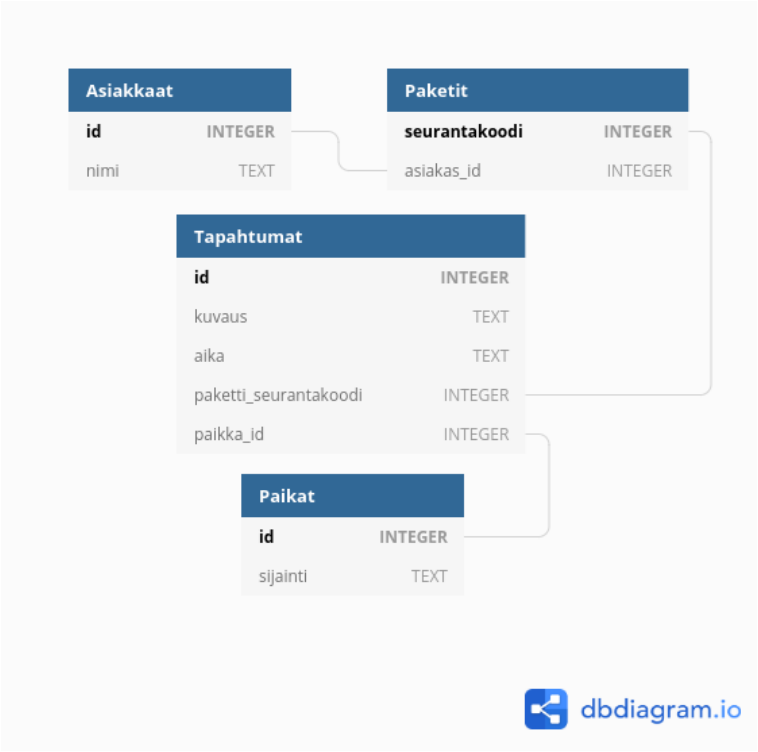
Tietokantojen perusteet lopputyö

Sebastian Coffeng
014731514
Mooc tunnus: SebastianCoffeng

29.2.2020

Harjoitustyön selostus

Harjoitustyössä oli tarkoitus luoda postipakettien hallintajärjestelmä. Järjestelmässä voidaan luoda asiakkaita, paketteja, paikkoja sekä tapahtumia. Jokainen paketti on yhdistettynä asiakkaaseen. Paketteihin yhdistetään paikka sekä tapahtuman kuvaus. Jokaisella asiakkaalla voi olla useampi paketti ja paketeilla useita tapahtumia, jotka ovat aikaleimattuja kirjaamishetkellä. Paketti täytyy luoda aina olemassa olevalle asiakkaalle ja paketilla saa esiintyä tapahtumia vain olemassa olevissa paikoissa. Kuvassa 1 nähdään miten SQL tietokanta toimii ja miten eri taulukot ovat kytkettynä toisiinsa.



Kuva 1: Tietokantakaavio

Harjoitustyössä toteutettiin kaikki tehtävänannon vaatimukset paitsi tehokkuustestiin vaadittava indeksointi. Ohjelman lähdekoodi löytyy raportin lopusta.

Alla nähdään SQL skeema.

```
-CREATE TABLE Asiakkaat (id INTEGER PRIMARY KEY, nimi TEXT);
-CREATE TABLE Paketit (seurantakoodi INTEGER PRIMARY KEY,
asiakas_id INTEGER));
-CREATE TABLE Paikat (id INTEGER PRIMARY KEY, sijainti TEXT);
-CREATE TABLE Tapahtumat (id INTEGER PRIMARY KEY,
kuvaus TEXT, paketti_seurantakoodi INTEGER, paikka_id INTEGER, ai-
ka TEXT);
```

Taulukoissa ei saa olla päälekkäistä tietoa asiakkaiden nimien eikä paikkojen suhteen. Myöskään useampaa seurantakoodia ei saa esiintyä järjestelmässä. Soveluksessa tämä on huomioitu tarkistamalla jokainen käyttäjän syöte ja vertailemalla sitä ei sallituihin arvoihin. Jos vertailussa todetaan, että syöte on jo olemassa taulukossa (tai syöte ei ole annettu halutussa muodossa) antaa sovellus virheviestin eikä ohjelma kirjaa mitään tietokantaan. Alla esimerkki kuinka ohjelman valikossa käsitellään ei toivottuja syötteitä.

```
if (!stringKoodi.matches("[0-9]+")){
          System.out.println("Syotteen tulee olla numerosarja");
} else {

}
```

