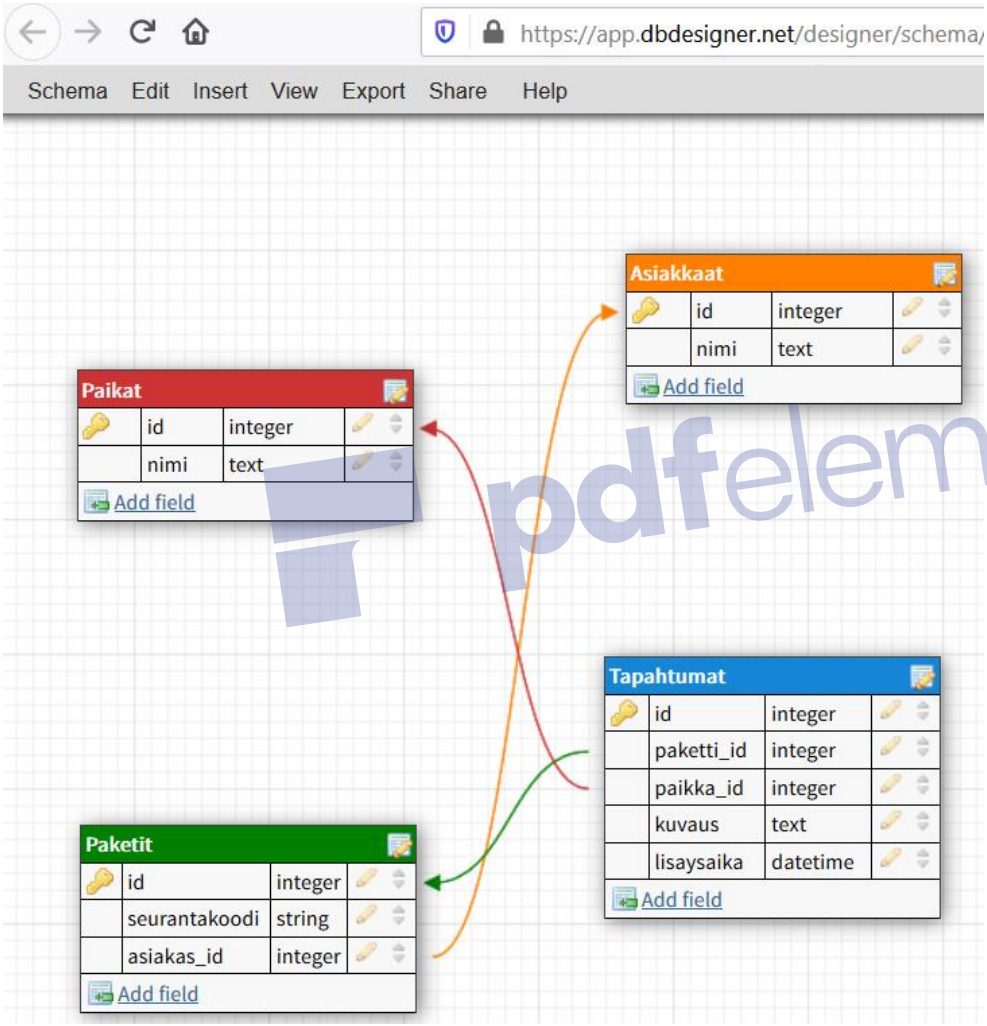
1. Harjoitustyön tekijän nimi, opiskelijanumero (jos tiedossa) ja mooc.fi-tunnus

Janne Piiroinen, 012395541, jpiiroin@hotmail.com

2. Selostus, mitkä toiminnot harjoitustyöhön on toteutettu

Toiminnot 1 - 6:
1. Luo sovelluksen tarvitsemat taulut tyhjään tietokantaan (tätä toimintoa voidaan käyttää, kun tietokantaa ei ole vielä olemassa).
2. Lisää uusi paikka tietokantaan, kun annetaan paikan nimi.
3. Lisää uusi asiakas tietokantaan, kun annetaan asiakkaan nimi.
4. Lisää uusi paketti tietokantaan, kun annetaan paketin seurantakoodi ja asiakkaan nimi. Asiakkaan tulee olla valmiiksi tietokannassa.
5. Lisää uusi tapahtuma tietokantaan, kun annetaan paketin seurantakoodi, tapahtuman paikka sekä kuvaus. Paketin ja paikan tulee olla valmiiksi tietokannassa.
6. Hae kaikki paketin tapahtumat seurantakoodin perusteella.

