
      // attempt database connection and create PreparedStatements
      try {
         Class.forName("com.mysql.jdbc.Driver");
         connection = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/animalsurvey" );
         // PreparedStatement to add one to vote total for a
         // specific animal
         updateVotes =
            connection.prepareStatement(
               "UPDATE surveyresults SET votes = votes + 1 " +
               "WHERE id = ?");
```

Servlet che gestisce la base di dati del sondaggio: SurveyServlet.java (2/5)

```
// PreparedStatement to sum the votes
         totalVotes =
            connection.prepareStatement(
               "SELECT sum( votes ) FROM surveyresults"
            );

         // PreparedStatement to obtain surveyoption table's data
         results =
            connection.prepareStatement(
               "SELECT surveyoption, votes, id " +
               "FROM surveyresults ORDER BY id"
            );
      }

      // for any exception throw an UnavailableException to
      // indicate that the servlet is not currently available
      catch ( Exception exception ) {
         exception.printStackTrace();
         throw new UnavailableException( exception.getMessage() );
      }

   }  // end of init method
```

Servlet che gestisce la base di dati del sondaggio: SurveyServlet.java (3/5)

```
// process survey response
   protected void doPost( HttpServletRequest request,
      HttpServletResponse response )
         throws ServletException, IOException
   {
      // set up response to client
      response.setContentType( "text/html" );
      PrintWriter out = response.getWriter();
      DecimalFormat twoDigits = new DecimalFormat( "0.00" );

      // start XHTML document

      out.println(
         "<html><head>" );

      // read current survey response
      int value =
         Integer.parseInt( request.getParameter( "animal" ) );
```

Servlet che gestisce la base di dati del sondaggio: SurveyServlet.java (4/5)

