Università degli Studi di Padova

Department of Information Engineering
University of Padova

Basi di Dati, A.A. 2014/2015
Corso di Laurea Magistrale in Ingegneria Informatica

Homework 4 - Progettazione Fisica
Data di consegna: 15 giugno 2015

Cognome	Nome	Numero di matricola
Vallerini	Luca	1110975

Schema fisico

```
1  -- Creazione del database e impostazione della codifica di default a UTF8
2  -- CREATE DATABASE fumetteria ENCODING 'UTF8';
3
4  -- Creazione del tipo di dati Media per il campo Media della tabella Serie
5  CREATE TYPE Media AS ENUM ('Anime', 'Drama', 'Light Novel', 'Live Action', 'Manga', 'Romanzo');
6
7  -- Creazione del tipo di dati StatoSerie per il campo StatoSerie della tabella Serie
8  CREATE TYPE StatoSerie AS ENUM ('Completa', 'In corso', 'Interrotta', 'Sospesa', 'Futura');
9
10 -- Creazione del tipo di dati Periodicita' per il campo Period della tabella Edizione
11 CREATE TYPE Periodicita AS ENUM ('Settimanale', 'Quindicinale', 'Mensile', 'Bimestrale', 'Trimestrale', 'Quadrimestrale', 'Semestrale'
       , 'Annuale', 'Irregolare', 'Uscita singola');
12
13 -- Creazione del tipo di dati Supporto per il campo Supporto della tabella Edizione
14 CREATE TYPE Supporto AS ENUM ('DVD', 'BD', 'DVD + BD');
15
16 -- Creazione del tipo di dati StatoOrdine per il campo StatoOrd della tabella Ordine
17 CREATE TYPE StatoOrdine AS ENUM ('Nuovo', 'In lavorazione', 'Evaso', 'Annullato', 'Rifiutato');
18
19 -- Creazione del tipo di dati TipoOrdine per il campo TipoOrdine della tabella Ordine
20 CREATE TYPE TipoOrdine AS ENUM ('Acquisto', 'Abbonamento', 'Arretrato', 'Fornitura');
21
22 -- Creazione del tipo di dati StatoAccount per il campo StatoAcc della tabella Account
23 CREATE TYPE StatoAccount AS ENUM ('Attivo', 'Sospeso');
24
25 -- Creazione del tipo di dati TipoAccount per il campo TipoAcc della tabella Account
26 CREATE TYPE TipoAccount AS ENUM ('Tesserato', 'Venditore', 'Titolare');
27
28 CREATE TABLE Genere (
29   NomeGen VARCHAR(50) PRIMARY KEY
30 );
31
32 CREATE TABLE Serie (
33   CodSerie CHARACTER(32) PRIMARY KEY, -- hash md5 di titolo+autore+anno+media
34   Titolo VARCHAR(100) NOT NULL,
35   Autore VARCHAR(100) NOT NULL,
36   Anno INT CHECK (Anno > 1800) NOT NULL,
37   Media Media NOT NULL,
38   StatoSerie StatoSerie NOT NULL,
39   Trama TEXT,
40   NumEp INT CHECK (NumEp > 0),
41   Studio VARCHAR(100),
42   NumVolOrig INT CHECK (NumVolOrig > 0),
43   Regista VARCHAR(100),
44   Rivista VARCHAR(100)
45 );
46
47 CREATE TABLE GenereSerie (
48   Genere VARCHAR(50),
49   Serie CHARACTER(32),
50   PRIMARY KEY (Genere, Serie),
51   FOREIGN KEY (Genere) REFERENCES Genere(NomeGen),
52   FOREIGN KEY (Serie) REFERENCES Serie(CodSerie)
53 );
54
55 CREATE TABLE Tratto (
56   SerieOrig CHARACTER(32),
57   SerieDeriv CHARACTER(32),
58   PRIMARY KEY (SerieOrig, SerieDeriv),
59   FOREIGN KEY (SerieOrig) REFERENCES Serie(CodSerie),
60   FOREIGN KEY (SerieDeriv) REFERENCES Serie(CodSerie)
61 );
62
63 CREATE TABLE Prodotto (
64   CodProd CHARACTER(32) PRIMARY KEY, -- hash md5
65   DataIns TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP(0)
66 );
67
68 CREATE TABLE Editore (
69   PIVA CHARACTER(13) PRIMARY KEY,
70   Nome VARCHAR(100) NOT NULL,
71   Sito VARCHAR(100) NOT NULL
72 );
```

```
73
74  CREATE TABLE Edizione (
75     Prodotto CHARACTER(32) PRIMARY KEY,
76     Nome VARCHAR(50),
77     Period Periodicita NOT NULL,
78     Supporto Supporto,
79     Col BOOLEAN,
80     InizPubbl DATE,
81     StatoEd StatoSerie,
82     SovraCop BOOLEAN,
83     Extra TEXT,
84     NumVolTot INT CHECK (NumVolTot > 0),
85     Audio TEXT,
86     Video TEXT,
87     Sub TEXT,
88     NumPag INT CHECK (NumPag > 0),
89     Editore CHARACTER(13),
90     Serie CHARACTER(32),
91     FOREIGN KEY (Prodotto) REFERENCES Prodotto(CodProd),
92     FOREIGN KEY (Editore) REFERENCES Editore(PIVA),
93     FOREIGN KEY (Serie) REFERENCES Serie(CodSerie)
94  );
95
96  CREATE TABLE Volume (
97     Prodotto CHARACTER(32) PRIMARY KEY, --hash md5
98     DataPubbl DATE NOT NULL,
99     Ristampa BOOLEAN DEFAULT FALSE,
100    NumRist INT DEFAULT 0 CHECK (NumRist >= 0),
101    PrezzoFin NUMERIC(6,2) NOT NULL CHECK (PrezzoFin > 0),
102    NumVol INT NOT NULL CHECK (NumRist >= 0),
103    Cop BYTEA,
104    Disp INT CHECK (Disp >= 0),
105    ISBN CHARACTER(13) NOT NULL UNIQUE,
106    Edizione CHARACTER(32) NOT NULL,
107    FOREIGN KEY (Prodotto) REFERENCES Prodotto(CodProd),
108    FOREIGN KEY (Edizione) REFERENCES Edizione(Prodotto)
109 );
110
111 CREATE TABLE Distributore (
112    PIVA CHARACTER(13) PRIMARY KEY,
113    Nome VARCHAR(100) NOT NULL,
114    Sito VARCHAR(100) NOT NULL
115 );
116
117 CREATE TABLE Mail (
118    Mail VARCHAR(100) PRIMARY KEY
119 );
120
121 CREATE TABLE TelFax (
122    CodTelFax CHARACTER(32) PRIMARY KEY, --hash md5 num+fax
123    Num VARCHAR(20) NOT NULL,
124    Fax BOOLEAN DEFAULT FALSE NOT NULL
125 );
126
127 CREATE TABLE TelFaxEd (
128    TelFax CHARACTER(32),
129    Editore CHARACTER(13),
130    PRIMARY KEY (TelFax, Editore),
131    FOREIGN KEY (TelFax) REFERENCES TelFax(CodTelFax),
132    FOREIGN KEY (Editore) REFERENCES Editore(PIVA)
133 );
134
135 CREATE TABLE MailEd (
136    Mail VARCHAR(100),
137    Editore CHARACTER(13),
138    PRIMARY KEY (Mail, Editore),
139    FOREIGN KEY (Mail) REFERENCES   Mail(Mail),
140    FOREIGN KEY (Editore) REFERENCES Editore(PIVA)
141 );
142
143 CREATE TABLE TelFaxDistr (
144    TelFax CHARACTER(32),
145    Distributore CHARACTER(13),
146    PRIMARY KEY (TelFax, Distributore),
147    FOREIGN KEY (TelFax) REFERENCES   TelFax(CodTelFax),
148    FOREIGN KEY (Distributore) REFERENCES Distributore(PIVA)
149 );
150
151 CREATE TABLE MailDistr (
152    Mail VARCHAR(100),
153    Distributore CHARACTER(13),
154    PRIMARY KEY (Mail, Distributore),
155    FOREIGN KEY (Mail) REFERENCES   Mail(Mail),
156    FOREIGN KEY (Distributore) REFERENCES Distributore(PIVA)
157 );
158
159 CREATE TABLE Distribuzione (
160    Distributore CHARACTER(13),
161    Editore CHARACTER(13),
162    PRIMARY KEY (Distributore, Editore),
163    FOREIGN KEY (Distributore) REFERENCES Distributore(PIVA),
164    FOREIGN KEY (Editore) REFERENCES Editore(PIVA)
165 );
166
167 CREATE TABLE Indirizzo (
168    CodInd CHARACTER(32) PRIMARY KEY, --md5 via+civico+citta+provincia
169    Via VARCHAR(100) NOT NULL,
170    Civico VARCHAR(10) NOT NULL,
171    CAP CHARACTER(5),
172    Citta VARCHAR(50) NOT NULL,
173    Provincia CHARACTER(2) NOT NULL
174 );
175
```

```
176 CREATE TABLE Tesserato (
177    CodiceFiscale CHARACTER(16) PRIMARY KEY,
178    Nome VARCHAR(50) NOT NULL,
179    Cognome VARCHAR(50) NOT NULL,
180    Nascita DATE NOT NULL,
181    Mail VARCHAR(100) NOT NULL,
182    FOREIGN KEY (Mail) REFERENCES Mail(Mail)
183 );
184
185 CREATE TABLE TelTesserato (
186    Tesserato CHARACTER(16),
187    TelFax CHARACTER(32),
188    PRIMARY KEY (Tesserato, TelFax),
189    FOREIGN KEY (Tesserato) REFERENCES Tesserato(CodiceFiscale),
190    FOREIGN KEY (TelFax) REFERENCES TelFax(CodTelFax)
191 );
192
193 CREATE TABLE IndirizzoTesserato (
194    Tesserato CHARACTER(16),
195    Indirizzo CHARACTER(32),
196    PRIMARY KEY (Tesserato, Indirizzo),
197    FOREIGN KEY (Tesserato) REFERENCES Tesserato(CodiceFiscale),
198    FOREIGN KEY (Indirizzo) REFERENCES Indirizzo(CodInd)
199 );
200
201 CREATE SEQUENCE Ordine_CodOrd_seq;
202 CREATE TABLE Ordine (
203    CodOrd INT PRIMARY KEY DEFAULT nextval('Ordine_CodOrd_seq'),
204    StatoOrd StatoOrdine,
205    DataIns TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP(0),
206    PrezzoTot NUMERIC(6,2) NOT NULL CHECK (PrezzoTot > 0),
207    TipoOrd TipoOrdine NOT NULL,
208    Tesserato CHARACTER(16),
209    FOREIGN KEY (Tesserato) REFERENCES Tesserato(CodiceFiscale)
210 );
211
212 CREATE TABLE Rifornimento (
213    Ordine INT,
214    Distributore CHARACTER(13),
215    PRIMARY KEY (Ordine, Distributore),
216    FOREIGN KEY (Ordine) REFERENCES Ordine(CodOrd),
217    FOREIGN KEY (Distributore) REFERENCES Distributore(PIVA)
218 );
219
220 CREATE TABLE Merce (
221    Ordine INT,
222    Prodotto CHARACTER(32),
223    Quantita INT NOT NULL DEFAULT 1 CHECK(Quantita > 0),
224    PrezzoElem NUMERIC(6,2) NOT NULL CHECK(PrezzoElem > 0),
225    PRIMARY KEY (Ordine, Prodotto),
226    FOREIGN KEY (Ordine) REFERENCES Ordine(CodOrd),
227    FOREIGN KEY (Prodotto) REFERENCES Prodotto(CodProd)
228 );
229
230 CREATE SEQUENCE Account_CodAcc_seq;
231 CREATE TABLE Account (
232    CodAcc INT PRIMARY KEY DEFAULT nextval('Account_CodAcc_seq'),
233    Username VARCHAR(30) NOT NULL UNIQUE,
234    PW CHARACTER(32) NOT NULL, --md5
235    StatoAcc StatoAccount NOT NULL,
236    TipoAcc TipoAccount NOT NULL
237 );
238
239 CREATE TABLE Lavorazione (
240    Ordine INT,
241    Acc INT,
242    DataLav TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP(0),
243    PRIMARY KEY (Ordine, Acc),
244    FOREIGN KEY (Ordine) REFERENCES Ordine(CodOrd),
245    FOREIGN KEY (Acc) REFERENCES Account(CodAcc)
246 );
247
248 CREATE SEQUENCE Notifica_CodMail_seq;
249 CREATE TABLE Notifica (
250    CodMail INT PRIMARY KEY DEFAULT nextval('Notifica_CodMail_seq'),
251    Mittente INT NOT NULL,
252    Destinatario INT NOT NULL,
253    DataInvio TIMESTAMP,
254    DataGen TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP(0),
255    Oggetto VARCHAR(100) NOT NULL,
256    Testo TEXT NOT NULL,
257    FOREIGN KEY (Mittente) REFERENCES Account(CodAcc),
258    FOREIGN KEY (Destinatario) REFERENCES Account(CodAcc)
259 );
260
261 CREATE TABLE Incasellamento (
262    Prodotto CHARACTER(32),
263    AccStaff INT,
264    AccTesserato INT,
265    Quantita INT NOT NULL DEFAULT 1 CHECK (Quantita > 0),
266    DataIns TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP(0),
267    DataScad TIMESTAMP NOT NULL,
268    DataRim TIMESTAMP,
269    PRIMARY KEY (Prodotto, AccStaff, AccTesserato),
270    FOREIGN KEY (Prodotto) REFERENCES Prodotto(CodProd),
271    FOREIGN KEY (AccStaff) REFERENCES Account(CodAcc),
272    FOREIGN KEY (AccTesserato) REFERENCES Account(CodAcc)
273 );
274
275 CREATE SEQUENCE Tessera_CodTess_seq;
276 CREATE TABLE Tessera (
277    CodTess INT PRIMARY KEY DEFAULT nextval('Tessera_CodTess_seq'),
278    InizioVal DATE NOT NULL,
```

```
    FineVal DATE NOT NULL,
    DurataCasella INT NOT NULL DEFAULT 1 CHECK (DurataCasella > 0),
    PrezzoTess NUMERIC(6,2) NOT NULL DEFAULT 5 CHECK (PrezzoTess > 0),
    Sconto INT NOT NULL DEFAULT 10 CHECK (Sconto >= 0 AND Sconto <= 100)
);

CREATE TABLE Tesseramento (
    AccTesserante INT,
    AccTesserato INT,
    Tesserato CHARACTER(16),
    Tessera INT,
    DataTess DATE NOT NULL,
    PRIMARY KEY (AccTesserante, AccTesserato, Tesserato, Tessera),
    FOREIGN KEY (AccTesserante) REFERENCES Account(CodAcc),
    FOREIGN KEY (AccTesserato) REFERENCES Account(CodAcc),
    FOREIGN KEY (Tesserato) REFERENCES Tesserato(CodiceFiscale),
    FOREIGN KEY (Tessera) REFERENCES Tessera(CodTess)
);
```

