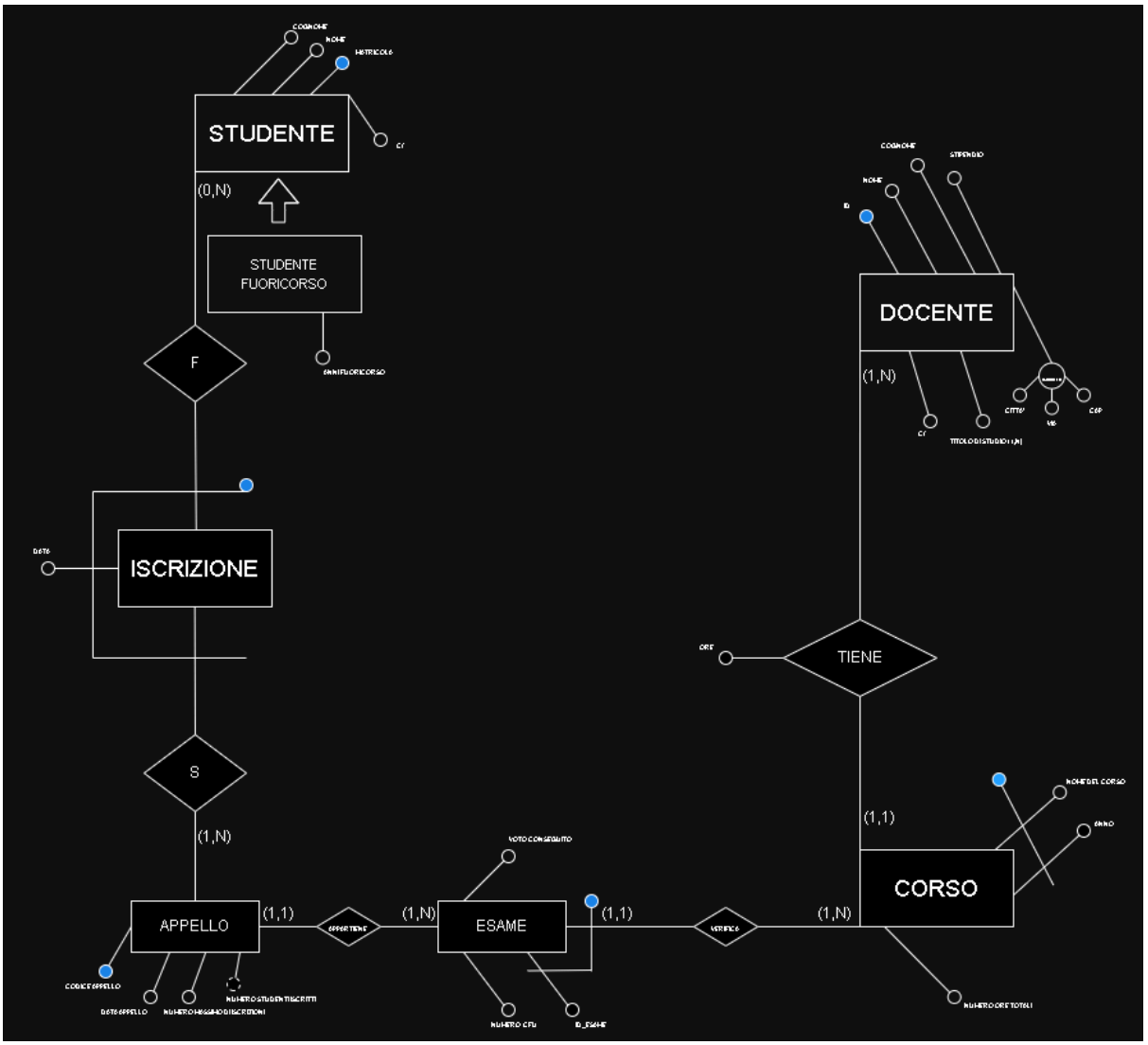
TRACCIA

L'università di Harvard del Massachusetts vuole gestire le iscrizioni agli appelli dei vari studenti. Uno studente è identificato da un numero di matricola ed è costituito dal nome,cognome,codice fiscale.Uno studente può essere fuoricorso e in questo caso si tiene traccia degli anni fuoricorso. Uno studente può iscriversi a vari appelli anche contemporaneamente. Un appello ha vari studenti iscritti, è identificato da un codice e caratterizzato,dalla data di appello,il numero massimo di iscrizioni,il numero di iscritti e l'esame da sostenere. Un appello può fare riferimento ad un solo esame mentre un esame fa riferimento a più appelli. Un esame è identificato dall'ID,dal