Ohjelma vertailee onko syöte numero vai ei. Jos syöte ei ole numero tulostetaan virheviesti. Muuten ohjelma jatkaa suoritustaan ”else”lauseen sisälle. Vastaavan lailla syötteelle voidaan tehdä kysely tietokannan sisällä ja vertailla löytyykö syöte jo tietokannasta vertaillusta sarakkeesta. Alla olevassa esimerkissä tehdään kysely ja kyselyn tulosta vertaillaan. Jos kyselyn tuloksesta löytyy jo paikka tulostetaan virheviesti, muuten kirjataan taulukkoon uusi paikka.

```
if (r.next()) {
   System.out.println("Paikka on jo olemassa");
} else {
   PreparedStatement p2 = db.prepareStatement("INSERT INTO
      ↪ Paikat(sijainti) VALUES (?)");
   p2.setString(1,paikka);
   p2.executeUpdate();
   System.out.println("Paikka lisatty");
```

Ohjelman käyttö

Käyttäjällä on ohjelman valikossa kymmenen eri vaihtoehtoa, jotka valitaan numeroilla 0-9. Valikon vaihtoehdot ovat seuraavat:

1. Lisää asiakas
2. Lisää paikka
3. Lisää paketti
4. Lisää tapahtuma
5. Hae pakettien tapahtumat
6. Asiakkaan paketit
7. Anna paikkaan ja päivämäärän liittyvien tapahtumien määrä
8. Tehokkuustesti
9. :)
0. Sulje sovellus

Komennot 1-4 lisäävät SQL taulukkoihin tietoa ja 5-7 hakevat niistä tietoa. Valinta 8 suorittaa ohjelman tehokkuustestin ja 9 tulostaa hymiön. Syötteen ollessa jotain muuta kuin numero välillä 0-9 antaa ohjelma virheviestin "Syötteen tulee olla numero välillä 0-9".

Jos käyttäjä syöttää nimen, toiminnon tai paikan kenttään jossa vaaditaan numeroa, antaa ohjelma siitä virheimoituksen. Nimet, toiminnat sekä paikat voidaan kuitenkin nimetä numeroilla. Komento 5 hakee tietyn paketin kaikki tapahtumat ja ilmoittaa tapahtuman kirjaushetken. Komento 6 kertoo asiakkaan kaikki paketit sekä laskee kuinka monta tapahtumaa kullakin paketilla on. Komento 7 laskee tietyn päivän sekä paikan kaikki tapahtumat.

Tehokkuustesti

Tehokkuustestissä kokeillaan kuinka tehokas ohjelman suorittaminen on. SQL taulukkojen kasvaessa suuriksi alkaa tiedon hakeminen hidastumaan merkittävästi. Testissä luotiin tuhat asiakasta, tuhat paikkaa sekä jokaiselle asiakkaalle yksi paketti. Paketteja luotiin yhteensä siis tuhat. Tämän jälkeen jokaiselle paketille lisättiin tuhat tapahtumaa niin, että tapahtumia olisi yhteensä miljoona. Alla on ohjelman tulostus tehokkuustestistä.

1. Tuhannen asiakkaan lisäys suoritusaika millisekunneissa: 48
2. Tuhannen paikan lisäys suoritusaika millisekunneissa: 23
3. Tuhannen paketin lisäyksen jälkeen suoritusaika millisekunneissa: 22
4. Miljoonan tapahtuman lisäyksen jälkeen suoritusaika millisekunneissa: 2812
5. Tuhannen kyselyn jälkeen kulunut aika, jossa haetaan jokaisen asiakkaan pakettien määrä: 136
6. Suoritetaan tuhat kyselyä, joista jokaisessa haetaan jonkin paketin tapahtumien määrä: 108949

Kohdan 6 suorittamiseen menee huomattavan paljon aikaa. Voidaan siis todeta ettei ohjelma ole kovin tehokas käsittelemään suuria datamääriä.[1] Ohjelman tehokkuutta voitaisiin parantaa huomattavasti indeksoinnilla. Näin ohjelma voi hakea tehokkaasti, mistä hakuehdon täyttävä rivi löytyy.