Esempio di popolamento della base di dati

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Types;
import java.util.Calendar;
import java.util.Scanner;

/**
 * Universita' degli Studi di Padova
 * Dipartimento di Ingegneria dell'Informazione
 * Corso di laurea magistrale in Ingegneria Informatica
 *
 * Basi di dati - Anno Accademico 2014/2015
 *
 * Homework 4 - Popolamento della base di dati 'fumetteria'.
 *
 * @author Luca Vallerini - matr. 1110975
 * @version 0.3.20150615
 */

public class InsertDataFumetteria {

    // Informazioni essenziali per la connessione al database
    private static final String DRIVER = "org.postgresql.Driver";
    private static final String DATABASE = "jdbc:postgresql://localhost/database"; // DATABASE
    private static final String USER = "username"; // USERNAME
    private static final String PASSWORD = "password"; // PASSWORD

    // Input files
    private static final String INPUT_EDITORE = "data/editore.csv";
    private static final String INPUT_DISTRIBUTORE = "data/distributore.csv";
    private static final String INPUT_DISTRIBUZIONE = "data/distribuzione.csv";
    private static final String INPUT_SERIE = "data/serie.csv";
    private static final String INPUT_VOLUME = "data/volume.csv";
    private static final String INPUT_STAFF = "data/staff.csv";
    private static final String INPUT_TESSERATO = "data/tesserato.csv";
    private static final String INPUT_INCASELLAMENTO = "data/incasellamento.csv";
    private static final String INPUT_ORDINE = "data/ordine.csv";

    private static final String[] INPUT = {INPUT_EDITORE, INPUT_DISTRIBUTORE, INPUT_DISTRIBUZIONE, INPUT_SERIE,
      INPUT_VOLUME, INPUT_STAFF, INPUT_TESSERATO, INPUT_INCASELLAMENTO, INPUT_ORDINE};

    // SQL statement
    private static final String INSERT_INTO_EDITORE = "INSERT INTO editore (piva, nome, sito) VALUES (?, ?, ?)";
    private static final String INSERT_INTO_DISTRIBUTORE = "INSERT INTO distributore (piva, nome, sito) VALUES (?, ?, ?)";
    private static final String INSERT_INTO_MAIL = "INSERT INTO mail (mail) VALUES (?)";
    private static final String INSERT_INTO_MAILDISTR = "INSERT INTO maildistr (mail, distributore) VALUES (?, ?)";
    private static final String INSERT_INTO_MAILED = "INSERT INTO mailed (mail, editore) VALUES (?, ?)";
    private static final String INSERT_INTO_TELFAX = "INSERT INTO telfax (codtelfax, num, fax) VALUES (md5(?), ?, ?) RETURNING codtelfax
        ";
    private static final String INSERT_INTO_TELFAXDISTR = "INSERT INTO telfaxdistr (telfax, distributore) VALUES (?, ?)";
    private static final String INSERT_INTO_TELFAXED = "INSERT INTO telfaxed (telfax, editore) VALUES (?, ?)";
    private static final String INSERT_INTO_DISTRIBUZIONE = "INSERT INTO distribuzione (distributore, editore) VALUES (?, ?)";

    private static final String INSERT_INTO_SERIE = "INSERT INTO serie (codserie, titolo, autore, anno, media, statoserie, trama, " +
        "numep, studio, numvolorig, regista, rivista) VALUES (md5(?), ?, ?, ?, ?::media, ?::statoserie, ?, ?, ?, ?, ?, ?) RETURNING
            codserie";
    private static final String INSERT_INTO_GENERE = "INSERT INTO genere (nomegen) SELECT ? WHERE NOT EXISTS (SELECT nomegen FROM genere
        WHERE nomegen = ?)";
    private static final String INSERT_INTO_GENERESERIE = "INSERT INTO genereserie (genere, serie) VALUES (?, ?)";
    private static final String INSERT_INTO_TRATTO = "INSERT INTO tratto (serieorig, seriederiv) VALUES (md5(?), ?)";

    private static final String INSERT_INTO_PRODOTTO = "INSERT INTO prodotto (codprod, datains) VALUES (md5(?), DEFAULT) RETURNING
        codprod";
    private static final String INSERT_INTO_EDIZIONE = "INSERT INTO edizione (prodotto, nome, period, supporto, " +
        "col, inizpubbl, statoed, sovracop, extra, numvoltot, audio, video, sub, numpag, editore, serie) " +
        "VALUES (?, ?, ?::periodicita, ?::supporto, ?, ?::date, ?::statoserie, ?, ?, ?, ?, ?, ?, ?, ?, md5(?)) RETURNING prodotto";
    private static final String INSERT_INTO_VOLUME = "INSERT INTO volume (prodotto, datapubbl, ristampa, numrist, prezzofin, numvol, " +
        "disp, isbn, edizione) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, md5(?))";

    private static final String INSERT_INTO_ACCOUNT = "INSERT INTO account (codacc, username, pw, statoacc, tipoacc) " +
        "VALUES (DEFAULT, ?, md5(?), ?::statoaccount, ?::tipoaccount) RETURNING codacc";

```

```
77      private static final String INSERT_INTO_TESSERAMENTO = "INSERT␣INTO␣tesseramento␣(acctesserante,␣acctesserato,␣tesserato,␣tessera,␣
            datatess)␣" +
78          "VALUES␣(?,␣?,␣?,␣?,␣?)";
79      private static final String INSERT_INTO_TESSERA = "INSERT␣INTO␣tessera␣(codtess,␣inizioval,␣fineval,␣duratacasella,␣prezzotess,␣
            sconto)␣" +
80          "VALUES␣(DEFAULT,␣?,␣?,␣?,␣?,␣?)␣RETURNING␣codtess";
81      private static final String INSERT_INTO_TESSERATO = "INSERT␣INTO␣tesserato␣(codicefiscale,␣nome,␣cognome,␣nascita,␣mail)␣" +
82          "VALUES␣(?,␣?,␣?,␣?,␣?)";
83      private static final String INSERT_INTO_TELTESSERATO = "INSERT␣INTO␣teltesserato␣(tesserato,␣telfax)␣VALUES␣(?,␣?)";
84      private static final String INSERT_INTO_INDIRIZZO = "INSERT␣INTO␣indirizzo␣(codind,␣via,␣civico,␣cap,␣citta,␣provincia)␣" +
85          "VALUES␣(md5(?),␣?,␣?,␣?,␣?,␣?)␣RETURNING␣codind";
86      private static final String INSERT_INTO_INDIRIZZOTESSERATO = "INSERT␣INTO␣indirizzotesserato␣(tesserato,␣indirizzo)␣" +
87          "VALUES␣(?,␣?)";
88
89      private static final String INSERT_INTO_INCASELLAMENTO = "INSERT␣INTO␣incasellamento␣(accstaff,␣acctesserato,␣quantita,␣datains,␣
            datascad,␣" +
90          "datarim,␣prodotto)␣VALUES␣(?,␣?,␣?,␣?,␣?,␣?,␣?)";
91
92      private static final String INSERT_INTO_ORDINE = "INSERT␣INTO␣ordine␣(codord,␣statoord,␣datains,␣prezzotot,␣tipoord,␣tesserato)␣" +
93          "VALUES␣(DEFAULT,␣?::statoordine,␣?,␣?,␣?::tipoordine,␣?)␣RETURNING␣codord";
94      private static final String INSERT_INTO_MERCE = "INSERT␣INTO␣merce␣(ordine,␣quantita,␣prezzoelem,␣prodotto)␣VALUES␣(?,␣?,␣?,␣?)";
95      private static final String INSERT_INTO_LAVORAZIONE = "INSERT␣INTO␣lavorazione␣(ordine,␣acc,␣datalav)␣VALUES␣(?,␣?,␣?)";
96      private static final String INSERT_INTO_RIFORNIMENTO = "INSERT␣INTO␣rifornimento␣(ordine,␣distributore)␣VALUES␣(?,␣?)";
97
98      private static final String INSERT_INTO_NOTIFICA = "INSERT␣INTO␣notifica␣(mittente,␣destinatario,␣datainvio,␣datagen,␣oggetto,␣testo
            )␣" +
99          "VALUES␣(?,␣?,␣?,␣?,␣?,␣?)";
100
101
102     // insertSQLErrorException handling
103     private static void insertSQLExceptionHandler(SQLException e, String relation, String id) {
104
105       if (e.getSQLState().equals("23505")) {
106
107         System.out.printf("%s␣%s␣gia'␣inserito,␣verra'␣ignorato.␣[%s]%n", relation, id, e.getMessage());
108
109       } else {
110
111         System.out.printf("Errore␣nell'inserire␣%s␣%s:%n", relation, id);
112         System.out.printf("-␣Messaggio:␣%s%n", e.getMessage());
113         System.out.printf("-␣Codice␣di␣stato␣SQL:␣%s%n", e.getSQLState());
114         System.out.printf("-␣Codice␣di␣errore␣SQL:␣%s%n", e.getErrorCode());
115         System.out.printf("%n");
116
117       }
118     }
119
120     public static void main(String[] args) {
121
122       // Connessione al database
123       Connection connessione = null;
124
125       // Prepared statement
126       PreparedStatement insertIntoEditore = null;
127       PreparedStatement insertIntoDistributore = null;
128       PreparedStatement insertIntoDistribuzione = null;
129       PreparedStatement insertIntoMail = null;
130       PreparedStatement insertIntoMailDistr = null;
131       PreparedStatement insertIntoMailEd = null;
132       PreparedStatement insertIntoTelfax = null;
133       PreparedStatement insertIntoTelfaxDistr = null;
134       PreparedStatement insertIntoTelfaxEd = null;
135
136       PreparedStatement insertIntoSerie = null;
137       PreparedStatement insertIntoGenere = null;
138       PreparedStatement insertIntoGenereSerie = null;
139       PreparedStatement insertIntoTratto = null;
140
141       PreparedStatement insertIntoProdotto = null;
142       PreparedStatement insertIntoEdizione = null;
143       PreparedStatement insertIntoVolume = null;
144
145       PreparedStatement insertIntoAccount = null;
146
147       PreparedStatement insertIntoTesseramento = null;
148       PreparedStatement insertIntoTessera = null;
149       PreparedStatement insertIntoTesserato = null;
150       PreparedStatement insertIntoTelTesserato = null;
151       PreparedStatement insertIntoIndirizzo = null;
152       PreparedStatement insertIntoIndirizzoTesserato = null;
153
154       PreparedStatement insertIntoIncasellamento = null;
155
156       PreparedStatement insertIntoOrdine = null;
157       PreparedStatement insertIntoMerce = null;
158       PreparedStatement insertIntoLavorazione = null;
159       PreparedStatement insertIntoRifornimento = null;
160
161       PreparedStatement insertIntoNotifica = null;
162
163
164       // Input file
165       Reader inputFile = null;
166
167       // Scanner
168       Scanner scanner = null;
169
170       // Caricamento del driver
171       try {
172         Class.forName(DRIVER);
173       } catch (ClassNotFoundException e) {
174         System.out.printf(
175             "Driver␣%s␣non␣trovato:␣%s.%n", DRIVER, e.getMessage());
```