nome del corso a cui fa riferimento e l'anno del corso ed è caratterizzato dal voto conseguito, e dal numero di cfu. Un corso è identificato dal nome, l'anno e caratterizzato dall'ID del docente e ha un numero di ore totali. Un docente è identificato dall'ID e caratterizzato dal nome , il cognome , il codice fiscale , il titolo di studio,l'indirizzo composto dalla via, il cap e la città e lo stipendio. Ogni docente può tenere vari corsi , mentre un corso fa riferimento ad un solo docente.

SCHEMA CONCETTUALE



BUSINESS RULES:

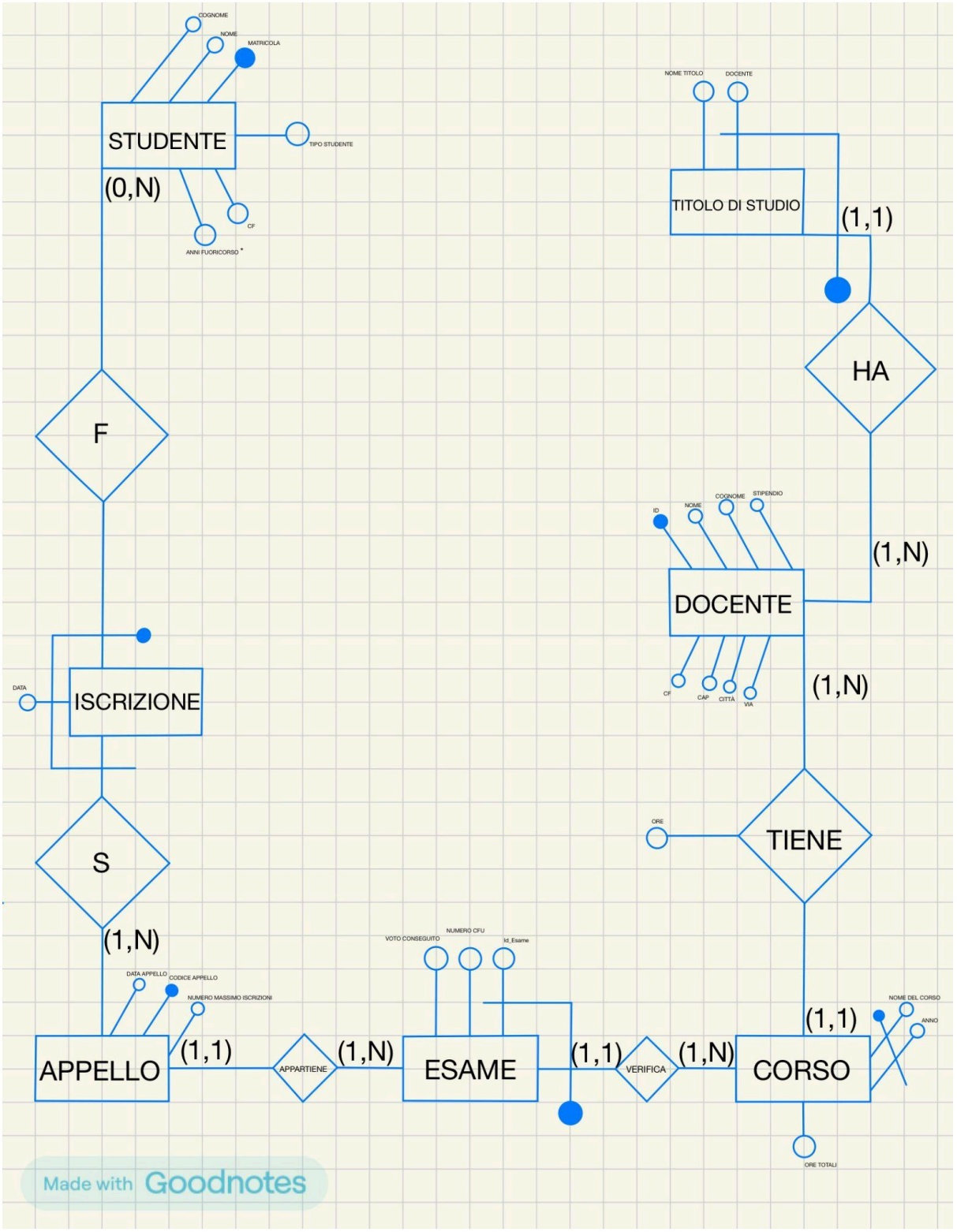
1)Regola di derivazione : L'attributo 'NUMERO DI STUDENTI ISCRITTI' dell'entità 'APPELLO' si ottiene contando gli studenti che si sono iscritti all'appello.