3. Tietokantakaavio



4. Ei tehty

5. Ei tehty
Törmäsin resultset closed -ongelmiin enkä päässyt niistä eteenpäin

6. Toteutetun sovelluksen lähdekoodi

Käyttöliittymä (Main.java):


```
import java.sql.SQLException;
import java.util.Scanner;

 public class Main {

    public static void main(String[] args) throws SQLException {

       Scanner lukija = new Scanner(System.in);

       Tietokanta kanta = new Tietokanta();

       while (true) {

          System.out.print("Toiminnot: [1]Luo kanta [2]Lisää paikka [3]Lisää asiakas [4]Lisää paketti"
                + " [5]Lisää tapahtuma [6]Näytä paketti"
                +"\n          [L]Lopeta [P]Poista kanta ja lopeta [N]Näytä kaikki"
                + "\n          Valitse toiminto (1-9): ");

          String komento = lukija.nextLine();

          if (komento.equals("L")) {
             break;
          }
          if (komento.equals("1")) {
             kanta.luoKanta();
          }
          if (komento.equals("2")) {
             System.out.print("Anna paikan nimi: ");
             String nimi = lukija.nextLine();
             kanta.luoPaikka(nimi);
          }
          if (komento.equals("3")) {
             System.out.print("Anna asiakkaan nimi: ");
             String nimi = lukija.nextLine();
             kanta.luoAsiakas(nimi);
          }
          if (komento.equals("4")) {
             System.out.print("Anna paketin seurantakoodi: ");
             String koodi = lukija.nextLine();
             System.out.print("Anna asiakkaan nimi: ");
             String nimi = lukija.nextLine();
             kanta.luoPaketti(koodi, nimi);
          }
          if (komento.equals("5")) {
             System.out.print("Anna paketin seurantakoodi: ");
             String koodi = lukija.nextLine();
             System.out.print("Anna tapahtuman paikka: ");
             String paikka = lukija.nextLine();
             System.out.print("Anna tapahtuman kuvaus: ");
             String kuvaus = lukija.nextLine();
             kanta.luoTapahtuma(koodi, paikka, kuvaus);
          }
          if (komento.equals("6")) {
             System.out.print("Anna paketin seurantakoodi: ");
             String koodi = lukija.nextLine();
             kanta.haePaketti(koodi);
          }
          if (komento.equals("P")) {
             kanta.poistaKanta();
             break;
          }
          if (komento.equals("N")) {
             kanta.naytaKanta();
          }
       }
    }
}
```