```
176
177         System.exit(-1);
178       }
179
180       // Connessione al DB e prepareStatement
181       try {
182         connessione = DriverManager.getConnection(DATABASE, USER, PASSWORD);
183
184         insertIntoEditore = connessione.prepareStatement(INSERT_INTO_EDITORE);
185         insertIntoDistributore = connessione.prepareStatement(INSERT_INTO_DISTRIBUTORE);
186         insertIntoMail = connessione.prepareStatement(INSERT_INTO_MAIL);
187         insertIntoMailDistr = connessione.prepareStatement(INSERT_INTO_MAILDISTR);
188         insertIntoMailEd = connessione.prepareStatement(INSERT_INTO_MAILED);
189         insertIntoTelfax = connessione.prepareStatement(INSERT_INTO_TELFAX);
190         insertIntoTelfaxDistr = connessione.prepareStatement(INSERT_INTO_TELFAXDISTR);
191         insertIntoTelfaxEd = connessione.prepareStatement(INSERT_INTO_TELFAXED);
192         insertIntoDistribuzione = connessione.prepareStatement(INSERT_INTO_DISTRIBUZIONE);
193
194         insertIntoSerie = connessione.prepareStatement(INSERT_INTO_SERIE);
195         insertIntoGenere = connessione.prepareStatement(INSERT_INTO_GENERE);
196         insertIntoGenereSerie = connessione.prepareStatement(INSERT_INTO_GENERESERIE);
197         insertIntoTratto = connessione.prepareStatement(INSERT_INTO_TRATTO);
198
199         insertIntoProdotto = connessione.prepareStatement(INSERT_INTO_PRODOTTO);
200         insertIntoEdizione = connessione.prepareStatement(INSERT_INTO_EDIZIONE);
201         insertIntoVolume = connessione.prepareStatement(INSERT_INTO_VOLUME);
202
203         insertIntoAccount = connessione.prepareStatement(INSERT_INTO_ACCOUNT);
204
205         insertIntoTesseramento = connessione.prepareStatement(INSERT_INTO_TESSERAMENTO);
206         insertIntoTessera = connessione.prepareStatement(INSERT_INTO_TESSERA);
207         insertIntoTesserato = connessione.prepareStatement(INSERT_INTO_TESSERATO);
208         insertIntoTelTesserato = connessione.prepareStatement(INSERT_INTO_TELTESSERATO);
209         insertIntoIndirizzo = connessione.prepareStatement(INSERT_INTO_INDIRIZZO);
210         insertIntoIndirizzoTesserato = connessione.prepareStatement(INSERT_INTO_INDIRIZZOTESSERATO);
211
212         insertIntoIncasellamento = connessione.prepareStatement(INSERT_INTO_INCASELLAMENTO);
213
214         insertIntoOrdine = connessione.prepareStatement(INSERT_INTO_ORDINE);
215         insertIntoMerce = connessione.prepareStatement(INSERT_INTO_MERCE);
216         insertIntoLavorazione = connessione.prepareStatement(INSERT_INTO_LAVORAZIONE);
217         insertIntoRifornimento = connessione.prepareStatement(INSERT_INTO_RIFORNIMENTO);
218
219         insertIntoNotifica = connessione.prepareStatement(INSERT_INTO_NOTIFICA);
220
221
222       } catch (SQLException e) {
223         System.out.printf("Errore di connessione:%n");
224
225         // Recupero dell'errore
226         while (e != null) {
227           System.out.printf("- Messaggio: %s%n", e.getMessage());
228           System.out.printf("- Codice di stato SQL: %s%n", e.getSQLState());
229           System.out.printf("- Codice di errore SQL: %s%n", e.getErrorCode());
230           System.out.printf("%n");
231           e = e.getNextException();
232         }
233
234         System.exit(-1);
235       }
236
237       // Variabili di supporto generali
238       int num; String input_file = null;
239
240       // Variabili di supporto per INSERT_INTO_EDITORE e INSERT_INTO_DISTRIBUTORE
241       String piva, nomeEdDistr, sito;
242       String[] mail, tel, fax;
243
244       // Variabili di supporto per INSERT_INTO_DISTRIBUZIONE
245       String editore, distributore;
246
247       // Variabili di supporto per INSERT_INTO_SERIE
248       String titolo, autore, media, statoSerie, trama, studio, regista, rivista;
249       int anno, numEp, numVolOrig;
250       String[] genere, serie;
251
252       // Variabili di supporto per INSERT_INTO_EDIZIONE, INSERT_INTO_VOLUME
253       String nome, period, supporto, dataPubblEd, statoEd, extra, audio, video, sub, editoreEd;
254       String dataPubblVol, isbn;
255       boolean col, sovraCop, rist;
256       int numVolTot, numPag, numVol, numRist, disp;
257       double prezzoVol;
258
259       // Variabili di supporto per INSERT_INTO_ACCOUNT
260       String user, pw, statoAcc, tipoAcc;
261
262       // Variabili di supporto per INSERT_INTO_TESSERATO
263       String codicefiscale, nomeTess, cognomeTess, mailTess, via, citta, civico, cap;
264       String provincia, accTess;
265       double prezzoTess;
266       int sconto, duratacasella;
267       String[] numTel;
268
269       // Variabili di supporto per INSERT_INTO_INCASELLAMENTO
270       String accStaff, ogg, prodotto;
271       Date datains, datarim;
272       int quantita;
273
274       // Variabili di supporto per INSERT_INTO_ORDINE
275       String statoOrd, tipoOrd, codiceordine;
276       Date datalav;
277       int codord;
278       double prezzoTot;
```

```
279
280       // Lettura dei file e inserimento dei dati nel database
281       try {
282         System.out.println("Popolamento della base di dati in corso...\n");
283
284         for (int n = 0; n < INPUT.length; n++) {
285           input_file = INPUT[n];
286
287           // Apertura del file di input da riga di comando
288           try {
289             inputFile = new BufferedReader(new FileReader(input_file));
290             System.out.printf("\nFile %s aperto con successo.%n", input_file);
291           } catch (IOException e) {
292             System.out.printf("Impossibile leggere il file %s: %s%n", input_file, e.getMessage());
293             System.exit(-1);
294           }
295
296           scanner = new Scanner(inputFile);
297           scanner.useDelimiter(";");
298
299           // Numero di riga
300           int riga = 0;
301
302           while (scanner.hasNext()) {
303             riga++;
304
305             switch (input_file) {
306
307             // INSERT_INTO_EDITORE
308             case INPUT_EDITORE:
309               piva = scanner.next();
310               nomeEdDistr = scanner.next();
311               sito = scanner.next();
312
313               num = scanner.nextInt();
314               mail = new String[num];
315
316               for (int i = 0; i < num; i++) {
317                 mail[i] = scanner.next();
318               }
319
320               num = scanner.nextInt();
321               tel = new String[num];
322
323               for (int i = 0; i < num; i++) {
324                 tel[i] = scanner.next();
325               }
326
327               num = scanner.nextInt();
328               fax = new String[num];
329
330               for (int i = 0; i < num; i++) {
331                 fax[i] = scanner.next();
332               }
333
334               try {
335                 insertIntoEditore.setString(1, piva);
336                 insertIntoEditore.setString(2, nomeEdDistr);
337                 insertIntoEditore.setString(3, sito);
338
339                 insertIntoEditore.execute();
340
341                 if (mail.length != 0) {
342                   for (int i = 0; i < mail.length; i++) {
343                     insertIntoMail.setString(1, mail[i]);
344                     insertIntoMailEd.setString(1, mail[i]);
345                     insertIntoMailEd.setString(2, piva);
346                     insertIntoMail.execute();
347                     insertIntoMailEd.execute();
348                   }
349                 }
350
351                 if (tel.length != 0) {
352                   for (int i = 0; i < tel.length; i++) {
353                     insertIntoTelfax.setString(1, tel[i] + "false");
354                     insertIntoTelfax.setString(2, tel[i]);
355                     insertIntoTelfax.setBoolean(3, false);
356                     ResultSet tmpRS = insertIntoTelfax.executeQuery();
357                     tmpRS.next();
358                     String tmpCod = tmpRS.getString(1);
359                     insertIntoTelfaxEd.setString(1, tmpCod);
360                     insertIntoTelfaxEd.setString(2, piva);
361                     insertIntoTelfaxEd.execute();
362                   }
363                 }
364
365                 if (fax.length != 0) {
366                   for (int i = 0; i < fax.length; i++) {
367                     insertIntoTelfax.setString(1, tel[i] + "true");
368                     insertIntoTelfax.setString(2, tel[i]);
369                     insertIntoTelfax.setBoolean(3, true);
370                     ResultSet tmpRS = insertIntoTelfax.executeQuery();
371                     tmpRS.next();
372                     String tmpCod = tmpRS.getString(1);
373                     insertIntoTelfaxEd.setString(1, tmpCod);
374                     insertIntoTelfaxEd.setString(2, piva);
375                     insertIntoTelfaxEd.execute();
376                   }
377                 }
378
379                 System.out.println("Riga " + riga + " inserita con successo!");
380               } catch (SQLException e) {
381                 insertSQLExceptionHandler(e, "Editore", nomeEdDistr + " (" + piva + ")");
```

```
382             }
383             break;
384
385             // INSERT_INTO_DISTRIBUTORE
386           case INPUT_DISTRIBUTORE:
387             piva = scanner.next();
388             nomeEdDistr = scanner.next();
389             sito = scanner.next();
390
391             num = scanner.nextInt();
392             mail = new String[num];
393
394             for (int i = 0; i < num; i++) {
395               mail[i] = scanner.next();
396             }
397
398             num = scanner.nextInt();
399             tel = new String[num];
400
401             for (int i = 0; i < num; i++) {
402               tel[i] = scanner.next();
403             }
404
405             num = scanner.nextInt();
406             fax = new String[num];
407
408             for (int i = 0; i < num; i++) {
409               fax[i] = scanner.next();
410             }
411
412             try {
413               insertIntoDistributore.setString(1, piva);
414               insertIntoDistributore.setString(2, nomeEdDistr);
415               insertIntoDistributore.setString(3, sito);
416
417               insertIntoDistributore.execute();
418
419               if (mail.length != 0) {
420                 for (int i = 0; i < mail.length; i++) {
421                   insertIntoMail.setString(1, mail[i]);
422                   insertIntoMailDistr.setString(1, mail[i]);
423                   insertIntoMailDistr.setString(2, piva);
424                   insertIntoMail.execute();
425                   insertIntoMailDistr.execute();
426                 }
427               }
428
429               if (tel.length != 0) {
430                 for (int i = 0; i < tel.length; i++) {
431                   insertIntoTelfax.setString(1, tel[i] + "false");
432                   insertIntoTelfax.setString(2, tel[i]);
433                   insertIntoTelfax.setBoolean(3, false);
434                   ResultSet tmpRS = insertIntoTelfax.executeQuery();
435                   tmpRS.next();
436                   insertIntoTelfaxDistr.setString(1, tmpRS.getString(1));
437                   insertIntoTelfaxDistr.setString(2, piva);
438                   insertIntoTelfaxDistr.execute();
439                 }
440               }
441
442               if (fax.length != 0) {
443                 for (int i = 0; i < fax.length; i++) {
444                   insertIntoTelfax.setString(1, tel[i] + "true");
445                   insertIntoTelfax.setString(2, fax[i]);
446                   insertIntoTelfax.setBoolean(3, true);
447                   ResultSet tmpRS = insertIntoTelfax.executeQuery();
448                   tmpRS.next();
449                   insertIntoTelfaxDistr.setString(1, tmpRS.getString(1));
450                   insertIntoTelfaxDistr.setString(2, piva);
451                   insertIntoTelfaxDistr.execute();
452                 }
453               }
454
455               System.out.println("Riga␣" + riga + "␣inserita␣con␣successo!");
456             } catch (SQLException e) {
457               insertSQLExceptionHandler(e, "Distributore", nomeEdDistr + "␣(" + piva + ")");
458             }
459             break;
460
461             // INSERT_INTO_DISTRIBUZIONE
462           case INPUT_DISTRIBUZIONE:
463             distributore = scanner.next();
464             editore = scanner.next();
465
466             try {
467               insertIntoDistribuzione.setString(1, distributore);
468               insertIntoDistribuzione.setString(2, editore);
469               insertIntoDistribuzione.execute();
470
471               System.out.println("Riga␣" + riga + "␣inserita␣con␣successo!");
472             } catch (SQLException e) {
473               insertSQLExceptionHandler(e, "Distribuzione", distributore + "␣<->␣" + editore);
474             }
475             break;
476
477             // INSERT_INTO_SERIE
478           case INPUT_SERIE:
479             //            codSerie = scanner.next();
480             titolo = scanner.next();
481             autore = scanner.next();
482             anno = scanner.nextInt();
483             media = scanner.next();
484             statoSerie = scanner.next();
```

```
485             trama = scanner.next();
486
487             if (media.equals("Anime"))
488               numEp = scanner.nextInt();
489             else
490               numEp = scanner.next().length(); // dummy
491
492             studio = scanner.next();
493
494             if (media.equals("Manga"))
495               numVolOrig = scanner.nextInt();
496             else
497               numVolOrig = scanner.next().length(); // dummy
498
499             regista = scanner.next();
500             rivista = scanner.next();
501
502             num = scanner.nextInt();
503             genere = new String[num];
504
505             for (int i = 0; i < genere.length; i++) {
506               genere[i] = scanner.next();
507             }
508
509             num = scanner.nextInt();
510             serie = new String[num*4];
511
512             for (int i = 0; i < serie.length; i++) {
513               serie[i] = scanner.next();
514             }
515
516
517             try {
518               //              insertIntoSerie.setString(1, codSerie);
519               insertIntoSerie.setString(1, (titolo+autore+anno+media));
520               insertIntoSerie.setString(2, titolo);
521               insertIntoSerie.setString(3, autore);
522               insertIntoSerie.setInt(4, anno);
523               insertIntoSerie.setString(5, media);
524               insertIntoSerie.setString(6, statoSerie);
525               insertIntoSerie.setString(7, trama);
526
527               if (media.equals("Anime"))
528                 insertIntoSerie.setInt(8, numEp);
529               else
530                 insertIntoSerie.setNull(8, 0);
531
532               if (media.equals("Anime") && !studio.equals("NULL"))
533                 insertIntoSerie.setString(9, studio);
534               else
535                 insertIntoSerie.setNull(9, 0);
536
537               if (media.equals("Manga"))
538                 insertIntoSerie.setInt(10, numVolOrig);
539               else
540                 insertIntoSerie.setNull(10, 0);
541
542               if (media.equals("Anime") && !regista.equals("NULL"))
543                 insertIntoSerie.setString(11, regista);
544               else
545                 insertIntoSerie.setNull(11, 0);
546
547               if (media.equals("Manga") && !rivista.equals("NULL"))
548                 insertIntoSerie.setString(12, rivista);
549               else
550                 insertIntoSerie.setNull(12, 0);
551
552               ResultSet getCodSerie = insertIntoSerie.executeQuery();
553               getCodSerie.next();
554
555               if (genere.length != 0) {
556                 for (int i = 0; i < genere.length; i++) {
557                   insertIntoGenere.setString(1, genere[i]);
558                   insertIntoGenere.setString(2, genere[i]);
559                   insertIntoGenere.execute();
560                   insertIntoGenereSerie.setString(1, genere[i]);
561                   //                  insertIntoGenereSerie.setString(2, codserie);
562                   insertIntoGenereSerie.setString(2, getCodSerie.getString(1));
563                   insertIntoGenereSerie.execute();
564                 }
565               }
566
567               if (serie.length != 0) {
568                 for (int i = 0; i < serie.length/4; i++) {
569                   insertIntoTratto.setString(1, serie[i]+serie[i+1]+serie[i+2]+serie[i+3]);
570                   //                  insertIntoTratto.setString(2, codSerie);
571                   insertIntoTratto.setString(2, getCodSerie.getString(1));
572                   insertIntoTratto.execute();
573                 }
574               }
575
576               System.out.println("Riga " + riga + " inserita con successo!");
577             } catch (SQLException e) {
578               insertSQLExceptionHandler(e, "Serie", titolo);
579             }
580             break;
581
582             // INPUT_VOLUME
583           case INPUT_VOLUME:
584             String tmpDatePubblVol = scanner.next();
585             dataPubblVol = tmpDatePubblVol.substring(6, 10) + tmpDatePubblVol.substring(2, 6) + tmpDatePubblVol.substring(0, 2);
586             Date dateVol = Date.valueOf(dataPubblVol);
587
```