2)Regola di aggiornamento automatico : L'attributo 'NUMERO STUDENTI ISCRITTI' dell'entità 'APPELLO' deve essere aggiornato automaticamente ogni volta che uno studente si iscrive o cancella all'appello.

3)Validità degli studenti iscritti : solo gli studenti validi e ammessi dovrebbero essere considerati nel numero degli iscritti.

4)Integrità dei dati : Assicurarsi che il numero di studenti iscritti sia consistente con i dati effettivi dell'appello(es:numero iscritti massimo) e non vi siano discrepanze o errori nei dati.

SCHEMA CONCETTUALE RISTRUTTURATO



SCHEMA LOGICO

STUDENTE(MATRICOLA , NOME , COGNOME , CF , TIPO STUDENTE , ANNI FUORICORSO *)
ISCRIZIONE(DATA , *MATRICOLA STUDENTE* , *CODICE APPELLO*)
APPELLO(CODICE APPELLO , *ESAME* , DATA APPELLO , NUMERO MASSIMO ISCRIZIONI)
ESAME(IdESAME , *CORSO* , NUMERO CFU , VOTO CONSEGUITO)
CORSO(NOME DEL CORSO , ANNO , ID DOCENTE , ORE , ORE TOTALI)
DOCENTE(ID , NOME , COGNOME , CF , CITTA' , VIA , CAP,STIPENDIO)
TITOLO DI STUDIO(*DOCENTE* , NOME TITOLO)

ISCRIZIONE(MATRICOLA STUDENTE) V.I.R. STUDENTE(MATRICOLA)
ISCRIZIONE(CODICE APPELLO) V.I.R. APPELLO(CODICE APPELLO)
APPELLO(IdESAME,CORSO) V.I.R ESAME(IdESAME,CORSO)
ESAME(NOME DEL CORSO , ANNO) V.I.R CORSO(NOME DEL CORSO , ANNO)
CORSO(ID DOCENTE) V.I.R DOCENTE(ID)
TITOLO DI STUDIO(DOCENTE) V.I.R DOCENTE(ID)