Viitteet

[1] w3schools - SQL CREATE INDEX Statement - haettu 29.2.2020
https://www.w3schools.com/sql/sql_create_index.asp

Lähdekoodi

```
package tietokantaprojekti;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.sql.*;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

public class TietokantaProjekti {


private static void asiakkaanLisays(Connection db) throws
   ↪ SQLException {

      Scanner lukija = new Scanner(System.in);
      System.out.print("Anna asiakkaan nimi: ");
      String asiakas = lukija.nextLine();

      PreparedStatement p = db.prepareStatement("SELECT nimi
         ↪ FROM Asiakkaat WHERE nimi = ?");
      p.setString(1,asiakas);

      ResultSet r = p.executeQuery();

      if (r.next()) {
         System.out.println("Asiakas on jo olemassa");
      } else {
         PreparedStatement p2 = db.prepareStatement("INSERT INTO
            ↪ Asiakkaat(nimi) VALUES (?)");
         p2.setString(1,asiakas);
         p2.executeUpdate();
         System.out.println("Asiakas lisätty");
      }
   }

   private static void paikanLisays(Connection db) throws
      ↪ SQLException {
```

```
        Scanner lukija = new Scanner(System.in);
        System.out.print("Anna paikan nimi: ");
        String paikka = lukija.nextLine();

        PreparedStatement p = db.prepareStatement("SELECT sijainti
            ↪ FROM Paikat WHERE sijainti=?");
        p.setString(1,paikka);

        ResultSet r = p.executeQuery();

        if (r.next()) {
            System.out.println("Paikka on jo olemassa");
        } else {
            PreparedStatement p2 = db.prepareStatement("INSERT INTO
                ↪ Paikat(sijainti) VALUES (?)");
            p2.setString(1,paikka);
            p2.executeUpdate();
            System.out.println("Paikka lisätty");
        }

    }

    private static void paketinLisays(Connection db) throws
        ↪ SQLException {

        Scanner lukija = new Scanner(System.in);
        System.out.print("Anna paketin seurantakoodi: ");
        String skoodi = lukija.nextLine();

        if (!skoodi.matches("[0-9]+")){
            System.out.println("Syötteen tulee olla numerosarja");
        } else {
            int seurantakoodi = Integer.parseInt(skoodi);

            System.out.print("Anna asiakkaan nimi: ");
            String nimi = lukija.nextLine();

            PreparedStatement p1 = db.prepareStatement("SELECT *
                ↪ FROM Asiakkaat WHERE nimi = ?");
            PreparedStatement p2 = db.prepareStatement("SELECT *
                ↪ FROM Paketit WHERE seurantakoodi = ?");
```

```
            p1.setString(1, nimi);
            p2.setInt(1, seurantakoodi);

            ResultSet r2 = p2.executeQuery();
            ResultSet r1 = p1.executeQuery();

            if (r1.next()) {
                PreparedStatement p3 = db.prepareStatement("INSERT
                    ↪ INTO Paketit(seurantakoodi, asiakas_id) VALUES
                    ↪ (?,?)");
                p3.setInt(1,seurantakoodi);
                p3.setInt(2, r1.getInt("id"));

                if (r2.next()) {
                    System.out.println("Seurantakoodi on jo
                        ↪ olemassa");
                } else {
                    p3.executeUpdate();
                    System.out.println("Paketti lisätty");
                }

            } else {

                System.out.println("Asiakasta ei ole olemassa");
            }
        }
    }

    private static void tapahtumanLisays(Connection db) throws
        ↪ SQLException {

        SimpleDateFormat formatter = new
            ↪ SimpleDateFormat("dd.MM.yyyy HH:mm");
        Date date = new Date();

        Scanner lukija = new Scanner(System.in);
        System.out.print("Anna paketin seurantakoodi: ");
        String skoodi = lukija.nextLine();

        if (!skoodi.matches("[0-9]+")){
            System.out.println("Syötteen tulee olla numerosarja");
        } else {
```

```