```
588             rist = scanner.nextBoolean();
589             numRist = scanner.nextInt();
590             prezzoVol = Double.parseDouble(scanner.next());
591             numVol = scanner.nextInt();
592
593             scanner.next(); // ignoro la copertina
594
595             disp = scanner.nextInt();
596             isbn = scanner.next();
597
598             String tmpnome = scanner.next();
599             if (tmpnome.equals("NULL"))
600               nome = null;
601             else
602               nome = tmpnome;
603
604             period = scanner.next();
605
606             String tmpsupporto = scanner.next();
607             if (tmpsupporto.equals("NULL"))
608               supporto = null;
609             else
610               supporto = tmpsupporto;
611
612             String tmpcol = scanner.next();
613             if (tmpcol.equals("NULL"))
614               col = false;
615             else if (tmpcol.equals("true"))
616               col = true;
617             else
618               col = false;
619
620             dataPubblEd = scanner.next();
621
622             Date date = Date.valueOf(dataPubblEd);
623
624             statoEd = scanner.next();
625
626             String tmpsovracop = scanner.next();
627             if (tmpsovracop.equals("NULL"))
628               sovraCop = false;
629             else if (tmpsovracop.equals("true"))
630               sovraCop = true;
631             else
632               sovraCop = false;
633
634             String tmpextra = scanner.next();
635             if (tmpsovracop.equals("NULL"))
636               extra = null;
637             else
638               extra = tmpextra;
639
640             numVolTot = scanner.nextInt();
641
642             if (supporto != null) {
643               audio = scanner.next();
644               video = scanner.next();
645               sub = scanner.next();
646             } else {
647               audio = scanner.next();
648               video = scanner.next();
649               sub = scanner.next();
650               audio = video = sub = null;
651             }
652
653             String tmpnumpag = scanner.next();
654             if (tmpnumpag.equals("NULL"))
655               numPag = 0; // dummy
656             else
657               numPag = Integer.parseInt(tmpnumpag);
658
659             editoreEd = scanner.next();
660
661             String[] serieEdizione = new String[4];
662             for (int m = 0; m < serieEdizione.length; m++) {
663               serieEdizione[m] = scanner.next();
664             }
665
666             try {
667               String queryCheckProdotto = "SELECT codprod FROM prodotto WHERE codprod = md5(?)";
668               PreparedStatement checkProdotto = connessione.prepareStatement(queryCheckProdotto);
669               checkProdotto.setString(1, serieEdizione[0]+serieEdizione[1]+serieEdizione[2]+serieEdizione[3]);
670               ResultSet prodottoEsiste = checkProdotto.executeQuery();
671
672               if (prodottoEsiste.next()) {
673                 // violazione prodotto_codprod_pkey e edizione_prodotto_pkey
674                 // l'edizione esiste gia', ignoro e passo all'inserimento del volume
675               } else {
676                 insertIntoProdotto.setString(1, serieEdizione[0]+serieEdizione[1]+serieEdizione[2]+serieEdizione[3]);
677                 ResultSet prodottoedizione = insertIntoProdotto.executeQuery();
678                 prodottoedizione.next();
679                 String prodottoEdizione = prodottoedizione.getString(1);
680
681                 insertIntoEdizione.setString(1, prodottoEdizione);
682
683                 if (nome != null)
684                   insertIntoEdizione.setString(2, nome);
685                 else
686                   insertIntoEdizione.setNull(2, Types.VARCHAR);
687
688                 insertIntoEdizione.setString(3, period);
689
690                 if (supporto != null) {
```

```
691                   insertIntoEdizione.setString(4, supporto);
692                   insertIntoEdizione.setString(11, audio);
693                   insertIntoEdizione.setString(12, video);
694                   insertIntoEdizione.setString(13, sub);
695                 } else {
696                   insertIntoEdizione.setNull(4, Types.VARCHAR);
697                   insertIntoEdizione.setNull(11, Types.VARCHAR);
698                   insertIntoEdizione.setNull(12, Types.VARCHAR);
699                   insertIntoEdizione.setNull(13, Types.VARCHAR);
700                 }
701
702                 if (!col && !tmpcol.equals("NULL"))
703                   insertIntoEdizione.setBoolean(5, false);
704                 else
705                   insertIntoEdizione.setBoolean(5, true);
706
707                 insertIntoEdizione.setDate(6, date);
708
709                 insertIntoEdizione.setString(7, statoEd);
710
711                 if (!sovraCop && !tmpsovracop.equals("NULL"))
712                   insertIntoEdizione.setBoolean(8, false);
713                 else
714                   insertIntoEdizione.setBoolean(8, true);
715
716                 if (extra != null)
717                   insertIntoEdizione.setString(9, extra);
718                 else
719                   insertIntoEdizione.setNull(9, Types.NULL);
720
721                 insertIntoEdizione.setInt(10, numVolTot);
722
723                 if (numPag != 0)
724                   insertIntoEdizione.setInt(14, numPag);
725                 else
726                   insertIntoEdizione.setNull(14, Types.INTEGER);
727
728                 insertIntoEdizione.setString(15, editoreEd);
729                 insertIntoEdizione.setString(16, serieEdizione[0]+serieEdizione[1]+serieEdizione[2]+serieEdizione[3]);
730                 insertIntoEdizione.execute();
731               }
732
733               insertIntoProdotto.setString(1, isbn + numVol);
734               ResultSet prodottovolume = insertIntoProdotto.executeQuery();
735               prodottovolume.next();
736
737               insertIntoVolume.setString(1, prodottovolume.getString(1));
738               insertIntoVolume.setDate(2, dateVol);
739               insertIntoVolume.setBoolean(3, rist);
740               insertIntoVolume.setInt(4, numRist);
741               insertIntoVolume.setDouble(5, prezzoVol);
742               insertIntoVolume.setInt(6, numVol);
743               insertIntoVolume.setInt(7, disp);
744               insertIntoVolume.setString(8, isbn);
745               insertIntoVolume.setString(9, serieEdizione[0]+serieEdizione[1]+serieEdizione[2]+serieEdizione[3]);
746
747               insertIntoVolume.execute();
748
749               System.out.println("Riga " + riga + " inserita con successo!");
750             } catch (SQLException e) {
751               insertSQLExceptionHandler(e, "Volume", serieEdizione[0] + " (" + serieEdizione[3] + ") " + numVol);
752             }
753             break;
754
755           case INPUT_STAFF:
756             user = scanner.next();
757             pw = scanner.next();
758             statoAcc = scanner.next();
759             tipoAcc = scanner.next();
760
761             try {
762               insertIntoAccount.setString(1, user);
763               insertIntoAccount.setString(2, pw);
764               insertIntoAccount.setString(3, statoAcc);
765               insertIntoAccount.setString(4, tipoAcc);
766               insertIntoAccount.execute();
767
768               System.out.println("Riga " + riga + " inserita con successo!");
769             } catch (SQLException e) {
770               insertSQLExceptionHandler(e, "Account staff", user + " <-> " + tipoAcc);
771             }
772             break;
773
774           case INPUT_TESSERATO:
775             codicefiscale = scanner.next();
776             nomeTess = scanner.next();
777             cognomeTess = scanner.next();
778             Date datanascita = Date.valueOf(scanner.next());
779             mailTess = scanner.next();
780             via = scanner.next();
781             civico = scanner.next();
782             cap = scanner.next();
783             citta = scanner.next();
784             provincia = scanner.next();
785
786             int i = scanner.nextInt();
787             numTel = new String[i];
788             for (int k = 0; k < i; k++)
789               numTel[k] = scanner.next();
790
791             user = scanner.next();
792             pw = scanner.next();
793             statoAcc = scanner.next();
```

```
794                 tipoAcc = scanner.next();
795                 Date inizioVal = Date.valueOf(scanner.next());
796                 Date fineVal = Date.valueOf(scanner.next());
797                 duratacasella = scanner.nextInt();
798                 prezzoTess = Double.parseDouble(scanner.next());
799                 sconto = scanner.nextInt();
800                 accTess = scanner.next(); // account tesserante
801                 Date dataTess = Date.valueOf(scanner.next());
802
803                 try {
804                   insertIntoMail.setString(1, mailTess);
805                   insertIntoMail.execute();
806
807                   insertIntoTesserato.setString(1, codicefiscale);
808                   insertIntoTesserato.setString(2, nomeTess);
809                   insertIntoTesserato.setString(3, cognomeTess);
810                   insertIntoTesserato.setDate(4, datanascita);
811                   insertIntoTesserato.setString(5, mailTess);
812                   insertIntoTesserato.execute();
813
814                   insertIntoIndirizzo.setString(1, via+civico+citta+provincia);
815                   insertIntoIndirizzo.setString(2, via);
816                   insertIntoIndirizzo.setString(3, civico);
817                   insertIntoIndirizzo.setString(4, cap);
818                   insertIntoIndirizzo.setString(5, citta);
819                   insertIntoIndirizzo.setString(6, provincia);
820
821                   ResultSet indirizzo = insertIntoIndirizzo.executeQuery();
822                   indirizzo.next();
823
824                   insertIntoIndirizzoTesserato.setString(1, codicefiscale);
825                   insertIntoIndirizzoTesserato.setString(2, indirizzo.getString(1));
826                   insertIntoIndirizzoTesserato.execute();
827
828                   for (int j = 0; j < numTel.length; j++) {
829                     insertIntoTelfax.setString(1, numTel[j] + "false");
830                     insertIntoTelfax.setString(2, numTel[j]);
831                     insertIntoTelfax.setBoolean(3, false);
832
833                     ResultSet telefono = insertIntoTelfax.executeQuery();
834                     telefono.next();
835
836                     insertIntoTelTesserato.setString(1, codicefiscale);
837                     insertIntoTelTesserato.setString(2, telefono.getString(1));
838                     insertIntoTelTesserato.execute();
839                   }
840
841                   insertIntoAccount.setString(1, user);
842                   insertIntoAccount.setString(2, pw);
843                   insertIntoAccount.setString(3, statoAcc);
844                   insertIntoAccount.setString(4, tipoAcc);
845
846                   ResultSet account = insertIntoAccount.executeQuery();
847                   account.next();
848
849                   insertIntoTessera.setDate(1, inizioVal);
850                   insertIntoTessera.setDate(2, fineVal);
851                   insertIntoTessera.setInt(3, duratacasella);
852                   insertIntoTessera.setDouble(4, prezzoTess);
853                   insertIntoTessera.setInt(5, sconto);
854
855                   ResultSet tessera = insertIntoTessera.executeQuery();
856                   tessera.next();
857
858                   PreparedStatement getAccountTesserante = connessione.prepareStatement("SELECT codacc FROM account WHERE username = ?");
859                   getAccountTesserante.setString(1, accTess);
860                   ResultSet tesserante = getAccountTesserante.executeQuery();
861                   tesserante.next();
862
863                   insertIntoTesseramento.setInt(1, tesserante.getInt(1));
864                   insertIntoTesseramento.setInt(2, account.getInt(1));
865                   insertIntoTesseramento.setString(3, codicefiscale);
866                   insertIntoTesseramento.setInt(4, tessera.getInt(1));
867                   insertIntoTesseramento.setDate(5, dataTess);
868                   insertIntoTesseramento.execute();
869
870                   System.out.println("Riga " + riga + " inserita con successo!");
871                 } catch (SQLException e) {
872                   insertSQLExceptionHandler(e, "Tesseramento", user);
873                 }
874                 break;
875
876               case INPUT_INCASELLAMENTO:
877                 accStaff = scanner.next();
878                 accTess = scanner.next();
879                 quantita = scanner.nextInt();
880                 datains = Date.valueOf(scanner.next());
881
882                 String datetmp = scanner.next();
883                 if (!datetmp.equals("NULL"))
884                   datarim = Date.valueOf(datetmp);
885                 else
886                   datarim = null;
887
888                 prodotto = scanner.next();
889                 ogg = scanner.next();
890
891                 String titoloProdotto = ""; // dummy
892                 int numVolProdotto = 0; // dummy
893                 try {
894                   String queryAccount = "SELECT codacc FROM account WHERE username = ?";
895                   PreparedStatement getAccount = connessione.prepareStatement(queryAccount);
896
```

```
                getAccount.setString(1, accStaff);
                ResultSet codAccountStaff = getAccount.executeQuery();
                codAccountStaff.next();
                int codStaff = codAccountStaff.getInt(1);

                getAccount.setString(1, accTess);
                ResultSet codAccountTess = getAccount.executeQuery();
                codAccountTess.next();
                int codTess = codAccountTess.getInt(1);