```
// attempt to process a vote and display current results
      try {
         // update total for current survey response
         updateVotes.setInt( 1, value );
         updateVotes.executeUpdate();
         // get total of all survey responses
         ResultSet totalRS = totalVotes.executeQuery();
         totalRS.next();
         int total = totalRS.getInt( 1 );
         // get results
         ResultSet resultsRS = results.executeQuery();
         out.println( "<title>Thank you!</title></head><body> " );
         out.println( "<p>Thank you for participating." );
         out.println( "<br />Results:</p>" );
         // process results
         int votes;
         while ( resultsRS.next() ) {
            out.print( resultsRS.getString( 1 )+": " );
            votes = resultsRS.getInt( 2 );
            out.print( twoDigits.format(( double ) votes / total * 100 ) );
            out.print( "%  responses: "+ votes );
         }
         resultsRS.close();
```

Configura il primo parametro di **PreparedStatement updateVotes** dove avevamo "…where id=?"

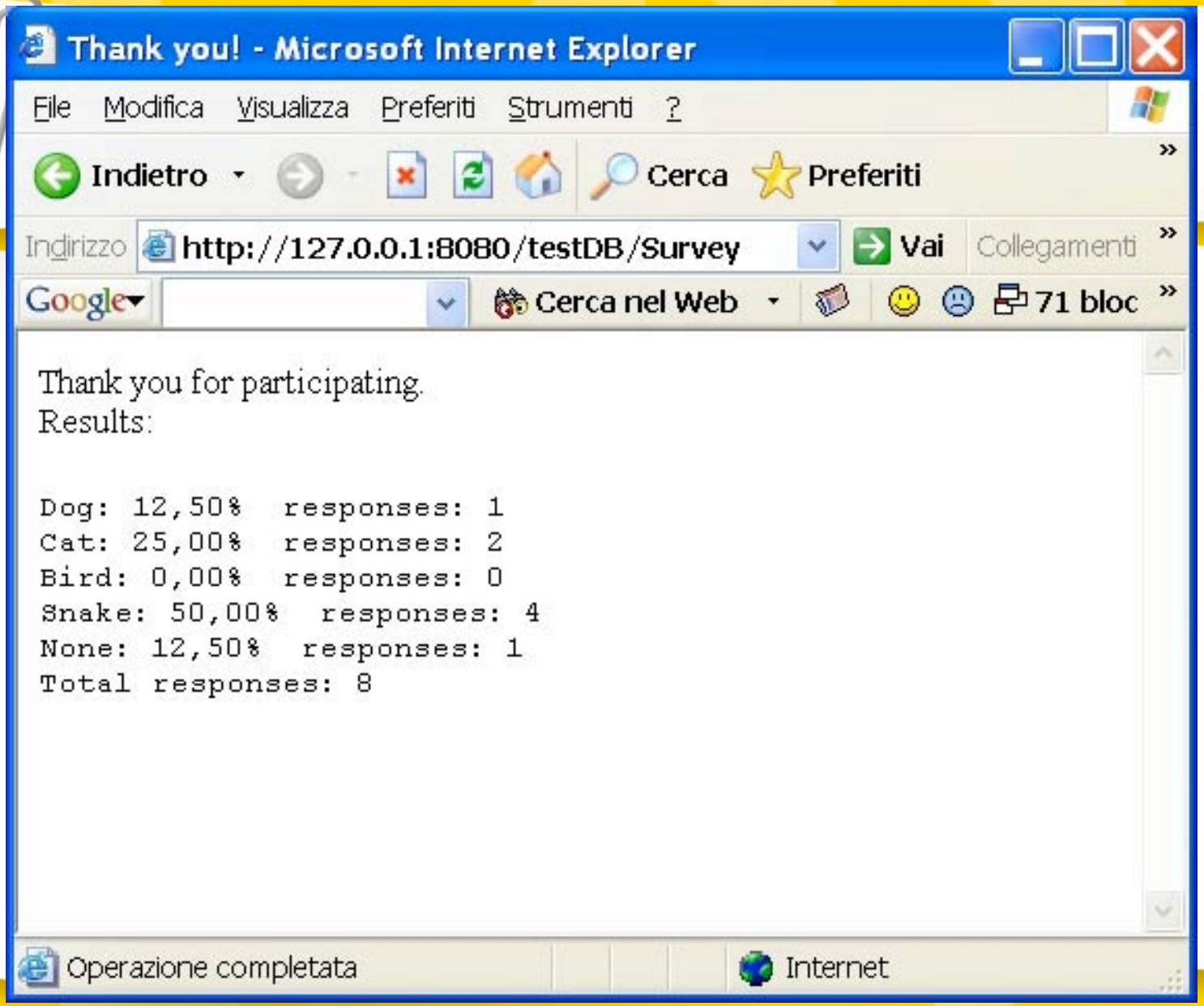
Esegue la query **totalVotes** per calcolare il numero di voti ricevuti.

Esegue la **PreparedStatement results** e elabora **ResultSet** per creare il risultato del sondaggio.

Servlet che gestisce la base di dati del sondaggio: SurveyServlet.java (5/5)