        int seurantakoodi = Integer.parseInt(skoodi);

        System.out.print("Anna tapahtuman paikka: ");
        String paikka = lukija.nextLine();

        System.out.print("Anna tapahtuman kuvaus: ");
        String kuvaus = lukija.nextLine();

        PreparedStatement p1 = db.prepareStatement("SELECT *
           ↪ FROM Paketit, Paikat "
              + "WHERE Paketit.seurantakoodi = ? AND
                 ↪ Paikat.sijainti = ?");
        p1.setInt(1, seurantakoodi);
        p1.setString(2, paikka);

        ResultSet r = p1.executeQuery();

        if (r.next()) {
           PreparedStatement p2 = db.prepareStatement("INSERT
              ↪ INTO Tapahtumat(kuvaus, aika,
              ↪ paketti_seurantakoodi, paikka_id) VALUES
              ↪ (?,?,?,?)");
           p2.setString(1, kuvaus);
           p2.setString(2, formatter.format(date));
           p2.setInt(3, r.getInt("seurantakoodi"));
           p2.setInt(4, r.getInt("id"));
           p2.executeUpdate();
           System.out.println("Tapahtuma lisätty");
        } else {
           System.out.println("Seurantakoodia tai tapahtuman
              ↪ paikkaa ei ole olemassa");
        }
     }
  }

  private static void paketinTapahtumat(Connection db) throws
     ↪ SQLException {

     Scanner lukija = new Scanner(System.in);
     System.out.print("Anna paketin seurantakoodi: ");
     //int seurantakoodi = Integer.parseInt(lukija.nextLine());
     String skoodi = lukija.nextLine();
```

```
    if (!skoodi.matches("[0-9]+")){
       System.out.println("Syötteen tulee olla numerosarja");
    } else {
       int seurantakoodi = Integer.parseInt(skoodi);
       PreparedStatement p = db.prepareStatement("SELECT *
          ↪ FROM Tapahtumat, Paikat WHERE
          ↪ Tapahtumat.paikka_id = Paikat.id AND
          ↪ paketti_seurantakoodi = ?");
       p.setInt(1, seurantakoodi);
       ResultSet r = p.executeQuery();
       if (!r.next() ) {
          System.out.println("Seurantakoodia ei ole olemassa");
       } else {
          do {
             System.out.println(r.getString("aika") + ", " +
                ↪ r.getString("sijainti") + ", " +
                ↪ r.getString("kuvaus"));
          } while (r.next());
       }
    }
}

private static void asiakkaanPaketit (Connection db) throws
   ↪ SQLException {

    Scanner lukija = new Scanner(System.in);
    System.out.print("Anna asiakkaan nimi: ");
    String nimi = lukija.nextLine();

    PreparedStatement p1 = db.prepareStatement("SELECT
       ↪ COUNT(Tapahtumat.id), Paketit.seurantakoodi FROM
       ↪ Asiakkaat, Tapahtumat, Paketit "
          + "WHERE Paketit.asiakas_id = Asiakkaat.id AND
             ↪ Tapahtumat.paketti_seurantakoodi =
             ↪ Paketit.seurantakoodi AND Asiakkaat.nimi = ?"
          + "GROUP BY Paketit.seurantakoodi");
    p1.setString(1, nimi);

    ResultSet r = p1.executeQuery();

    if (!r.next() ) {
```

```
            System.out.println("Asiakasta ei ole olemassa");
        } else {
            do {
                System.out.println("Paketin seurantakoodi:
                    ↪ "+r.getInt("seurantakoodi") + ", tapahtumia: "
                    ↪ + r.getInt("COUNT(Tapahtumat.id)"));
            } while (r.next());
        }
    }

    private static void paikanTapahtumat(Connection db) throws
        ↪ SQLException {

        Scanner lukija = new Scanner(System.in);

        System.out.print("Anna paikan nimi: ");
        String paikka = lukija.nextLine();

        System.out.print("Anna päivämäärä (dd.MM.yyyy): ");
        String aika = lukija.nextLine();

        PreparedStatement p = db.prepareStatement("SELECT
            ↪ COUNT(Tapahtumat.id) FROM Paikat, Tapahtumat "
                + "WHERE Tapahtumat.paikka_id = Paikat.id AND
                    ↪ cast(Tapahtumat.aika as date) = cast(? as
                    ↪ date) AND Paikat.sijainti = ?");
        p.setString(1, aika);
        p.setString(2, paikka);