                String queryScadenza = "SELECT T.duratacasella FROM tessera AS T INNER JOIN tesseramento AS TS " +
                    "ON T.codtess = TS.tessera WHERE acctesserato = ?";
                PreparedStatement getScadenza = connessione.prepareStatement(queryScadenza);
                getScadenza.setInt(1, codTess);
                ResultSet scadenza = getScadenza.executeQuery();
                scadenza.next();
                int mesiGiacenza = scadenza.getInt(1);
                String dataIns = datains.toString();
                Calendar dataScadenza = Calendar.getInstance(); // dummy
                dataScadenza.set(Integer.parseInt(dataIns.substring(0, 4)), (Integer.parseInt(dataIns.substring(5, 7)) - 1 +
                     mesiGiacenza),
                    Integer.parseInt(dataIns.substring(8, 10)));
                Date dataScad = new Date(dataScadenza.getTimeInMillis());

                String queryProdotto = "SELECT V.prodotto , S.titolo , V.numvol FROM volume AS V INNER JOIN edizione AS E " +
                    "ON V.edizione = E.prodotto INNER JOIN serie AS S ON E.serie = S.codserie WHERE isbn = ?";
                PreparedStatement getDatiProdotto = connessione.prepareStatement(queryProdotto);
                getDatiProdotto.setString(1, prodotto);
                ResultSet datiProdotto = getDatiProdotto.executeQuery();
                datiProdotto.next();
                String codProdotto = datiProdotto.getString(1);
                titoloProdotto = datiProdotto.getString(2);
                numVolProdotto = datiProdotto.getInt(3);
                String text = "Il prodotto " + titoloProdotto + " " + numVolProdotto + " e' stato depositato nella tua casella!";

                insertIntoIncasellamento.setInt(1, codStaff);
                insertIntoIncasellamento.setInt(2, codTess);
                insertIntoIncasellamento.setInt(3, quantita);
                insertIntoIncasellamento.setDate(4, datains);
                insertIntoIncasellamento.setDate(5, dataScad);

                if (datarim != null)
                  insertIntoIncasellamento.setDate(6, datarim);
                else
                  insertIntoIncasellamento.setNull(6, Types.DATE);

                insertIntoIncasellamento.setString(7, codProdotto);
                insertIntoIncasellamento.execute();

                insertIntoNotifica.setInt(1, codStaff);
                insertIntoNotifica.setInt(2, codTess);
                insertIntoNotifica.setDate(3, datains);
                insertIntoNotifica.setDate(4, datains);
                insertIntoNotifica.setString(5, ogg);
                insertIntoNotifica.setString(6, text);
                insertIntoNotifica.execute();

                System.out.println("Riga " + riga + " inserita con successo!");
              } catch (SQLException e) {
                insertSQLExceptionHandler(e, "Incasellamento", accTess + ": " + titoloProdotto + " " + numVolProdotto);
              }
              break;

            case INPUT_ORDINE:
              statoOrd = scanner.next();
              datains = Date.valueOf(scanner.next());
              tipoOrd = scanner.next();

              String tmp = scanner.next();
              if (tmp.equals("NULL"))
                accTess = null;
              else
                accTess = tmp;

              quantita = scanner.nextInt();
              prodotto = scanner.next();
              datalav = Date.valueOf(scanner.next());
              accStaff = scanner.next();
              codiceordine = "Ordine problematico!"; // dummy

              try {
                PreparedStatement getProdotto = connessione.prepareStatement("SELECT prodotto , prezzofin FROM volume WHERE isbn = ?");
                getProdotto.setString(1, prodotto);
                ResultSet codiceProdotto = getProdotto.executeQuery();
                codiceProdotto.next();

                String codprod = codiceProdotto.getString(1);
                prezzoVol = codiceProdotto.getDouble(2);

                PreparedStatement getAccount = connessione.prepareStatement("SELECT codacc FROM account WHERE username = ?");
                getAccount.setString(1, accStaff);
                ResultSet accountStaff = getAccount.executeQuery();
                accountStaff.next();
                int staff = accountStaff.getInt(1);

                String tess = ""; // dummy
                sconto = 0;
                if (accTess != null) {
                  String queryTess = "SELECT T.tesserato , TS.sconto FROM account AS A INNER JOIN tesseramento AS T ON A.codacc = T.
                       acctesserato " +
                      "INNER JOIN tessera AS TS ON TS.codtess = T.tessera WHERE A.username = ?";
                  PreparedStatement getTesserato = connessione.prepareStatement(queryTess);
                  getTesserato.setString(1, accTess);
```

```
998                 ResultSet tesserato = getTesserato.executeQuery();
999                 tesserato.next();
1000                tess = tesserato.getString(1);
1001                sconto = tesserato.getInt(2);
1002              }
1003
1004              prezzoTot = prezzoVol * quantita;
1005              double prezzoScontato = prezzoTot - prezzoTot * sconto / 100;
1006
1007              insertIntoOrdine.setString(1, statoOrd);
1008              insertIntoOrdine.setDate(2, datains);
1009              insertIntoOrdine.setDouble(3, prezzoScontato);
1010              insertIntoOrdine.setString(4, tipoOrd);
1011
1012              if (accTess != null)
1013                insertIntoOrdine.setString(5, tess);
1014              else
1015                insertIntoOrdine.setNull(5, Types.VARCHAR);
1016
1017              ResultSet ordine = insertIntoOrdine.executeQuery();
1018              ordine.next();
1019              codord = ordine.getInt(1);
1020              codiceordine = String.valueOf(codord);
1021
1022              insertIntoMerce.setInt(1, codord);
1023              insertIntoMerce.setInt(2, quantita);
1024              insertIntoMerce.setDouble(3, prezzoVol);
1025              insertIntoMerce.setString(4, codprod);
1026              insertIntoMerce.execute();
1027
1028              insertIntoLavorazione.setInt(1, codord);
1029              insertIntoLavorazione.setInt(2, staff);
1030              insertIntoLavorazione.setDate(3, datalav);
1031              insertIntoLavorazione.execute();
1032
1033              if (tipoOrd.equals("Fornitura")) {
1034                String queryEd = "SELECT D.distributore FROM (volume AS V INNER JOIN edizione AS E ON V.edizione = E.prodotto) AS VE "
                       +
1035                    " INNER JOIN distribuzione AS D ON D.editore = VE.editore WHERE VE.isbn = ?";
1036                PreparedStatement getEditore = connessione.prepareStatement(queryEd);
1037                getEditore.setString(1, prodotto);
1038                ResultSet editoreProdotto = getEditore.executeQuery();
1039                editoreProdotto.next();
1040
1041                insertIntoRifornimento.setInt(1, codord);
1042                insertIntoRifornimento.setString(2, editoreProdotto.getString(1));
1043                insertIntoRifornimento.execute();
1044              }
1045
1046              System.out.println("Riga " + riga + " inserita con successo!");
1047            } catch (SQLException e) {
1048              insertSQLExceptionHandler(e, "Ordine", codiceordine);
1049            }
1050            break;
1051
1052            // Gestione di input non validi
1053          default:
1054            System.out.println("Errore: il nome e/o il percorso del file non sono validi.");
1055            break;
1056          }
1057
1058          scanner.nextLine();
1059        }
1060      }
1061      System.out.println("\nPopolamento della base di dati terminato con successo.\n");
1062    } finally {
1063      scanner.close();
1064      System.out.println("File chiuso con successo.");
1065
1066      try {
1067        insertIntoEditore.close();
1068        insertIntoDistributore.close();
1069        insertIntoMail.close();
1070        insertIntoMailDistr.close();
1071        insertIntoMailEd.close();
1072        insertIntoTelfax.close();
1073        insertIntoTelfaxDistr.close();
1074        insertIntoTelfaxEd.close();
1075        insertIntoDistribuzione.close();
1076        insertIntoSerie.close();
1077        insertIntoGenere.close();
1078        insertIntoGenereSerie.close();
1079        insertIntoTratto.close();
1080        insertIntoProdotto.close();
1081        insertIntoEdizione.close();
1082        insertIntoVolume.close();
1083        insertIntoAccount.close();
1084        insertIntoTesseramento.close();
1085        insertIntoTessera.close();
1086        insertIntoTesserato.close();
1087        insertIntoTelTesserato.close();
1088        insertIntoIndirizzo.close();
1089        insertIntoIndirizzoTesserato.close();
1090        insertIntoIncasellamento.close();
1091        insertIntoOrdine.close();
1092        insertIntoMerce.close();
1093        insertIntoLavorazione.close();
1094        insertIntoRifornimento.close();
1095        insertIntoNotifica.close();
1096
1097        System.out.println("Statement chiusi con successo.");
1098
1099        connessione.close();
```

```
            System.out.println("Connessione chiusa con successo.");
        } catch (SQLException e) {
            System.out.printf("Errore nel rilasciare le risorse:%n");

            // Recupero l'errore
            while (e != null) {
                System.out.printf("- Messaggio: %s%n", e.getMessage());
                System.out.printf("- Codice di stato SQLe: %s%n", e.getSQLState());
                System.out.printf("- Codice di errore SQL: %s%n", e.getErrorCode());
                System.out.printf("%n");
                e = e.getNextException();
            }
        }
    }
  }
}
```

Interrogazioni principali

```
-- QUERY NUMERO 1 [FIGURA 1]
-- Recupera i prodotti depositati nella casella del cliente tesserato Luca Vallerini: di questi,
-- viene mostrato il titolo della serie, il numero del volume, la quantita' di volumi
-- presenti in casella e il prezzo per singolo volume.

SELECT S.titolo AS prodotto, numero, prezzo_cadauno, quantita, prezzo_cadauno*quantita AS prezzo_totale FROM
  serie AS S INNER JOIN (edizione AS E INNER JOIN
    (SELECT V.numvol AS numero, V.prezzofin AS prezzo_cadauno, P.quantita AS quantita, V.edizione FROM
      (volume AS V INNER JOIN (incasellamento AS I INNER JOIN
        ((SELECT codicefiscale FROM tesserato WHERE nome = 'Luca' AND cognome = 'Vallerini') AS TS INNER JOIN
          tesseramento AS ON TS.codicefiscale = T.tesserato) AS A ON A.acctesserato = I.acctesserato) AS P ON P.prodotto = V.prodotto)
            ) AS VP
        ON VP.edizione = E.prodotto) AS VPE ON VPE.serie = S.codserie
        GROUP BY S.titolo, numero, prezzo_cadauno, quantita, prezzo_totale ORDER BY numero ASC;

--  QUERY NUMERO 2 [FIGURA 2]
-- Oggi e' arrivato in fumetteria il volume 1 di 'Gen di Hiroshima':
-- tale volume viene inserito a catalogo con i sequenti parametri: titolo=Gen di Hiroshima,
-- volume=1, prezzo=E 42.00, ISBN=1522583690251, pagine a colori=nessuna, numero di pagine=1000,
-- sovraccoperta=nessuna, autore=Keiji Nakazawa, anno=1982, editore=Hikari,
-- stato edizione=In corso, periodicita'=Irregolare, volumi arrivati=3.

-- Query per l'inserimento
INSERT INTO serie (codserie, titolo, autore, anno, media, statoserie)
  SELECT md5('Gen di Hiroshima'||'Keiji Nakazawa'||1982||'Manga'), 'Gen di Hiroshima', 'Keiji Nakazawa', 1982, 'Manga', 'Completa'
    WHERE NOT EXISTS (SELECT codserie FROM serie WHERE codserie = md5('Gen di Hiroshima'||'Keiji Nakazawa'||1982||'Manga'));

INSERT INTO prodotto (codprod, datains) VALUES (md5('Gen di Hiroshima'||'Keiji Nakazawa'||1982||'Manga'||2015-06-15), DEFAULT);

INSERT INTO edizione (prodotto, period, col, inizpubbl, statoed, sovracop, numpag, editore, serie)
  VALUES (md5('Gen di Hiroshima'||'Keiji Nakazawa'||1982||'Manga'||2015-06-15), 'Irregolare', FALSE, '2015-06-15', 'In corso', FALSE,
      1000,
    (SELECT piva FROM editore WHERE nome LIKE '%Hikari%'), md5('Gen di Hiroshima'||'Keiji Nakazawa'||1982||'Manga'));

INSERT INTO prodotto (codprod, datains) VALUES (md5('1522583690251'||1), DEFAULT);
INSERT INTO volume (prodotto, datapubbl, prezzofin, numvol, disp, isbn, edizione) VALUES
  (md5('1522583690251'||1), '2015-06-15', '42', 1, 3, '1522583690251', md5('Gen di Hiroshima'||'Keiji Nakazawa'||1982||'Manga'
      ||2015-06-15));

-- Query per la verifica dell'inserimento [FIGURA 2]
SELECT S.titolo, V.numvol AS volume, V.prezzofin AS prezzo, V.disp AS quantita, E.period AS periodicita,
  E.col AS pagine_colori, E.numpag AS numero_pag, V.isbn, editore.nome AS editore FROM
    volume AS V INNER JOIN edizione AS E ON V.edizione = E.prodotto INNER JOIN serie AS S ON E.serie = S.codserie
      INNER JOIN editore ON E.editore = piva WHERE isbn = '1234567890123';

-- QUERY NUMERO 3 [FIGURA 3]
-- Ordina in ordine decrescente gli editori in base al prezzo medio dei loro prodotti a catalogo,
-- indicando prezzo medio e numero di volumi pubblicati a catalogo.