Toiminnallisuus (Tietokanta.java)


```
import java.sql.*;
import java.io.*;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class Tietokanta {

   Connection db = DriverManager.getConnection("jdbc:sqlite:testi.db");
   Statement s = db.createStatement();

   public Tietokanta() throws SQLException {

   }
   public void luoKanta() throws SQLException {
      s.execute("CREATE TABLE Paikat (id INTEGER PRIMARY KEY, nimi STRING)");
      s.execute("CREATE TABLE Paketit (id INTEGER PRIMARY KEY, seurantakoodi STRING, asiakas_id INTEGER)");
      s.execute("CREATE TABLE Asiakkaat (id INTEGER PRIMARY KEY, nimi STRING)");
      s.execute("CREATE TABLE Tapahtumat (id INTEGER PRIMARY KEY, paketti_id INTEGER, "
            + "paikka_id INTEGER, kuvaus STRING, lisaysaika STRING )");
      System.out.println("Tietokanta luotu");
   }
   public void naytaKanta() throws SQLException {
      ResultSet paikat = s.executeQuery("SELECT * FROM Paikat");
      System.out.println("Paikat:");
      while (paikat.next()) {
         System.out.println(paikat.getInt("id")+" "+paikat.getString("nimi"));
      }
      System.out.println("Asiakkaat:");
      ResultSet asiakkaat = s.executeQuery("SELECT * FROM Asiakkaat");
      while (asiakkaat.next()) {
         System.out.println(asiakkaat.getInt("id")+" "+asiakkaat.getString("nimi"));
      }
      System.out.println("Paketit:");
      ResultSet paketit = s.executeQuery("SELECT * FROM Paketit");
      while (paketit.next()) {
         System.out.println(paketit.getInt("id")+" "+paketit.getString("seurantakoodi")+" "+paketit.getString("asiakas_id"));
      }
      System.out.println("Tapahtumat:");
      ResultSet tapahtumat = s.executeQuery("SELECT * FROM Tapahtumat");
      while (tapahtumat.next()) {
         System.out.println(tapahtumat.getInt("id")+" "+tapahtumat.getString("paketti_id")
         +" "+tapahtumat.getString("paikka_id")
         +" "+tapahtumat.getString("kuvaus")+" "+tapahtumat.getString("lisaysaika"));
      }
   }
   public String naytaAika() {
      LocalDateTime myDateObj = LocalDateTime.now();
      DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd.M.yyyy HH:mm");
      String formattedDate = myDateObj.format(myFormatObj);
      return formattedDate;
   }
   public void poistaKanta() throws SQLException {
      s.close();
      db.close();
      new File("testi.db").delete();
      System.out.println("Tietokanta poistettu");
   }
```

```
public void luoPaikka(String nimi) throws SQLException {
        if (onkoOlemassa(nimi, "Paikat")) {
            System.out.println("VIRHE: Paikka on jo olemassa");
        } else {
            talletaKantaan(nimi, "Paikat");
            System.out.println("Paikka lisätty");
        }
    }
    public void luoAsiakas(String nimi) throws SQLException {
        if (onkoOlemassa(nimi, "Asiakkaat")) {
            System.out.println("VIRHE: Asiakas on jo olemassa");
        } else {
            talletaKantaan(nimi, "Asiakkaat");
            System.out.println("Asiakas lisätty");
        }
    }
    public void talletaKantaan(String nimi, String kanta) throws SQLException {
        PreparedStatement talleta = db.prepareStatement("INSERT INTO "+kanta+"(nimi) VALUES (?)");
        talleta.setString(1,nimi);
        talleta.executeUpdate();
    }
    public boolean onkoOlemassa (String nimi, String kanta) throws SQLException {
        PreparedStatement kysy = db.prepareStatement("SELECT nimi FROM "+kanta+" WHERE nimi=?");
        kysy.setString(1,nimi);
        ResultSet r = kysy.executeQuery();
        return r.next();
    }
    public void luoPaketti(String koodi, String nimi) throws SQLException {
        PreparedStatement kysy = db.prepareStatement("SELECT id FROM Asiakkaat WHERE nimi=?");
        kysy.setString(1,nimi);
        ResultSet r = kysy.executeQuery();
        int asiakas_id = r.getInt("id");

        PreparedStatement talleta = db.prepareStatement("INSERT INTO Paketit(seurantakoodi,asiakas_id) VALUES (?,?)");
        talleta.setString(1,koodi);
        talleta.setInt(2,asiakas_id);
        talleta.executeUpdate();
        System.out.println("Paketti lisätty");
    }
    public void luoTapahtuma(String seurantakoodi, String paikka, String kuvaus) throws SQLException {
        PreparedStatement kysykoodi = db.prepareStatement("SELECT id FROM Paketit WHERE seurantakoodi=?");
        kysykoodi.setString(1,seurantakoodi);
        ResultSet eka = kysykoodi.executeQuery();
        int paketti_id = eka.getInt("id");

        PreparedStatement kysypaikka = db.prepareStatement("SELECT id FROM Paikat WHERE nimi=?");
        kysypaikka.setString(1,paikka);
        ResultSet toka = kysypaikka.executeQuery();
        int paikka_id = toka.getInt("id");

        PreparedStatement talleta = db.prepareStatement("INSERT INTO Tapahtumat(paketti_id,paikka_id, "
                + "kuvaus, lisaysaika) VALUES (?,?,?,?)");
        talleta.setInt(1, paketti_id);
        talleta.setInt(2, paikka_id);
        talleta.setString(3, kuvaus);
        talleta.setString(4, naytaAika());
        talleta.executeUpdate();
    }
    public void haePaketti(String koodi) throws SQLException {
        ResultSet tapahtumat = s.executeQuery("SELECT T.lisaysaika, P.nimi, T.kuvaus "
                + "FROM Tapahtumat T, Paikat P, Paketit PT WHERE T.paketti_id = P.id "
                + "AND T.paikka_id = PT.id AND PT.seurantakoodi =" +koodi);
        while (tapahtumat.next()) {
            System.out.println(tapahtumat.getString("lisaysaika")+" "+tapahtumat.getString("nimi")
                    +" "+tapahtumat.getString("kuvaus"));
        }
    }
}
```