QUERY

CREAZIONE E POPOLAZIONE DELLE TABELLE:

```
CREATE DATABASE Università;
USE Università;

CREATE TABLE Studente(
Matricola int not null,
Nome varchar(20) not null,
Cognome varchar(20) not null,
Cf varchar(20) not null,
tipoStudente varchar(20) not null,
anniFuoricorso int ,
primary key(Matricola)
);

CREATE TABLE Docente(
Id int not null,
Nome varchar(20) not null,
Cognome varchar(20) not null,
Cf varchar(20) not null,
Citta varchar(20) not null,
Via varchar(20) not null,
Cap int not null,
Stipendio float not null,
primary key(Id));
```

```
CREATE TABLE Corso(
nomeDelCorso varchar(20) not null,
Anno int not null,
idDocente int not null,
Ore int not null,
oreTotali int not null,
primary key(nomeDelCorso,Anno),
foreign key(idDocente) REFERENCES Docente(Id)
);

CREATE TABLE Esame(
idEsame int not null,
nomeDelCorso varchar(20) not null,
annoCorso int not null,
numeroCfu int not null,
votoConseguito int not null,
primary key(idEsame,nomeDelCorso,annoCorso),
foreign key(nomeDelCorso,annoCorso) REFERENCES Corso(nomeDelCorso,Anno)
);

CREATE TABLE Appello(
codiceAppello int not null,
idEsame int not null,
nomeEsame varchar(20) not null,
annoEsame int not null,
dataAppello date not null,
numeroMassimoIscrizioni int not null,
primary key(codiceAppello),
foreign key(idEsame,nomeEsame,annoEsame) REFERENCES
Esame(idEsame,nomeDelCorso,annoCorso)
);

CREATE TABLE Iscrizione(
dataIscrizione date not null,
matricolaStudente int not null,
codiceAppello int not null,
primary key(dataIscrizione),
foreign key(matricolaStudente) REFERENCES Studente(Matricola),
foreign key(codiceAppello) REFERENCES Appello(codiceAppello)
);

CREATE TABLE TitoloDiStudio(
Docente int not null,
nomeTitolo varchar(20) not null,
primary key(Docente),
foreign key(Docente) REFERENCES Docente(Id)
);
```

```
INSERT INTO Studente (Matricola,Nome,Cognome,Cf,tipoStudente,anniFuoricorso)
VALUES
(1, 'Mario', 'Rossi', 'RSSMRA01M01H501Z', 'Regolare',0),
(2, 'Luca', 'Verdi', 'VRDLCA02M02H502Y', 'Fuori Corso', 2),
(3, 'Anna', 'Bianchi', 'BNCHAN03M03H503X', 'Regolare', 0);

INSERT INTO Docente(Id,Nome,Cognome,Cf,Citta,Via,Cap,Stipendio)
VALUES
(201, 'Giuseppe', 'Conte', 'CNTGPP01M01H501A', 'Roma', 'Via Roma 123', 00100,1800),
(202, 'Maria', 'Russo', 'RSSMRA02M02H502B', 'Milano', 'Via Milano 456', 20100,2000),
(203, 'Luigi', 'Ferrari', 'FRRLGI03M03H503C', 'Napoli', 'Via Napoli 789', 80100,4000.50);

INSERT INTO Corso(nomeDelCorso,Anno,idDocente,Ore,oreTotali)
VALUES
('Matematica', 2024 , 201, 40, 120),
('Fisica', 2024, 202, 30, 90),
('Chimica',2024, 203, 50, 150);

INSERT INTO Esame(idEsame,nomeDelCorso,annoCorso,numeroCfu,votoConseguito)
VALUES
(10,'Matematica', 2024 , 10, 28),
(11,'Fisica', 2024 , 8 , 30),
(12,'Chimica', 2024 , 7 , 22);

INSERT INTO
Appello(codiceAppello,idEsame,nomeEsame,annoEsame,dataAppello,numeroMassimoIscri
zioni)
VALUES
(101,10, 'Matematica', 2024 , '2024-03-01', 50),
(102,11, 'Fisica', 2024 ,'2024-03-02', 40),
(103,12, 'Chimica', 2024 ,'2024-03-03', 30);

INSERT INTO Iscrizione(dataIscrizione,matricolaStudente,codiceAppello)
VALUES
('2024-02-23', 1, 101),
('2024-02-24', 2, 102),
('2024-02-25', 3, 103);

INSERT INTO TitoloDiStudio (Docente,nomeTitolo)
VALUES
(201, 'Laurea in Matematica'),
(202, 'Laurea in Fisica '),
(203, 'Laurea in Chimica ');
```

Elenco degli studenti regolari o fuori corso da meno di due anni in ordine di Cognome:

```
Select*
FROM Studente S
WHERE (S.tipoStudente = 'Fuori Corso' and S.anniFuoricorso < 2) or ( S.tipoStudente = 'Regolare')
order by S.Cognome;
```