SELECT editore.nome AS editore, COUNT(volume.numvol) AS numero_vol_pubblicati, AVG(prezzofin) AS prezzo_medio FROM
  editore INNER JOIN edizione ON editore.piva = edizione.editore INNER JOIN volume ON edizione.prodotto = volume.edizione
    GROUP BY editore.nome ORDER BY prezzo_medio DESC;


-- QUERY NUMERO 4 [FIGURA 4]
-- Lista della spesa: l'ultimo numero di Monster, primo numero di Dr. Slump e primo numero di Maison Ikkoku.
-- L'ordine di acquisto viene effettuato dal 'Titolare' a beneficio del cliente con tessera numero 4.

-- Query per l'inserimento dell'ordine
SELECT SUM(V.prezzofin)-SUM(V.prezzofin)*((SELECT sconto FROM tesseramento INNER JOIN tessera ON tesseramento.tessera = tessera.
    codtess WHERE codtess = 4))/100
  INTO prezzo_totale FROM serie AS S INNER JOIN edizione AS E ON S.codserie = E.serie INNER JOIN volume as V ON V.edizione = E.
      prodotto
    WHERE (S.titolo = 'Dr. Slump' AND V.numvol = 1) OR (S.titolo = 'Maison Ikkoku' AND V.numvol = 1) OR (S.titolo = 'Monster' AND V.
        numvol = 9);

SELECT V.prezzofin INTO prezzo_elem_1 FROM serie AS S INNER JOIN edizione AS E ON S.codserie = E.serie INNER JOIN volume as V ON V.
    edizione = E.prodotto
  WHERE (S.titolo = 'Dr. Slump' AND V.numvol = 1);
SELECT V.prodotto INTO prodotto_1 FROM serie AS S INNER JOIN edizione AS E ON S.codserie = E.serie INNER JOIN volume as V ON V.
    edizione = E.prodotto
  WHERE (S.titolo = 'Dr. Slump' AND V.numvol = 1);

SELECT V.prezzofin INTO prezzo_elem_2 FROM serie AS S INNER JOIN edizione AS E ON S.codserie = E.serie INNER JOIN volume as V ON V.
    edizione = E.prodotto
  WHERE (S.titolo = 'Maison Ikkoku' AND V.numvol = 1);
SELECT V.prodotto INTO prodotto_2 FROM serie AS S INNER JOIN edizione AS E ON S.codserie = E.serie INNER JOIN volume as V ON V.
    edizione = E.prodotto
  WHERE (S.titolo = 'Maison Ikkoku' AND V.numvol = 1);

SELECT V.prezzofin INTO prezzo_elem_3 FROM serie AS S INNER JOIN edizione AS E ON S.codserie = E.serie INNER JOIN volume as V ON V.
    edizione = E.prodotto
  WHERE (S.titolo = 'Monster' AND V.numvol = 9);
```

```sql
SELECT V.prodotto INTO prodotto_3 FROM serie AS S INNER JOIN edizione AS E ON S.codserie = E.serie INNER JOIN volume as V ON V.
    edizione = E.prodotto
  WHERE (S.titolo = 'Monster' AND V.numvol = 9);

SELECT sconto INTO sconto_tessera FROM tesseramento INNER JOIN tessera ON tesseramento.tessera = tessera.codtess WHERE codtess = 4;

INSERT INTO ordine (statoord, datains, prezzotot, tipoord, tesserato) VALUES ('Evaso', '2015-06-18', (SELECT * FROM prezzo_totale), '
    Acquisto', (SELECT tesserato FROM tesseramento WHERE tessera = 4));

SELECT MAX(codord) INTO codice_ordine_ult FROM ordine;

INSERT INTO merce (ordine, prodotto, quantita, prezzoelem) VALUES ((SELECT * FROM codice_ordine_ult), (SELECT * FROM prodotto_1), 1, (
    SELECT * FROM prezzo_elem_1));
INSERT INTO merce (ordine, prodotto, quantita, prezzoelem) VALUES ((SELECT * FROM codice_ordine_ult), (SELECT * FROM prodotto_2), 1, (
    SELECT * FROM prezzo_elem_2));
INSERT INTO merce (ordine, prodotto, quantita, prezzoelem) VALUES ((SELECT * FROM codice_ordine_ult), (SELECT * FROM prodotto_3), 1, (
    SELECT * FROM prezzo_elem_3));

INSERT INTO lavorazione (ordine, acc, datalav) VALUES ((SELECT * FROM codice_ordine_ult), (SELECT codacc FROM account WHERE username =
     'Titolare'), '2015-06-18');

-- Query per la verifica dell'inserimento [FIGURA 4]
SELECT serie.titolo, volume.numvol, merce.quantita, volume.prezzofin, tesserato.nome || ' ' || tesserato.cognome AS cliente FROM
  ordine INNER JOIN merce ON ordine.codord = merce.ordine INNER JOIN lavorazione ON lavorazione.ordine = ordine.codord
    INNER JOIN volume ON volume.prodotto = merce.prodotto
      INNER JOIN tesserato ON ordine.tesserato = tesserato.codicefiscale
        INNER JOIN edizione ON volume.edizione = edizione.prodotto INNER JOIN serie ON
          edizione.serie = serie.codserie WHERE codord = (SELECT MAX(codord) FROM ordine);
```

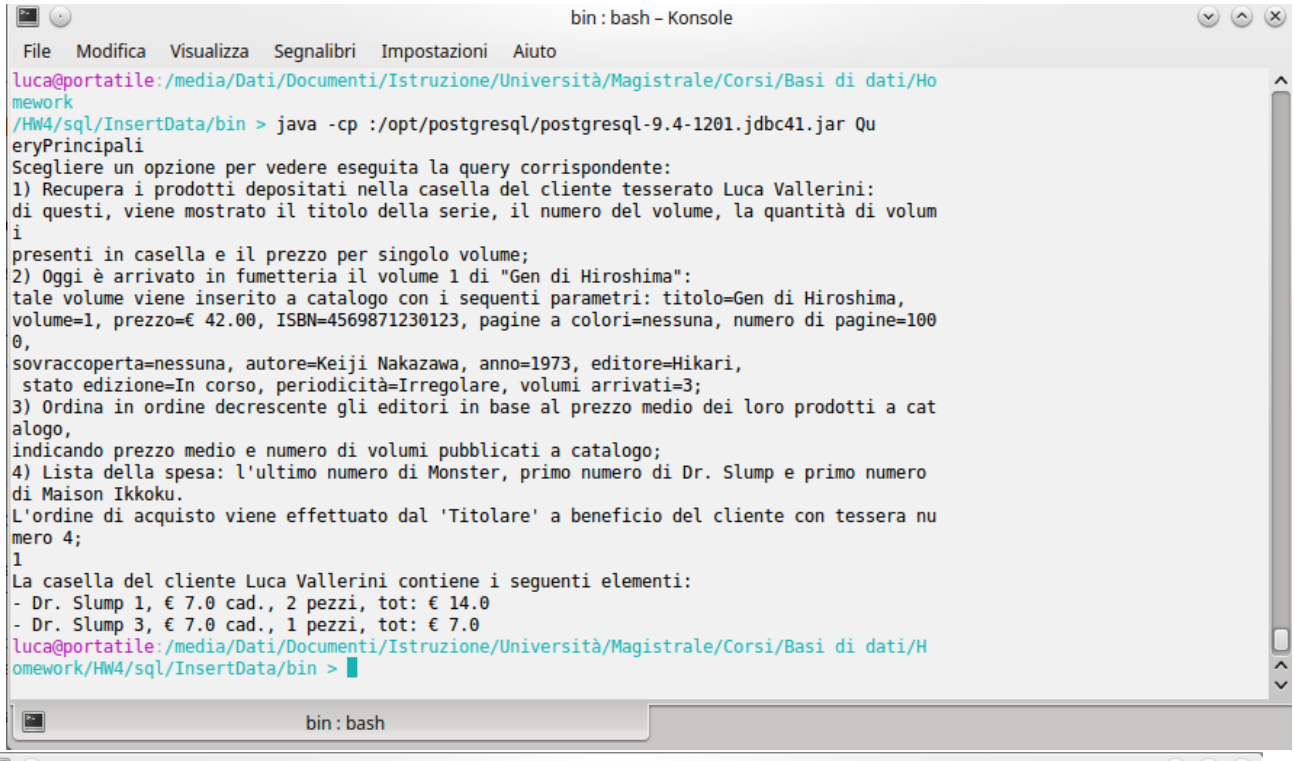
Implementazione JDBC delle interrogazioni principali e visualizzazione

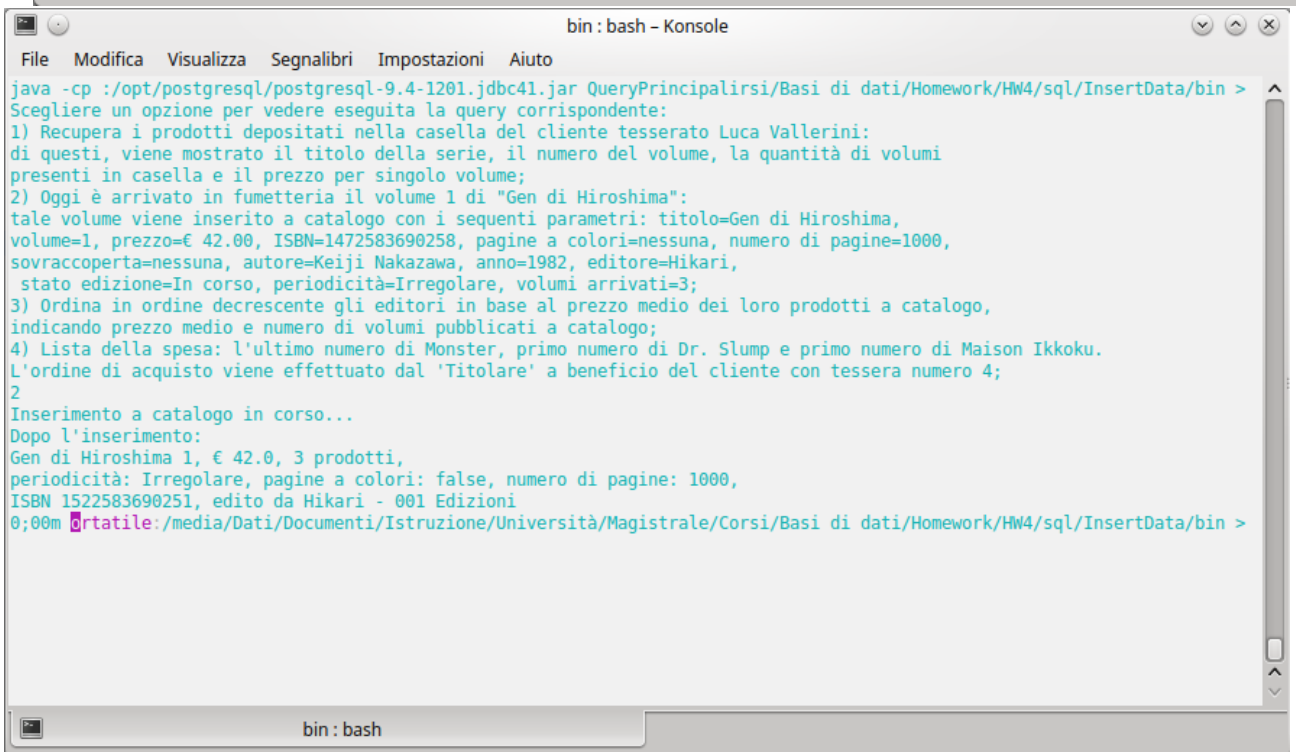
```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;

/**
 * Universita' degli Studi di Padova
 * Dipartimento di Ingegneria dell'Informazione
 * Corso di laurea magistrale in Ingegneria Informatica
 *
 * Basi di dati - Anno Accademico 2014/2015
 *
 * Homework 4 - Interrogazioni principali per la base di dati 'fumetteria'.
 *
 * @author Luca Vallerini - matr. 1110975
 * @version 0.1.20150614
 */
public class QueryPrincipali {

  // Informazioni essenziali per la connessione al database
  private static final String DRIVER = "org.postgresql.Driver";
  private static final String DATABASE = "jdbc:postgresql://localhost/fumetteria";
  private static final String USER = "luca";
  private static final String PASSWORD = "firefox";

  // Recupero dei prodotto depositati in una casella
  private static final String QUERY_1 = "Recupera i prodotti depositati nella casella del cliente tesserato Luca Vallerini: \n" +
      "di questi, viene mostrato il titolo della serie, il numero del volume, la quantita' di volumi \n" +
      "presenti in casella e il prezzo per singolo volume;";

  // Inserimento di un nuovo volume a catalogo
  private static final String QUERY_2 = "Oggi e' arrivato in fumetteria il volume 1 di \"Gen di Hiroshima\": \n" +
      "tale volume viene inserito a catalogo con i sequenti parametri: titolo=Gen di Hiroshima, \n" +
      "volume=1, prezzo=E 42.00, ISBN=1522583690251, pagine a colori=nessuna, numero di pagine=1000, \n" +
      "sovraccoperta=nessuna, autore=Keiji Nakazawa, anno=1982, editore=Hikari, \n" +
      "stato edizione=In corso, periodicita'=Irregolare, volumi arrivati=3;";

  private static final String QUERY_3 = "Ordina in ordine decrescente gli editori in base al prezzo medio dei loro prodotti a catalogo
      , \n" +
      "indicando prezzo medio e numero di volumi pubblicati a catalogo;";

  private static final String QUERY_4 = "Lista della spesa: l'ultimo numero di Monster, primo numero di Dr. Slump e primo numero di
       Maison Ikkoku. \n" +
      "L'ordine di acquisto viene effettuato dal 'Titolare' a beneficio del cliente con tessera numero 4;";

  // insertSQLErrorException handling
  private static void insertSQLExceptionHandler(SQLException e) {
    System.out.printf("Errore di accesso al database:%n");
    System.out.printf("- Messaggio: %s%n", e.getMessage());
    System.out.printf("- Codice di stato SQL: %s%n", e.getSQLState());
    System.out.printf("- Codice di errore SQL: %s%n", e.getErrorCode());
    System.out.printf("%n");

