Tietokantojen perusteet, kevät 2020 - Harjoitustyö

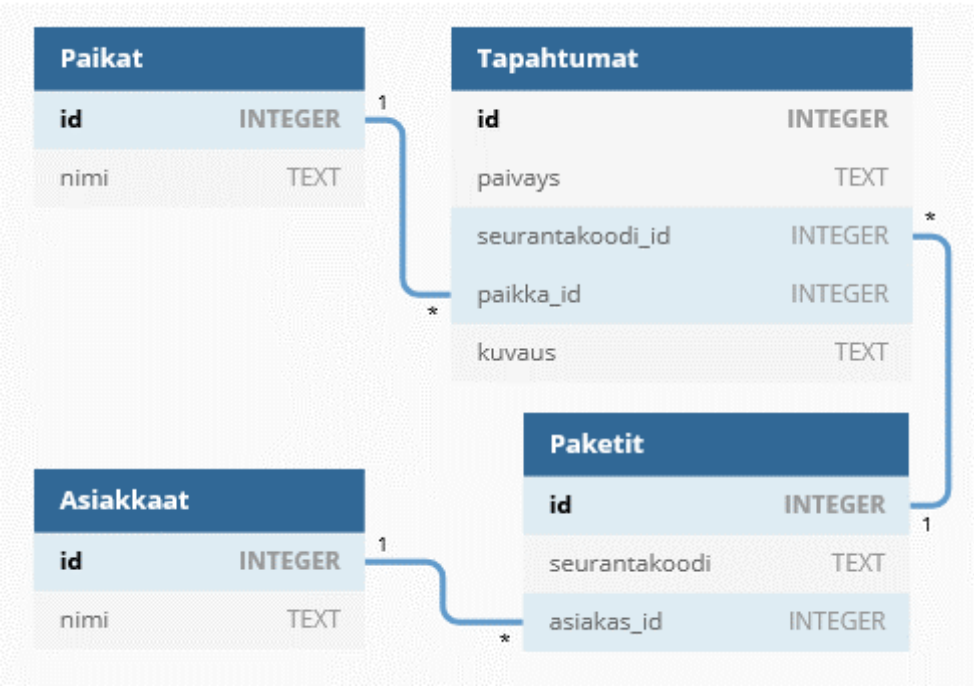
Harjoitustyön toteutus

Harjoitustyössä on toteutettu kaikki 9 toimintoa (alla), jotka olivat määritelty harjoitustyöhön Java-kieltä hyväksikäyttäen.

- Luo sovelluksen tarvitsemat taulut tyhjään tietokantaan (tätä toimintoa voidaan käyttää, kun tietokantaa ei ole vielä olemassa).
- Lisää uusi paikka tietokantaan, kun annetaan paikan nimi.
- Lisää uusi asiakas tietokantaan, kun annetaan asiakkaan nimi.
- Lisää uusi paketti tietokantaan, kun annetaan paketin seurantakoodi ja asiakkaan nimi. Asiakkaan tulee olla valmiiksi tietokannassa.
- Lisää uusi tapahtuma tietokantaan, kun annetaan paketin seurantakoodi, tapahtuman paikka sekä kuvaus. Paketin ja paikan tulee olla valmiiksi tietokannassa.
- Hae kaikki paketin tapahtumat seurantakoodin perusteella.
- Hae kaikki asiakkaan paketit ja niihin liittyvien tapahtumien määrä.
- Hae annetusta paikasta tapahtumien määrä tiettynä päivänä.
- Suorita tietokannan tehokkuustesti.

Tietokantakaavio ja SQL-skeema

Tietokantakaavio:



SQL-skeema:

- CREATE TABLE Paikat (id INTEGER PRIMARY KEY, nimi TEXT UNIQUE NOT NULL);
- CREATE TABLE Asiakkaat (id INTEGER PRIMARY KEY, nimi TEXT UNIQUE NOT NULL);

- CREATE TABLE Paketit (id INTEGER PRIMARY KEY, seurantakoodi TEXT UNIQUE NOT NULL, asiakas_id INTEGER NOT NULL REFERENCES Asiakkaat ON DELETE CASCADE);
- CREATE TABLE Tapahtumat (id INTEGER PRIMARY KEY, paivays TEXT, seurantakoodi_id INTEGER NOT NULL REFERENCES Paketit ON DELETE CASCADE, paikka_id INTEGER NOT NULL REFERENCES Paikat ON DELETE CASCADE, kuvaus TEXT NOT NULL);

Tehokkuustestin tulokset

Ilman indeksejä:
Tehokkuustestin 1 tulos: 0.0176613 s, kun lisättiin 1000 paikkaa
Tehokkuustestin 2 tulos: 0.0111946 s, kun lisättiin 1000 asiakasta
Tehokkuustestin 3 tulos: 0.0193926 s, kun lisättiin 1000 pakettia
Tehokkuustestin 4 tulos: 8.6664922 s, kun lisättiin 1000000 tapahtumaa
Tehokkuustestin 5 tulos: 0.0499959 s, kun suoritettiin 1000 kyselyä asiakkaan pakettimääristä
Tehokkuustestin 6 tulos: 78.8731832 s, kun suoritettiin 1000 kyselyä pakettien tapahtumamääristä