Elenco degli studenti regolari iscritti ad un appello(Matricola,Nome,Cognome,Esame,Data):

```
Select S.Matricola, S.Nome,S.Cognome,A.nomeEsame,A.dataAppello
FROM Studente S JOIN Iscrizione I
on S.Matricola = I.matricolaStudente
JOIN Appello A
on I.codiceAppello = A.codiceAppello
WHERE S.tipoStudente = 'Regolare';
```

Somma degli stipendi di tutti i docenti (SommaStipendi):

```
Select sum(D.Stipendio) as SommaStipendi
FROM Docente D;
```

Somma degli stipendi dei docenti per anno del corso(AnnoDelCorso,SommaStipendi):

```
Select C.Anno as AnnoDelCorso, sum(D.Stipendio) as SommaStipendi
FROM Docente D JOIN Corso C
on D.Id = C.idDocente
group by C.Anno;
```

Somma degli stipendi dei docenti raggruppati per anno maggiore di 5000(AnnoDelCorso,SommaStipendi):

```
Select C.Anno as AnnoDelCorso, sum(D.Stipendio) as SommaStipendi
FROM Docente D JOIN Corso C
on D.Id = C.idDocente
group by C.Anno
FROM ContaStipendi
HAVING SommaStipendi >  5000;
```

L'Anno in cui la somma degli stipendi dei docenti che hanno tenuto corsi in quell'anno è massima(AnnoDelCorso , SommaStipendi):

```
CREATE VIEW ContaStipendi as
(
Select C.Anno as AnnoDelCorso, sum(D.Stipendio) as SommaStipendi
FROM Docente D JOIN Corso C
on D.Id = C.idDocente
group by C.Anno
);
Select *
FROM ContaStipendi
WHERE SommaStipendi = (
Select max(SommaStipendi)
FROM ContaStipendi
);
```

Il Docente la cui somma dei cfu dei suoi esami è massima:

```
CREATE VIEW ContaCfu as
(
Select D.Nome as NomeDocente, sum(E.numeroCfu) as SommaCfu
FROM Docente D JOIN Corso C
on D.Id = C.idDocente
JOIN Esame E
on E.nomeDelCorso = C.nomeDelCorso
group by D.Nome
);
Select *
FROM ContaCfu
WHERE SommaCfu = (
Select max(SommaCfu)
FROM ContaCfu
);
```

Unione tra esame e corso:

```
SELECT e.nomeDelCorso
FROM Esame e
UNION
SELECT c.nomeDelCorso
FROM Corso c;
```

Elenco degli studenti che si sono iscritti a tutti gli appelli di matematica:

```
SELECT *
FROM Studente s
WHERE NOT EXISTS (
  SELECT a.nomeEsame
  FROM Appello a
  WHERE a.nomeEsame = 'Matematica'
  AND NOT EXISTS (
    SELECT i.codiceAppello
    FROM Iscrizione i
    WHERE i.matricolaStudente = s.Matricola
      AND i.codiceAppello = a.codiceAppello
  )
);
```