  }

  private static void connectSQLExceptionHandler(SQLException e) {
    System.out.printf("Errore di connessione:%n");

    while (e != null) {
      System.out.printf("- Messaggio: %s%n", e.getMessage());
      System.out.printf("- Codice di stato SQL: %s%n", e.getSQLState());
      System.out.printf("- Codice di errore SQL: %s%n", e.getErrorCode());
      System.out.printf("%n");
      e = e.getNextException();
    }
  }

  public static void main(String[] args) {
    System.out.println("Scegliere un opzione per vedere eseguita la query corrispondente:");
```

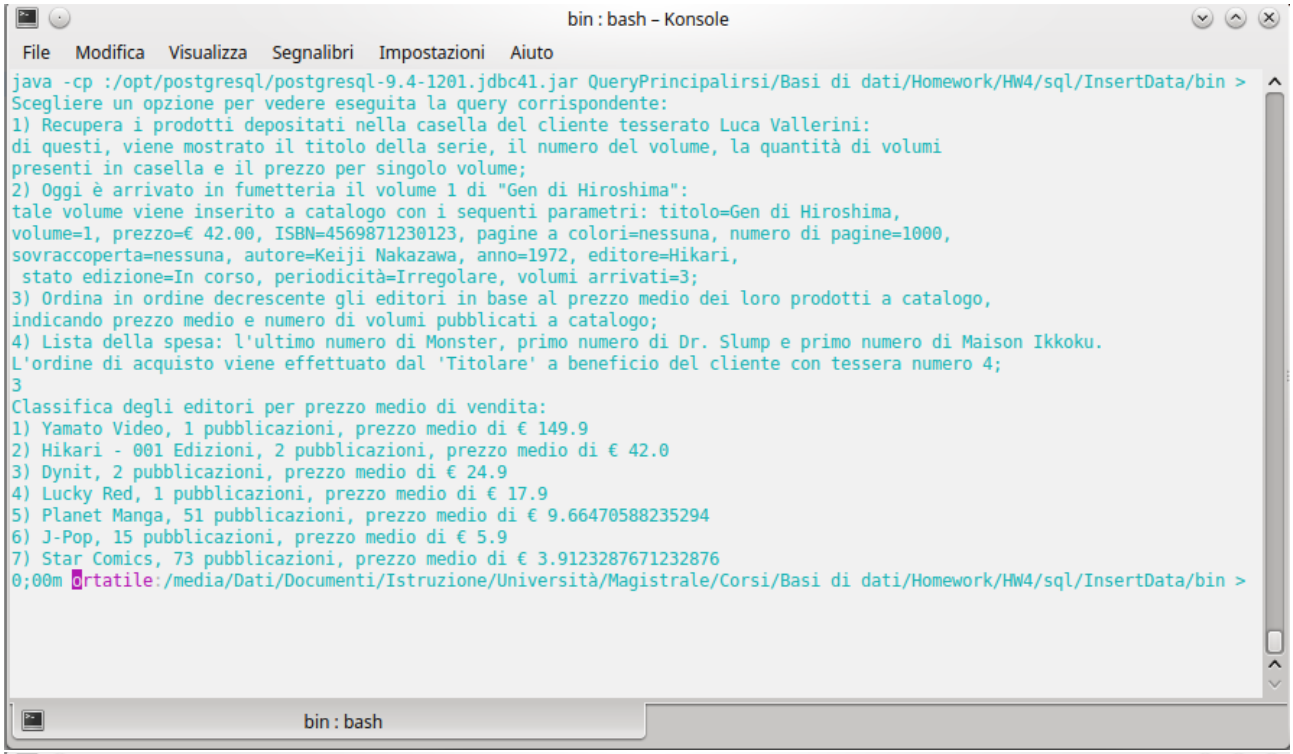
```
70        System.out.println("1) " + QUERY_1);
71        System.out.println("2) " + QUERY_2);
72        System.out.println("3) " + QUERY_3);
73        System.out.println("4) " + QUERY_4);
74
75        Scanner input = new Scanner(System.in);
76        int choice = new Scanner(System.in).nextInt();
77
78        // Connessione al database
79        Connection connessione = null;
80
81        // PreparedStatement e ResultSet
82        PreparedStatement queryPS = null;
83        PreparedStatement verifyQuery = null;
84        ResultSet queryRS = null;
85
86        // Caricamento del driver
87        try {
88          Class.forName(DRIVER);
89        } catch (ClassNotFoundException e) {
90          System.out.printf("Driver %s non trovato: %s.%n", DRIVER, e.getMessage());
91          System.exit(-1);
92        }
93
94        // Connessione al DB
95        try {
96          connessione = DriverManager.getConnection(DATABASE, USER, PASSWORD);
97        } catch (SQLException e) {
98          connectSQLExceptionHandler(e);
99          System.exit(-1);
100       }
101
102       switch (choice) {
103       case 1:
104         String query1 = "SELECT S.titolo AS prodotto, numero, prezzo_cadauno, quantita, prezzo_cadauno*quantita AS prezzo_totale FROM " +
105             "serie AS S INNER JOIN (edizione AS E INNER JOIN " +
106             "(SELECT V.numvol AS numero, V.prezzofin AS prezzo_cadauno, P.quantita AS quantita, V.edizione FROM " +
107             "(volume AS V INNER JOIN (incasellamento AS I INNER JOIN " +
108             "((SELECT codicefiscale FROM tesserato WHERE nome = 'Luca' AND cognome = 'Vallerini') AS TS INNER JOIN tesseramento AS T " +
109             "ON TS.codicefiscale = T.tesserato) AS A ON A.acctesserato = I.acctesserato) AS P ON P.prodotto = V.prodotto)) AS VP ON " +
110             "VP.edizione = E.prodotto) AS VPE ON VPE.serie = S.codserie " +
111             "GROUP BY S.titolo, numero, prezzo_cadauno, quantita, prezzo_totale ORDER BY numero ASC";
112
113         try {
114           queryPS = connessione.prepareStatement(query1);
115         } catch (SQLException e) {
116           connectSQLExceptionHandler(e);
117           System.exit(-1);
118         }
119
120         try {
121           queryRS = queryPS.executeQuery();
122
123           System.out.println("La casella del cliente Luca Vallerini contiene i seguenti elementi:");
124           while (queryRS.next()) {
125             System.out.println("- " + queryRS.getString(1) + " " + queryRS.getInt(2)+ ", E " + queryRS.getDouble(3) + " cad., " +
126                 queryRS.getInt(4) + " pezzi, tot: E " + queryRS.getDouble(5));
127           }
128         } catch (SQLException e) {
129           insertSQLExceptionHandler(e);
130         }
131         break;
132
133       case 2:
134         String query2 = "INSERT INTO serie (codserie, titolo, autore, anno, media, statoserie) " +
135             "SELECT md5('Gen di Hiroshima' || 'Keiji Nakazawa' ||1982|| 'Manga'), 'Gen di Hiroshima', 'Keiji Nakazawa', 1982, 'Manga', '
                  Completa' " +
136             "WHERE NOT EXISTS (SELECT codserie FROM serie WHERE codserie = md5('Gen di Hiroshima' || 'Keiji Nakazawa' ||1982|| 'Manga')); " +
137             "INSERT INTO prodotto (codprod, datains) VALUES (md5('Gen di Hiroshima' || 'Keiji Nakazawa' ||1982|| 'Manga'||2015-06-15),
                  DEFAULT); " +
138             "INSERT INTO edizione (prodotto, period, col, inizpubbl, statoed, sovracop, numpag, editore, serie) " +
139             "VALUES (md5('Gen di Hiroshima' || 'Keiji Nakazawa' ||1982|| 'Manga'||2015-06-15), 'Irregolare', FALSE, '2015-06-15', 'In corso
                  ', FALSE, 1000, " +
140             "(SELECT piva FROM editore WHERE nome LIKE '%Hikari%'), md5('Gen di Hiroshima' || 'Keiji Nakazawa' ||1982|| 'Manga')); " +
141             "INSERT INTO prodotto (codprod, datains) VALUES (md5('1522583690251'||1), DEFAULT); " +
142             "INSERT INTO volume (prodotto, datapubbl, prezzofin, numvol, disp, isbn, edizione) " +
143             "VALUES (md5('1522583690251'||1), '2015-06-15', '42', 1, 3, '1522583690251', md5('Gen di Hiroshima' || 'Keiji Nakazawa
                  '||1982|| 'Manga'||2015-06-15));";
144
145         String verifyQuery2 = "SELECT S.titolo, V.numvol AS volume, V.prezzofin AS prezzo, V.disp AS quantita, E.period AS periodicita, " +
146             "E.col AS pagine_colori, E.numpag AS numero_pag, V.isbn, editore.nome AS editore FROM " +
147             "volume AS V INNER JOIN edizione AS E ON V.edizione = E.prodotto INNER JOIN serie AS S ON E.serie = S.codserie " +
148             "INNER JOIN editore ON E.editore = piva WHERE isbn = '1522583690251';";
149
150         try {
151           queryPS = connessione.prepareStatement(query2);
152           verifyQuery = connessione.prepareStatement(verifyQuery2);
153         } catch (SQLException e) {
154           connectSQLExceptionHandler(e);
155           System.exit(-1);
156         }
157
158         try {
159           System.out.println("Inserimento a catalogo in corso...");
160           queryPS.execute();
161
162           ResultSet verifyQueryAfter = verifyQuery.executeQuery();
163           System.out.println("Dopo l'inserimento:");
164           while(verifyQueryAfter.next()) {
165             System.out.println(verifyQueryAfter.getString(1) + " " + verifyQueryAfter.getInt(2) + ", E " + verifyQueryAfter.getDouble(3)
```

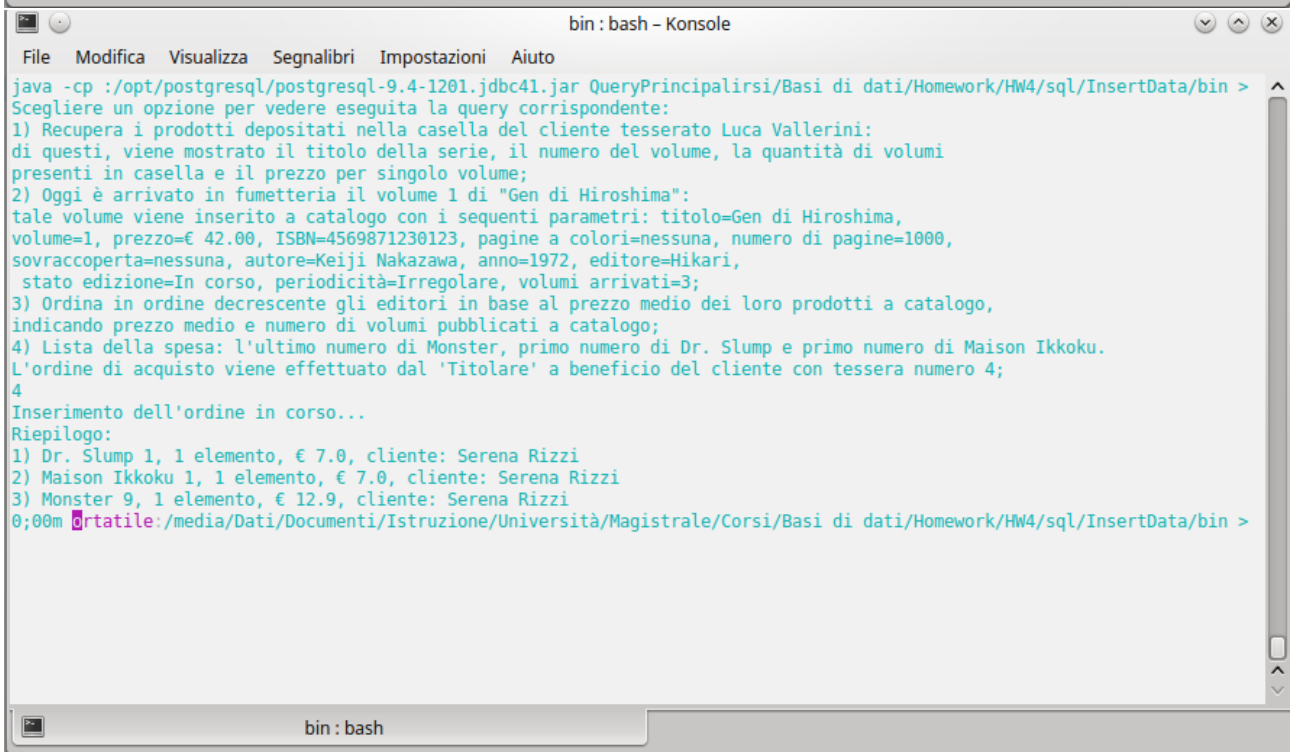
```
                +
                ",␣" + verifyQueryAfter.getInt(4) + "␣prodotti,\nperiodicita':␣" + verifyQueryAfter.getString(5) + ",␣pagine␣a␣colori:␣" +
                verifyQueryAfter.getBoolean(6) + ",␣numero␣di␣pagine:␣" + verifyQueryAfter.getInt(7) + ",\nISBN␣" + verifyQueryAfter.getString(8) +
                ",␣edito␣da␣" + verifyQueryAfter.getString(9));
            }
        } catch (SQLException e) {
            insertSQLExceptionHandler(e);
        }
        break;

    case 3:
        String query3 = "SELECT␣editore.nome␣AS␣editore,␣COUNT(volume.numvol)␣AS␣numero_vol_pubblicati,␣AVG(prezzofin)␣AS␣prezzo_medio␣FROM␣" +
            "editore␣INNER␣JOIN␣edizione␣ON␣editore.piva␣=␣edizione.editore␣INNER␣JOIN␣volume␣ON␣edizione.prodotto␣=␣volume.edizione␣" +
            "GROUP␣BY␣editore.nome␣ORDER␣BY␣prezzo_medio␣DESC";
        try {
            queryPS = connessione.prepareStatement(query3);
        } catch (SQLException e) {
            connectSQLExceptionHandler(e);
            System.exit(-1);
        }

        try {
            queryRS = queryPS.executeQuery();