        ResultSet r = p.executeQuery();

        System.out.println("Tapahtumien määrä: " +
            ↪ r.getInt("COUNT(Tapahtumat.id)"));

    }

        private static void tehokkuusTesti (Connection db) throws
            ↪ SQLException {
        //1. tuhannen asiakkaan
            ↪ lisäys----------------------------------------
        long startTime = System.currentTimeMillis();
        Statement s = db.createStatement();
```

```
s.execute("BEGIN TRANSACTION");
PreparedStatement p = db.prepareStatement("INSERT INTO
   ↪ Asiakkaat (nimi) VALUES (?)");
for (int i = 1; i <= 1000; i++) {
   p.setInt(1,i);
   p.executeUpdate();
}
s.execute("COMMIT");
long endTime = System.currentTimeMillis();
long timeElapsed = endTime - startTime;
System.out.println("1. Tuhannen asiakkaan lisäys
   ↪ suoritusaika millisekunneissa: " + timeElapsed);

//2. Tuhannen paikan
   ↪ lisäys--------------------------------------
long startTime1 = System.currentTimeMillis();
Statement s1 = db.createStatement();
s1.execute("BEGIN TRANSACTION");
PreparedStatement p1 = db.prepareStatement("INSERT INTO
   ↪ Paikat (sijainti) VALUES (?)");
for (int i = 1; i <= 1000; i++) {
   p1.setInt(1,i);
   p1.executeUpdate();
}
s1.execute("COMMIT");
long endTime1 = System.currentTimeMillis();
long timeElapsed1 = endTime1 - startTime1;
System.out.println("2. Tuhannen paikan lisäys suoritusaika
   ↪ millisekunneissa: " + timeElapsed1);

//3. Jokaiselle asiakkaalle luodaan uusi
   ↪ paketti---------------------
long startTime2 = System.currentTimeMillis();
Statement s4 = db.createStatement();
s4.execute("BEGIN TRANSACTION");
PreparedStatement p4 = db.prepareStatement("INSERT INTO
   ↪ Paketit(seurantakoodi, asiakas_id) VALUES (?,?)");
for (int i = 1; i <= 1000; i++) {
   p4.setInt(1,i);
   p4.setInt(2, i);
   p4.executeUpdate();
}
```

```
s4.execute("COMMIT");
long endTime2 = System.currentTimeMillis();
long timeElapsed2 = endTime2 - startTime2;
System.out.println("3. Tuhannen paketin lisäyksen jälkeen
   ↪ suoritusaika millisekunneissa: " + timeElapsed2);

//4 .Suoritetaan miljoona tapahtumaa jokaiselle jokin
   ↪ paketti.-------------------------------------
long startTime3 = System.currentTimeMillis();

SimpleDateFormat formatter = new
   ↪ SimpleDateFormat("dd.MM.yyyy HH:mm:ss");
Date date = new Date();
Statement c = db.createStatement();
c.execute("BEGIN TRANSACTION");
PreparedStatement p8 = db.prepareStatement("INSERT INTO
   ↪ Tapahtumat(kuvaus, paketti_seurantakoodi, paikka_id,
   ↪ aika) VALUES (?,?,?,?)");
for (int i = 1; i <= 1000; i++) {
   String kuvaus = Integer.toString(i);
   for (int i2 = 1; i2 <= 1000; i2++) { // Tänne muuta
      ↪ kuinkamonta tapahtumaan luodaan
      String kuvaus2 = Integer.toString(i2);
      p8.setString(1, kuvaus2);
      p8.setInt(2, i);
      p8.setInt(3, i);
      p8.setString(4, formatter.format(date));
      p8.executeUpdate();
   }
}
s4.execute("COMMIT");
long endTime3 = System.currentTimeMillis();
long timeElapsed3 = endTime3 - startTime3;
System.out.println("4. Miljoonan tapahtuman lisäyksen
   ↪ jälkeen suoritusaika millisekunneissa: " +
   ↪ timeElapsed3);

//5. Suoritetaan tuhat kyselyä, joista jokaisessa haetaan
   ↪ jonkin asiakkaan pakettien määrä.
long startTimer5 = System.currentTimeMillis();
Statement c1 = db.createStatement();
c1.execute("BEGIN TRANSACTION");
```

```
PreparedStatement p9 = db.prepareStatement("SELECT
   ↪ COUNT(Paketit.seurantakoodi) FROM Asiakkaat, Paketit
   ↪ "
      + "WHERE Paketit.asiakas_id = Asiakkaat.id AND
         ↪ Asiakkaat.nimi = ?"
      + "GROUP BY Paketit.seurantakoodi");

for (int i5 = 1; i5 <= 1000; i5++) {
   String nimi = Integer.toString(i5);
   p9.setString(1, nimi);
   ResultSet r1 = p9.executeQuery();

}
c1.execute("COMMIT");
long endTime5 = System.currentTimeMillis();
long timeElapsed5 = endTime5 - startTimer5;
System.out.println("5. Tuhannen kyselyn jälkeen kulunut
   ↪ aika, jossa haetaan jokaisen asiakkaan pakettien