```
    out.print( "Total responses: " + total + "</body></html>");
    out.close();
      }
      // if database exception occurs, return error page
      catch ( SQLException sqlException ) {
         sqlException.printStackTrace();
         out.println( "<title>Error</title></head>"" );
         out.println( "<body><p>Database error occurred. " );
         out.println( "Try again later.</p></body></html>" );
         out.close();
      }
   }  // end of doPost method
   // close SQL statements and database when servlet terminates
   public void destroy()
   {
      // attempt to close statements and database connection
      try {
         updateVotes.close();
         totalVotes.close();
         results.close();
         connection.close();
      }
      // handle database exceptions by returning error to client
      catch( SQLException sqlException ) {sqlException.printStackTrace();}
   }  // end of destroy method
}
```

Il metodo destroy elimina ciascuna **PreparedStatement** e chiude tutte le connessioni.


Thank you! - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro
Cerca
Preferiti
Indirizzo
http://127.0.0.1:8080/testDB/Survey
Vai
Collegamenti
Google
Cerca nel Web
71 bloc
Thank you for participating.
Results:
Dog: 12,50% responses: 1
Cat: 25,00% responses: 2
Bird: 0,00% responses: 0
Snake: 50,00% responses: 4
None: 12,50% responses: 1
Total responses: 8
Operazione completata
Internet

JDBC (Java Database Connectivity)

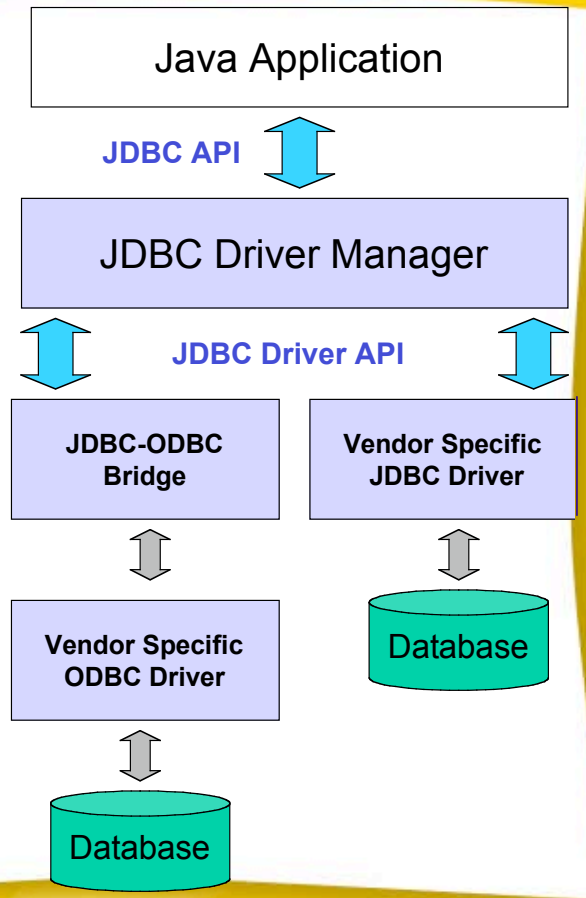
- JDBC è costituito da una libreria per l'accesso a basi di dati di tipo relazionale
  - Le API JDBC standardizzano
    - Le modalità di connessione con una base dati
    - Modalità di interrogazione della base dati
    - Modalità di creazione di query parametrizzate
    - Le strutture dati con cui il risultato di una query può essere trattato
      - Come determinare il numero di colonne di una tabella
      - Utilizzo di metadati, etc.
  - Le API JDBC non standarizzano la sintassi SQL
  - Le classi che costituiscono il supporto JDBC si trovano nel package java.sql

Risorse online

- Sito della Sun su JDBC
  - http://java.sun.com/products/jdbc/
- Tutorial su JDBC
  - http://java.sun.com/docs/books/tutorial/jdbc/
- Elenco dei driver JDBC disponibili
  - http://industry.java.sun.com/products/jdbc/drivers/
- Informazioni sulle API di java.sql
  - http://java.sun.com/j2se/1.4/docs/api/java/sql/package-summary.html

Driver JDBC

- JDBC consiste di:
  - API JDBC API, puramente basate su JAVA
  - Un manager di driver JDBC, che comunica con i driver specifici dell'applicativo in uso per realizzare il DB.



Driver JDBC

- L'applicazione inoltra all'API JDBC le chiamate per l'apertura di una connessione con il database, recupera e aggiorna i dati, esegue i comandi previsti e chiude la connessione.
- I driver del database si connettono ad un database specifico oppure ad un protocollo intermedio (come ODBC o altro middleware)

Driver JDBC

- I database riconoscono il linguaggio SQL in modi diversi.
- I database forniscono protocolli diversi per la connessione al loro motore.
- Compete al driver provvedere a tutti i problemi di conversione tra i comandi JDBC e il motore del database.

n.b.: Le API JDBC e il driver manager fanno parte del JDK, mentre i driver sono reperibili presso il fornitore della base di dati

Sette passi per connettersi ad una base di dati sfruttando il supporto JDBC

1. Caricare il driver
2. Definire l'URL per la connessione con la base dati
3. Instaurare la connessione
4. Creare un oggetto Statement che rappresenta la query da inoltrare
5. Eseguire la query
6. Elaborare il risultato
7. Chiudere la connessione

I sette passi nel dettaglio

1. Caricare il driver

```
try {
   Class.forName("com.mysql.jdbc.Driver");
//oppure
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  } catch { ClassNotFoundException cnfe) {
   System.out.println("Error loading driver: " cnfe);
  }
```

= *caricamento dinamico della classe e registrazione del driver presso il driver manager*

Note sul caricamento del driver

- Il caricamento della classe del driver avviene in modo dinamico, a tempo di esecuzione.
- Il codice che inizializza la classe del driver provvede a registrare il driver presso il driver manager
- Il driver può essere caricato anche nel seguente modo:

