Esercizio 1

Si consideri un database relazionale contenente due tabelle con schema:

Fornitore(FID: int, Fnome: string, indirizzo:string)
Prodotti(PID:int, Pnome: string, tipo:string)
Catalogo(PID: int, FID: int, costo: int)

Si richiede di:

1. Scrivere un'applicazione JDBC per:
    a) Determinare i fornitori di prodotti di prezzo minore di MYCOSTO, dove MYCOSTO è fornito in input dall'utente, e stampare i nomi dei fornitori a video;
    b) cancellare dal catalogo le informazioni relative a prodotti che costano più di MYCOSTO;
    c) aggiornare infine i prezzi a catalogo aggiungendo 100 a tutti i costi.

    Si vuole attribuire un comportamente transazionale all'applicazione.
    Si utilizzino le seguenti informazioni:
    - Driver: "oracle.jdbc.OracleDriver"
    - URL:jdbc:Oracle:@MyDB
    - Login: Scott
    - Password: tiger

2. Progettare un'applicazione Web per determinare i fornitori di prodotti di prezzo maggiore a MYCOSTO, dove MYCOSTO è fornito in input dall'utente, tramite una FORM HTML, elaborata con il metodo POST. L'output è rappresentato da un pagina HTML contenente i fornitori determinati. La stessa pagina deve permettere di inserire nuovamente un valore per MYCOSTO e rieseguire la query. A questo proposito, si ipotizzi di utilizzare la tecnologia Servlet e si presenti:
    a) L'interfaccia della classe da definire.
    b) Per ogni metodo, la descrizione della sua funzionalità. A questo proposito, si associ ad ogni metodo l'elenco degli item, tra quelli sottoelencati, che rappresentano le operazioni principali eseguite dal metodo (non è detto che tutti gli item debbano essere utilizzati), nell'ordine con cui devono essere effettuate:
        1) Connessione al database
        2) Costruzione stringa query parametrica
        3) Costruzione stringa query non parametrica
        4) Specifica parametri query
        5) Preparazione query
        6) Esecuzione query
        7) Elaborazione e preparazione pagina html dinamica
        8) Disconnessione database
        9) Invio pagina
        10) Lettura argomenti FORM

        Se le operazioni precedenti devono essere effettuate solo sotto opportune condizioni, indicare le condizioni in pseudo-codice o linguaggio naturale.
        Motivare le scelte effettuate per ogni metodo.

Soluzione

1. L'applicazione deve eseguire una query dinamica, con un parametro fornito in input. Tuttavia, l'applicazione esegue la query solo una volta, quindi non è il caso di preparare lo statement. Poiché viene richiesto un comportamento transazionale, sarà necessario porre a false l'autocommit e gestire la transazione a livello JDBC. Il codice risulta essere il seguente:

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


public class FornitoriJDBC
{

  public static void main (String args [])
{
        int MYCOSTO;
        Class.forName ("oracle.jdbc.OracleDriver");
        String con_URL = "jdbc:oracle:@MyDB";

        Connection  con =DriverManager.getConnection(con_URL, "scott", "tiger");
        con.setAutoCommit(false);

        Reader r=new BufferedReader(new InputStreamReader(System.in));
        StreamTokenizer s=new StreamTokenizer(r);
        while (s.nextToken()!=StreamTokenizer.TT_NUMBER);
        MYCOSTO=s.nval;
        String query = “select Fnome from Fornitore, Catalogo “ +
                    “where Fornitore.FID = Catalogo.FID and costo < “ + MYCOSTO;
        String upd1= “ delete from Catalog where costo >” + MYCOSTO;
        String upd2= “update catalog set costo = costo + 100”;
        Statement stmt = con.createStatement ();
        ResultSet res = stmt.executeQuery (query);
        while (res.next())
                System.out.println(res.getString(2));
        stmt.executeUpdate (upd1);
        stmt.executeUpdate (upd2);
        con.commit();
        con.close();
} }
```

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


public class FornitoriJDBC
{
  public static void main (String args [])
{
        try{
        int MYCOSTO;
        Class.forName ("oracle.jdbc.OracleDriver");
        String con_URL = "jdbc:oracle:@MyDB";
        Connection  con =DriverManager.getConnection(con_URL, "scott", "tiger");
        con.setAutoCommit(false);
        Reader r=new BufferedReader(new InputStreamReader(System.in));
        StreamTokenizer s=new StreamTokenizer(r);
```

```
      while (s.nextToken()!=StreamTokenizer.TT_NUMBER);
      MYCOSTO=s.nval;
      String query = "select Fnome from Fornitore, Catalogo " +
                    "where Fornitore.FID = Catalogo.FID and costo < " + MYCOSTO;
      String upd1= " delete from Catalog where costp >" + MYCOSTO;
      String upd2= "update catalog set costo = costo + 100";
      Statement stmt = con.createStatement ();
      ResultSet res = stmt.executeQuery (query);
      while (res.next())
                        System.out.println(res.getString(2));
      stmt.executeUpdate (upd1);
      stmt.executeUpdate (upd2);
      con.commit();
      con.close();
      }   catch (SQLException e) {
            if( e!=null){
                 System.out.println(" Message: " + e.getMessage());
                 con.rollback();
                        }}
}}
```

2. Poiché MYCOSTO viene fornita dall'utente e l'utente potrebbe volere eseguire la query più volte, con valori di MYCOSTO diversi, in questo caso potrebbe essere conveniente preparare lo statement. Per garantire che la preparazione venga effettuata una sola volta, è opportuno preparare lo statement nel metodo init(). Quindi la classe conterrà i seguenti metodi con le seguenti funzionalità

   public void init(): 1, 2, 5

   public void doPost(HTTPRequest, HTTPResponse): 10, 4, 6, 7, 9

   public void destroy(): 8

-

Esercizio 2

Si consideri una relazione Fornitore avente il seguente schema:

Fornitore(Id: String, Nome: String, Prodotti_XML: CLOB)

Dove Prodotti_XML rappresenta un documento XML che contiene informazioni circa i prodotti forniti da un certo fornitore.
Si supponga che i documenti XML soddisfino il seguente DTD:

```
<!ELEMENT PRODOTTI(PRODOTTO+)>
<!ELEMENT PRODOTTO(NOME,DESCRIZIONE)>
<!ELEMENT DESCRIZIONE(#PCDATA)>
<!ELEMENT NOME(#PCDATA)>
<!ATTLIST PRODOTTO
    Codice  ID          #REQUIRED
    Tipo    CDATA       #REQUIRED>
```

Scrivere un'interrogazione in SQL per Oracle esteso con Intermedia Context per:
1) determinare tutti i fornitori di giocattoli
2) determinare tutti i fornitori del giocattolo di nome "Pallino"

Soluzione

1)
```
SELECT Id
FROM Fornitore
WHERE CONTAINS(Prodotti_XML, "giocattoli WITHIN PRODOTTO@TIPO")
```

2)
```
SELECT Id
FROM Fornitore
WHERE CONTAINS(Prodotti_XML, "Pallino WITHIN NOME")
```