   ↪ määrä: " + timeElapsed5);


//6. Haetaan pakettien tapahtumien
   ↪ määrä-------------------------------------
long startTime4 = System.currentTimeMillis();
Statement c2 = db.createStatement();
c2.execute("BEGIN TRANSACTION");
PreparedStatement k = db.prepareStatement("SELECT
   ↪ COUNT(Tapahtumat.id), Paketit.seurantakoodi FROM
   ↪ Tapahtumat, Paketit "
      + "WHERE Tapahtumat.paketti_seurantakoodi =
         ↪ Paketit.seurantakoodi AND
         ↪ Paketit.seurantakoodi = ?"
      + "GROUP BY Paketit.seurantakoodi");
for (int i3 = 1; i3 <= 1000; i3++) {
   String seurantakoodi = Integer.toString(i3);
   k.setString(1, seurantakoodi);
   ResultSet l = k.executeQuery();
}
c2.execute("COMMIT");
long endTime4 = System.currentTimeMillis();
long timeElapsed4 = endTime4 - startTime4;
```

```
        System.out.println("6. Suoritetaan tuhat kyselyä, joista
            ↪ jokaisessa haetaan jonkin paketin tapahtumien määrä:
            ↪ " + timeElapsed4);




    }

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

            Scanner lukija = new Scanner(System.in);

            System.out.println("Tervetuloa
                ↪ paketinseurantasovellukseen");

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

            s.execute("CREATE TABLE Asiakkaat (id INTEGER PRIMARY
                ↪ KEY, nimi TEXT)");
            s.execute("CREATE TABLE Paketit (seurantakoodi INTEGER
                ↪ PRIMARY KEY, asiakas_id INTEGER, FOREIGN
                ↪ KEY(asiakas_id) REFERENCES Asiakas(id))");
            s.execute("CREATE TABLE Paikat (id INTEGER PRIMARY KEY,
                ↪ sijainti TEXT)");
            s.execute("CREATE TABLE Tapahtumat (id INTEGER PRIMARY
                ↪ KEY, kuvaus TEXT, paketti_seurantakoodi INTEGER,
                ↪ paikka_id INTEGER, aika TEXT,"
                    + "FOREIGN KEY(paketti_seurantakoodi) REFERENCES
                        ↪ Paketit(seurantakoodi),"
                    + "FOREIGN KEY(paikka_id) REFERENCES
                        ↪ Paikat(id))");

            System.out.println("1. Lisää asiakas");
            System.out.println("2. Lisää paikka");
            System.out.println("3. Lisää paketti");
            System.out.println("4. Lisää tapahtuma");
            System.out.println("5. Hae pakettien tapahtumat");
            System.out.println("6. Asiakkaan paketit");
```

```
        System.out.println("7. Anna paikkaan ja päivämäärän
           ↪ liittyvien tapahtumien määrä");
        System.out.println("8. Tehokkuustesti");
        System.out.println("9. :)");
        System.out.println("0. Sulje sovellus");

        while (true) {
           System.out.print("Valitse toiminto (0-9): ");
           String syote = lukija.nextLine(); //joo tiedän se on
              ↪ kirjoitettu väärin

           if (!syote.matches("[0-9]+") ||
              ↪ Double.parseDouble(syote) < 0 ||
              ↪ Double.parseDouble(syote) > 9 ){
              System.out.println("Syötteen tulee olla numero
                 ↪ välillä 0-9");
           }

           if (syote.equals("0")){
              break;
           }

           if (syote.equals("1")) {
              asiakkaanLisays(db);

           }

           if (syote.equals("2")) {
              paikanLisays(db);

           }

           if (syote.equals("3")) {
              paketinLisays(db);
           }

           if (syote.equals("4")) {
              tapahtumanLisays(db);
           }
           if (syote.equals("5")) {
              paketinTapahtumat(db);
           }
```

```
            if (syote.equals("6")) {
                asiakkaanPaketit(db);
            }
            if (syote.equals("7")) {
                paikanTapahtumat(db);

            }
            if (syote.equals("8")) {
                tehokkuusTesti(db);
            }
            if (syote.equals("9")) {
                System.out.println(":)");
            }

        }

    }
}
```