DRIVER JDBC

```
import javax.swing.*;
import java.awt.*;
import java.sql.*;

public class ProgettoJDBC extends JFrame
{
   private Connection con = null;
   private final JTextArea outputTextArea;
   public ProgettoJDBC()
   {
      //Creo l'interfaccia grafica
      JFrame frame = new JFrame("Progetto JDBC GUI");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(600, 400);
      JPanel mainPanel = new JPanel(new BorderLayout());
      outputTextArea = new JTextArea();
      outputTextArea.setEditable(false);
      JScrollPane scrollPane = new JScrollPane(outputTextArea);
      mainPanel.add(scrollPane, BorderLayout.CENTER);
      JButton executeButton = new JButton("Esegui Operazioni");
      executeButton.addActionListener(e -> {
         try {
            executeOperation();
         } catch (SQLException ex) {
            throw new RuntimeException(ex);
         }
      });

      mainPanel.add(executeButton, BorderLayout.SOUTH);

      frame.getContentPane().add(mainPanel);
```

```
    frame.setVisible(true);

    //Collego il driver
    try
    {
        //Carico il driver
        Class.forName("com.mysql.jdbc.Driver");
    }
    catch(java.lang.ClassNotFoundException e)
    {
        System.err.print("ClassNotFoundException:"+e.getMessage());
    }

    //Stabilisco la connessione con il database
    try
    {
        String url = "jdbc:mysql://localhost:3306/Università";
        con = DriverManager.getConnection(url, "root", "Gabriel82@");
        outputTextArea.append("Connessione effettuata\n");
    }
    catch (SQLException ex)
    {
        outputTextArea.append("SQLException:" + ex.getMessage() + "\n");
    }
}

//Gestisco il database
public void executeOperation() throws SQLException
{
    try
    {
        outputTextArea.setText("");

        //Creo lo statement
        Statement st = con.createStatement();

        //Stampa gli studenti
        String sql = "SELECT* FROM Studente";
        ResultSet rs = st.executeQuery(sql);
        int i = 1;
        while(rs.next())
        {
            int matricola = rs.getInt("Matricola");
            String nome = rs.getString("Nome");
            String cognome = rs.getString("Cognome");
            String cf = rs.getString("Cf");
            String tipoStudente = rs.getString("tipoStudente");
            int anniFuoricorso = rs.getInt("anniFuoricorso");
```

```
            outputTextArea.append("\nMatricola Studente"+ i+": "+matricola);
            outputTextArea.append("\nNome Studente"+ i+": "+nome);
            outputTextArea.append("\nCognome Studente"+ i+": "+cognome);
            outputTextArea.append("\nCF Studente"+ i+": "+cf);
            outputTextArea.append("\nTipo Studente"+ i+": "+tipoStudente);
            outputTextArea.append("\nAnniFuoriCorso Studente"+ i+": "+anniFuoricorso+"\n");
            i++;
        }

        //Inserisce una nuova iscrizione
        int n = st.executeUpdate("INSERT INTO
Iscrizione(dataIscrizione,matricolaStudente,codiceAppello)"+"VALUES('2024-01-01',1,102)");
        if(n == 1)
            outputTextArea.append("\nInserimento effettuato");

        //Modifica l'iscrizione che rispetta le condizioni della clausola WHERE
        String updateString = "UPDATE Iscrizione " + "SET matricolaStudente = 3 " +
"WHERE matricolaStudente = 1 AND codiceAppello = 102";
        Statement st2 = con.createStatement();
        int resultUpdate = st2.executeUpdate(updateString);
        if(resultUpdate == 1)
            outputTextArea.append("\nAggiornamneto effettuato");

        //Stampa l'iscrizione che rispetta le condizioni della clausola WHERE
        String sqlQuery = "SELECT* FROM Iscrizione WHERE matricolaStudente = ? AND
codiceAppello = ?";
        int matricolaStudente = 3;
        int codiceAppello = 102;
        PreparedStatement ps = con.prepareStatement(sqlQuery);
        ps.setInt(1,matricolaStudente);
        ps.setInt(2,codiceAppello);
        ResultSet rs1 = ps.executeQuery();
        while(rs1.next())
        {
            Date dataIscrizione = rs1.getDate("dataIscrizione");
            int matricola = rs1.getInt("matricolaStudente");
            int codice = rs1.getInt("codiceAppello");
            outputTextArea.append("\nData Iscrizione:" + dataIscrizione);
            outputTextArea.append("\nMatricola Studente: " + matricola);
            outputTextArea.append("\nCodice Appello:" + codice + "\n");
        }

        //Cancella l'iscrizione che rispetta le condizioni della clausola WHERE
        String sql2 = "DELETE FROM Iscrizione WHERE matricolaStudente = 3 AND
codiceAppello = 102";
        int result = st.executeUpdate(sql2);
        if(result > 0)
            outputTextArea.append("\nCancellazione effettuata");
```

```
            else
                outputTextArea.append("\n cancellare il record");

            //Cancello tutti gli oggetti creati
            rs.close();
            rs1.close();
            st.close();
            st2.close();
            con.close();
        }
        catch(SQLException ex)
        {
            System.err.println("SQLException:" + ex.getMessage());
        }
        finally
        {
            if(con != null)
                con.close();
        }
    }

    public static void main(String[] args)
    {
        SwingUtilities.invokeLater(ProgettoJDBC::new);
    }
}
```