```
DriverManager.registerDriver("classe.del.driver");
```

- Cambiando il driver è possibile usare un db completamente diverso senza modificare il codice

I sette passi nel dettaglio

2. Definire il percorso (URL) della connessione

```
String host = "dbhost.yourcompany.com";
String dbName = "someName";
int port = 1234;
String accessURL = "jdbc:mySQL:" + host +
                   ":" + port + ":" + dbName;
```

I sette passi nel dettaglio

3. Instaurare una connessione

```
String username = "Paperone";
   String password = "numero1";
   Connection connection =
        DriverManager.getConnection(accessURL,
                                    username,
                                    password);
```

Informazioni sulla base di dati

- Una volta instaurata la connessione è possibile richiedere informazioni generiche sulla base dati:

```
DatabaseMetaData dbMetaData =
     connection.getMetaData();
String productName =
           dbMetaData.getDatabaseProductName();
String productVersion =
           dbMetaData.getDatabaseProductVersion();
System.out.println("Database: " +  productName);
System.out.println("Version: " +  productVersion);
```

I sette passi nel dettaglio

4. Creare un oggetto Statement che rappresenta la query da inoltrare

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

I sette passi nel dettaglio

5. Eseguire una query

```
Statement statement = connection.createStatement();
String query =
    "SELECT col1, col2, col3 FROM sometable";
ResultSet resultSet =
    statement.executeQuery(query);
```

- Notare che executeQuery() restituisce un ResultSet e non altera la base di dati
- Per modificare la base dati, usare l'istruzione ***executeUpdate,*** fornendo comandi SQL come ***UPDATE, INSERT, o DELETE***. Viene restituito un intero corrispondente al numero di righe modificate
- Usare ***setQueryTimeout*** per specificare il massimo intervallo di tempo che si è disposti ad aspettare prima per ottenere la risposta

I sette passi nel dettaglio

6. Elaborazione della risposta

```
while(resultSet.next()) {
    System.out.println(resultSet.getString(1)
  +
    " " + resultSet.getString(2) +
    " " + resultSet.getString(3));
 }
```

- La prima colonna ha indice 1, non 0
- **`ResultSet`** fornisce numerosi metodi **`getXxx`** con argomento un indice di colonna *o il nome della colonna*

I sette passi nel dettaglio

7. Chiudere la connessione

```
connection.close();
```

   - Dal momento che le operazioni di apertura di una connessione sono molto costose, posporre questa operazione se sono necessarie ulteriori interazioni con la base dati

Utilizzo di MetaData

- Dati che riguardano l'applicazione
  - ***`connection.getMetaData().getDatabaseProductName()`***
  - ***`connection.getMetaData().getDatabaseProductVersion()`***
- Dati che riguardano le tabelle della base dati
  - ***`resultSet.getMetaData().getColumnCount()`***
    - Se usate il risultato per elencare gli elementi di un record, ricordate che l'indice deve partire da 1 e non da 0
  - ***`resultSet.getMetaData().getColumnName(indice_colonna)`***

Utilizzo di ***Statement***

- Attraverso l'oggetto ***Statement***, si possono inviare comandi SQL al database.
- Esistono alcuni tipi di statement:
  - ***Statement***
    - Per eseguire un comando SQL semplice
  - ***PreparedStatement***
    - Per eseguire un comando SQL precompilato e parametrizzato

Metodi di Statement

- ***executeQuery***
  - Esegue la query e fornisce il risultato in una tabella (ResultSet)
  - La tabella del risultato può essere vuota ma mai ***null***

```
ResultSet results =
    statement.executeQuery("SELECT a, b FROM table");
```

- ***executeUpdate***
  - Usato per eseguire modifiche attraverso i comandi SQL ***INSERT, UPDATE, e DELETE***
  - Il risultato è il numero di righe che sono state modificate
  - Supporta anche comandi Data Definition Language (DDL) come ***CREATE TABLE, DROP TABLE e ALTER TABLE***