            System.out.println("Classifica␣degli␣editori␣per␣prezzo␣medio␣di␣vendita:");
            int i = 0;
            while (queryRS.next()) {
                System.out.println(++i + ")␣" + queryRS.getString(1) + ",␣" + queryRS.getInt(2)+ "␣pubblicazioni,␣prezzo␣medio␣di␣E␣" + queryRS.getDouble(3));
            }
        } catch (SQLException e) {
            insertSQLExceptionHandler(e);
        }
        break;

    case 4:
        String query4 = "DROP␣TABLE␣IF␣EXISTS␣prezzo_totale,␣prezzo_elem_1,␣prezzo_elem_2,␣prezzo_elem_3,␣prodotto_1,␣prodotto_2,␣prodotto_3,␣sconto_tessera,␣codice_ordine_ult;" +
            "SELECT␣SUM(V.prezzofin)-SUM(V.prezzofin)*((SELECT␣sconto␣FROM␣tesseramento␣INNER␣JOIN␣tessera␣ON␣tesseramento.tessera␣=␣tessera.codtess␣WHERE␣codtess␣=␣4))/100␣" +
            "INTO␣prezzo_totale␣FROM␣serie␣AS␣S␣INNER␣JOIN␣edizione␣AS␣E␣ON␣S.codserie␣=␣E.serie␣INNER␣JOIN␣volume␣as␣V␣ON␣V.edizione␣=␣E.prodotto␣␣" +
            "WHERE␣(S.titolo␣=␣'Dr.␣Slump'␣AND␣V.numvol␣=␣1)␣OR␣(S.titolo␣=␣'Maison␣Ikkoku'␣AND␣V.numvol␣=␣1)␣OR␣(S.titolo␣=␣'Monster'␣AND␣V.numvol␣=␣9);␣" +
            "SELECT␣V.prezzofin␣INTO␣prezzo_elem_1␣FROM␣serie␣AS␣S␣INNER␣JOIN␣edizione␣AS␣E␣ON␣S.codserie␣=␣E.serie␣INNER␣JOIN␣volume␣as␣V␣ON␣V.edizione␣=␣E.prodotto␣" +
            "WHERE␣(S.titolo␣=␣'Dr.␣Slump'␣AND␣V.numvol␣=␣1);␣" +
            "SELECT␣V.prodotto␣INTO␣prodotto_1␣FROM␣serie␣AS␣S␣INNER␣JOIN␣edizione␣AS␣E␣ON␣S.codserie␣=␣E.serie␣INNER␣JOIN␣volume␣as␣V␣ON␣V.edizione␣=␣E.prodotto␣" +
            "WHERE␣(S.titolo␣=␣'Dr.␣Slump'␣AND␣V.numvol␣=␣1);␣" +
            "SELECT␣V.prezzofin␣INTO␣prezzo_elem_2␣FROM␣serie␣AS␣S␣INNER␣JOIN␣edizione␣AS␣E␣ON␣S.codserie␣=␣E.serie␣INNER␣JOIN␣volume␣as␣V␣ON␣V.edizione␣=␣E.prodotto␣" +
            "WHERE␣(S.titolo␣=␣'Maison␣Ikkoku'␣AND␣V.numvol␣=␣1);␣" +
            "SELECT␣V.prodotto␣INTO␣prodotto_2␣FROM␣serie␣AS␣S␣INNER␣JOIN␣edizione␣AS␣E␣ON␣S.codserie␣=␣E.serie␣INNER␣JOIN␣volume␣as␣V␣ON␣V.edizione␣=␣E.prodotto␣" +
            "WHERE␣(S.titolo␣=␣'Maison␣Ikkoku'␣AND␣V.numvol␣=␣1);␣" +
            "SELECT␣V.prezzofin␣INTO␣prezzo_elem_3␣FROM␣serie␣AS␣S␣INNER␣JOIN␣edizione␣AS␣E␣ON␣S.codserie␣=␣E.serie␣INNER␣JOIN␣volume␣as␣V␣ON␣V.edizione␣=␣E.prodotto␣" +
            "WHERE␣(S.titolo␣=␣'Monster'␣AND␣V.numvol␣=␣9);␣" +
            "SELECT␣V.prodotto␣INTO␣prodotto_3␣FROM␣serie␣AS␣S␣INNER␣JOIN␣edizione␣AS␣E␣ON␣S.codserie␣=␣E.serie␣INNER␣JOIN␣volume␣as␣V␣ON␣V.edizione␣=␣E.prodotto␣" +
            "WHERE␣(S.titolo␣=␣'Monster'␣AND␣V.numvol␣=␣9);␣␣" +
            "SELECT␣sconto␣INTO␣sconto_tessera␣FROM␣tesseramento␣INNER␣JOIN␣tessera␣ON␣tesseramento.tessera␣=␣tessera.codtess␣WHERE␣codtess␣=␣4;␣" +
            "INSERT␣INTO␣ordine␣(statoord,␣datains,␣prezzotot,␣tipoord,␣tesserato)␣VALUES␣('Evaso',␣'2015-12-18',␣(SELECT␣*␣FROM␣prezzo_totale),␣'Acquisto',␣(SELECT␣tesserato␣FROM␣tesseramento␣WHERE␣tessera␣=␣4));␣" +
            "SELECT␣MAX(codord)␣INTO␣codice_ordine_ult␣FROM␣ordine;␣" +
            "INSERT␣INTO␣merce␣(ordine,␣prodotto,␣quantita,␣prezzoelem)␣VALUES␣((SELECT␣*␣FROM␣codice_ordine_ult),␣(SELECT␣*␣FROM␣prodotto_1),␣1,␣(SELECT␣*␣FROM␣prezzo_elem_1));␣" +
            "INSERT␣INTO␣merce␣(ordine,␣prodotto,␣quantita,␣prezzoelem)␣VALUES␣((SELECT␣*␣FROM␣codice_ordine_ult),␣(SELECT␣*␣FROM␣prodotto_2),␣1,␣(SELECT␣*␣FROM␣prezzo_elem_2));␣" +
            "INSERT␣INTO␣merce␣(ordine,␣prodotto,␣quantita,␣prezzoelem)␣VALUES␣((SELECT␣*␣FROM␣codice_ordine_ult),␣(SELECT␣*␣FROM␣prodotto_3),␣1,␣(SELECT␣*␣FROM␣prezzo_elem_3));␣" +
            "INSERT␣INTO␣lavorazione␣(ordine,␣acc,␣datalav)␣VALUES␣((SELECT␣*␣FROM␣codice_ordine_ult),␣(SELECT␣codacc␣FROM␣account␣WHERE␣username␣=␣'Titolare'),␣'2015-12-18');␣" +
            "DROP␣TABLE␣IF␣EXISTS␣prezzo_totale,␣prezzo_elem_1,␣prezzo_elem_2,␣prezzo_elem_3,␣prodotto_1,␣prodotto_2,␣prodotto_3,␣sconto_tessera,␣codice_ordine_ult;";

        try {
            queryPS = connessione.prepareStatement(query4);
        } catch (SQLException e) {
            connectSQLExceptionHandler(e);
            System.exit(-1);
        }

        try {
            System.out.println("Inserimento␣dell'ordine␣in␣corso...");
            queryPS.execute();

            System.out.println("Riepilogo:");
            String verifyQuery4 = "SELECT␣serie.titolo,␣volume.numvol,␣merce.quantita,␣volume.prezzofin,␣tesserato.nome␣||␣'␣'␣||␣tesserato.cognome␣AS␣cliente␣" +
                "FROM␣ordine␣INNER␣JOIN␣merce␣ON␣ordine.codord␣=␣merce.ordine␣INNER␣JOIN␣lavorazione␣ON␣lavorazione.ordine␣=␣ordine.codord␣" +
                "INNER␣JOIN␣volume␣ON␣volume.prodotto␣=␣merce.prodotto␣" +
                "INNER␣JOIN␣tesserato␣ON␣ordine.tesserato␣=␣tesserato.codicefiscale␣" +
                "INNER␣JOIN␣edizione␣ON␣volume.edizione␣=␣edizione.prodotto␣INNER␣JOIN␣serie␣ON␣edizione.serie␣=␣serie.codserie␣WHERE␣codord␣=␣(SELECT␣MAX(codord)␣FROM␣ordine);";

            verifyQuery = connessione.prepareStatement(verifyQuery4);
```

```
245           ResultSet verifyQueryRS = verifyQuery.executeQuery();
246           int i = 0;
247           while (verifyQueryRS.next()) {
248             System.out.println(++i + ") " + verifyQueryRS.getString(1) + " " + verifyQueryRS.getInt(2)+ ", " +
249                 verifyQueryRS.getInt(3) + " elemento, E " + verifyQueryRS.getDouble(4) + ", cliente: " + verifyQueryRS.getString(5));
250           }
251         } catch (SQLException e) {
252           insertSQLExceptionHandler(e);
253         }
254         break;
255
256       default:
257         System.out.println("La scelta non e' valida. Chiusura inaspettata del programma.");
258         System.exit(-1);
259         break;
260       }
261
262     }
263
264 }
```






bin : bash – Konsole
File Modifica Visualizza Segnalibri Impostazioni Aiuto
java -cp :/opt/postgresql/postgresql-9.4-1201.jdbc41.jar QueryPrincipalirsi/Basi di dati/Homework/HW4/sql/InsertData/bin >
Scegliere un opzione per vedere eseguita la query corrispondente:
1) Recupera i prodotti depositati nella casella del cliente tesserato Luca Vallerini:
di questi, viene mostrato il titolo della serie, il numero del volume, la quantità di volumi
presenti in casella e il prezzo per singolo volume;
2) Oggi è arrivato in fumetteria il volume 1 di "Gen di Hiroshima":
tale volume viene inserito a catalogo con i sequenti parametri: titolo=Gen di Hiroshima,
volume=1, prezzo=€ 42.00, ISBN=4569871230123, pagine a colori=nessuna, numero di pagine=1000,
sovraccoperta=nessuna, autore=Keiji Nakazawa, anno=1972, editore=Hikari,
stato edizione=In corso, periodicità=Irregolare, volumi arrivati=3;
3) Ordina in ordine decrescente gli editori in base al prezzo medio dei loro prodotti a catalogo,
indicando prezzo medio e numero di volumi pubblicati a catalogo;
4) Lista della spesa: l'ultimo numero di Monster, primo numero di Dr. Slump e primo numero di Maison Ikkoku.
L'ordine di acquisto viene effettuato dal 'Titolare' a beneficio del cliente con tessera numero 4;
3
Classifica degli editori per prezzo medio di vendita:
1) Yamato Video, 1 pubblicazioni, prezzo medio di € 149.9
2) Hikari - 001 Edizioni, 2 pubblicazioni, prezzo medio di € 42.0
3) Dynit, 2 pubblicazioni, prezzo medio di € 24.9
4) Lucky Red, 1 pubblicazioni, prezzo medio di € 17.9
5) Planet Manga, 51 pubblicazioni, prezzo medio di € 9.66470588235294
6) J-Pop, 15 pubblicazioni, prezzo medio di € 5.9
7) Star Comics, 73 pubblicazioni, prezzo medio di € 3.9123287671232876
0;00m ortatile:/media/Dati/Documenti/Istruzione/Università/Magistrale/Corsi/Basi di dati/Homework/HW4/sql/InsertData/bin >
bin : bash


bin : bash – Konsole
File Modifica Visualizza Segnalibri Impostazioni Aiuto
java -cp :/opt/postgresql/postgresql-9.4-1201.jdbc41.jar QueryPrincipalirsi/Basi di dati/Homework/HW4/sql/InsertData/bin >
Scegliere un opzione per vedere eseguita la query corrispondente:
1) Recupera i prodotti depositati nella casella del cliente tesserato Luca Vallerini:
di questi, viene mostrato il titolo della serie, il numero del volume, la quantità di volumi
presenti in casella e il prezzo per singolo volume;
2) Oggi è arrivato in fumetteria il volume 1 di "Gen di Hiroshima":
tale volume viene inserito a catalogo con i sequenti parametri: titolo=Gen di Hiroshima,
volume=1, prezzo=€ 42.00, ISBN=4569871230123, pagine a colori=nessuna, numero di pagine=1000,
sovraccoperta=nessuna, autore=Keiji Nakazawa, anno=1972, editore=Hikari,
stato edizione=In corso, periodicità=Irregolare, volumi arrivati=3;
3) Ordina in ordine decrescente gli editori in base al prezzo medio dei loro prodotti a catalogo,
indicando prezzo medio e numero di volumi pubblicati a catalogo;
4) Lista della spesa: l'ultimo numero di Monster, primo numero di Dr. Slump e primo numero di Maison Ikkoku.
L'ordine di acquisto viene effettuato dal 'Titolare' a beneficio del cliente con tessera numero 4;
4
Inserimento dell'ordine in corso...
Riepilogo:
1) Dr. Slump 1, 1 elemento, € 7.0, cliente: Serena Rizzi
2) Maison Ikkoku 1, 1 elemento, € 7.0, cliente: Serena Rizzi
3) Monster 9, 1 elemento, € 12.9, cliente: Serena Rizzi
0;00m ortatile:/media/Dati/Documenti/Istruzione/Università/Magistrale/Corsi/Basi di dati/Homework/HW4/sql/InsertData/bin >
bin : bash