```
int rows =
        statement.executeUpdate("DELETE FROM EMPLOYEES" +
                                "WHERE STATUS=0");
```

Metodi di ***Statement***

- ***getMaxRows/setMaxRows***
  - Determina il massimo numero di righe che possono essere contenute in un ***ResultSet***
  - A meno che non venga esplicitamente dichiarato, il numero di righe della risposta è illimitato (return value: 0)
- ***getQueryTimeout/setQueryTimeout***
  - Specifica il periodo di tempo che il driver attenderà per ottenere una risposta prima di lanciare una eccezione ***SQLException***

PreparedStatements

- Idea
  - Se dovete eseguire comandi simili ripetutamente, potete farlo efficientemente usando una statement precompilata e parametrizzata
  - Si crea una statement in una forma standard che viene compilata prima di essere utilizzata
  - Ogni volta che volete utilizzare una statement precompilata, dovete rimpiazzare i parametri marcati con “?” utilizzando i metodi ***setXxx***

PreparedStatements

- ***PreparedStatement*** estende ***Statement***.
  I metodi
  - executeQuery()
  - executeUpdate()

  vengono ereditati ma non prevedono nessun parametro.

PreparedStatement, Esempio

```
Connection connection =
  DriverManager.getConnection(url, user, password);
PreparedStatement statement =
  connection.prepareStatement("UPDATE employees "+
                              "SET salary = ? " +
                              "WHERE id = ?");
int[] newSalaries = getSalaries();
int[] employeeIDs = getIDs();
for(int i=0; i<employeeIDs.length; i++) {
  statement.setInt(1, newSalaries[i]);
  statement.setInt(2, employeeIDs[i]);
  statement.executeUpdate();
}
```

Metodi di PreparedStatement

- ***setXxx***
  - Configura i parametri indicati con (?)
- ***clearParameters***
  - Annulla tutti i parametri configurati

Realizzazione di una query attraverso l'interfaccia ***PreparedStatement***

- Interfaccia ***PreparedStatements***
  - Più flessibile
  - Più efficiente
- Definire una query con un oggetto che implementa l'interfaccia ***PreparedStatement***
  - ```
    PreparedStatement authorBooks =
    connection.prepareStatement(
            "SELECT cognome, nome, titolo " +
            "FROM autori, libro, autore_libro" +
            "WHERE autori.autoreID = autore_libro.autoreID "  +
            "AND libro.libroID = autore_libro.libroID AND " +
            "cognome = ? AND nome = ?" );
    ```
    ```

Realizzazione di una query attraverso l'interfaccia PreparedStatement (cont.)

- Configurare i parametri in **PreparedStatement**
  - `authorBooks.setString( 1, "Rossi" );`
  - `authorBooks.setString( 2, "Paolo" );`
- **PreparedStatement** con i parametri configurati
  - ```
    SELECT cognome, nome, titolo
      FROM autori, libro, autore_libro
      WHERE autori.autoreID = autore_libro.autoreID AND
            libro.libroID = autore_libro.libroID AND
            cognome = 'Rossi' AND nome = 'Paolo'
    ```
    ```

Transazioni

- Per impostazione predefinita, dopo l'esecuzione di un comando SQL, il commit dei cambiamenti avviene immediatamente
- Impostando al valore off la configurazione di auto-commit si possono raggruppare due o più statement in una transazione
  - ***`connection.setAutoCommit(false)`***
- La chiamata di ***`commit`*** modifica permanentemente i record registrando i cambiamenti conseguenti alla transazione
- La chiamata di ***`rollback`*** serve per il ripristino in caso di errori

Transazioni: esempio

```
Connection connection =
  DriverManager.getConnection(url, username, passwd);
connection.setAutoCommit(false);
try {
  statement.executeUpdate(...);
  statement.executeUpdate(...);

  connection.commit();
} catch (Exception e) {
  try {
    connection.rollback();
  } catch (SQLException sqle) {
     // report problem
  }
} finally {
  try {
   connection.close();
  } catch (SQLException sqle) { }
}
```

Metodi di Connection per la gestione delle transazioni

- ***`getAutoCommit/setAutoCommit`***
  - Imposta o legge il valore della modalità auto-commit
- ***`commit`***
  - Forza tutti i cambiamenti richiesti dall'ultima chiamata di commit
- ***`rollback`***
  - Elimina i cambiamenti richiesti a partire dalla precedente chiamata di commit