Indeksien lisäämisen jälkeen:
Tehokkuustestin 1 tulos: 0.0183096 s, kun lisättiin 1000 paikkaa
Tehokkuustestin 2 tulos: 0.0122793 s, kun lisättiin 1000 asiakasta
Tehokkuustestin 3 tulos: 0.0191614 s, kun lisättiin 1000 pakettia
Tehokkuustestin 4 tulos: 27.4155122 s, kun lisättiin 1000000 tapahtumaa
Tehokkuustestin 5 tulos: 0.0496778 s, kun suoritettiin 1000 kyselyä asiakkaan pakettimääristä
Tehokkuustestin 6 tulos: 0.0591278 s, kun suoritettiin 1000 kyselyä pakettien tapahtumamääristä

Tulkinta:
Tuloksista voidaan havaita se, että kun indeksi lisättiin ohjelmaan, niin viimeinen tehokkuustestin tulos nopeutui huomattavasti ja näin ollen voidaan todeta että indeksin lisääminen toi tehokkuutta kyseiseen kyselyyn. Kun taas tehokkuustesti 4 tuloksen aika moninkertaistui. Tämä johtuu luentomateriaalin mukaan siitä, että kun taulun sisältö muuttuu, niin muutos täytyy myös päivittää kaikkiin tauluun liittyviin indekseihin.

UNIQUE

Alla ovat koodit, jonka avulla on varmistettu, että jokaisella paikalla ja asiakkaalla on eri nimi ja jokaisella paketilla on eri seurantakoodi. Toisin sanoen jokainen arvo on UNIQUE (lihavoituna alla), jolloin ohjelma ei anna lisätä samanlaista arvoa, joten ohjelma ilmoittaa virheilmoituksella että kyseinen arvo on jo olemassa.

s.execute("CREATE TABLE Paikat (id INTEGER PRIMARY KEY, **nimi TEXT UNIQUE** NOT NULL)");
s.execute("CREATE TABLE Asiakkaat (id INTEGER PRIMARY KEY, **nimi TEXT UNIQUE** NOT NULL)");
s.execute("CREATE TABLE Paketit (id INTEGER PRIMARY KEY, **seurantakoodi TEXT UNIQUE** NOT NULL, asiakas_id INTEGER NOT NULL REFERENCES Asiakkaat ON DELETE CASCADE)");

Lähdekoodi

Koodi on toteutettu siten, että ohjelma ajetaan kayttoliittymaluokassa, joka kutsuu eri komennoilla tietokantaluokan metodeja. Alla siis sekä kayttoliittyma- että tietokantaluokka.

Kayttoliittymaluokka:

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

public class Kayttoliittyma {

    public static void main(String[] args) throws SQLException {
        Scanner input = new Scanner(System.in);
        Tietokanta t = new Tietokanta();
        System.out.println("Toiminnot:");
        System.out.println("0 - lopettaa");
        System.out.println("1 - luo tietokannan");
        System.out.println("2 - lisää paikan");
        System.out.println("3 - lisää asiakkaan");
        System.out.println("4 - lisää paketin");
        System.out.println("5 - lisää tapahtuman");
        System.out.println("6 - hakee paketin tapahtumat");
        System.out.println("7 - hakee asiakkaan paketit ja tapahtumamäärät");
        System.out.println("8 - hakee paikan tapahtumamäärät tiettynä päivämääränä");
        System.out.println("9 - ajaa tehokkuustestit (mahdollista tehdä vain kerran)");

        while(true) {
            System.out.print("Anna toiminto: ");
            String toiminto = input.nextLine();

            if(toiminto.equals("0")) {
                System.out.println("Hei Hei");
                break;
            }

            if(toiminto.equals("1")) {
                t.luonti();
            }

            if(toiminto.equals("2")) {
                System.out.print("Anna paikan nimi: ");
                String paikka = input.nextLine();
                t.paikka(paikka);
            }

            if(toiminto.equals("3")) {
                System.out.print("Anna asiakkaan nimi: ");
                String asiakas = input.nextLine();
                t.asiakas(asiakas);
            }

            if(toiminto.equals("4")) {
                System.out.print("Anna paketin seurantakoodi: ");
                String seurantakoodi = input.nextLine();
                System.out.print("Anna asiakkaan nimi: ");
                String asiakas = input.nextLine();
                t.paketti(seurantakoodi,asiakas);
            }

            if(toiminto.equals("5")) {
                System.out.print("Anna paketin seurantakoodi: ");
                String seurantakoodi = input.nextLine();
                System.out.print("Anna tapahtuman paikka: ");
                String paikka = input.nextLine();
                System.out.print("Anna tapahtuman kuvaus: ");
                String kuvaus = input.nextLine();
                t.tapahtuma(seurantakoodi,paikka,kuvaus);
            }
```

```
        if(toiminto.equals("6")) {
            System.out.print("Anna paketin seurantakoodi: ");
            String seurantakoodi = input.nextLine();
            t.haku(seurantakoodi);
        }

        if(toiminto.equals("7")) {
            System.out.print("Anna asiakkaan nimi: ");
            String animi = input.nextLine();
            t.asiakastapahtumat(animi);
        }

        if(toiminto.equals("8")) {
            System.out.print("Anna paikan nimi: ");
            String pnimi = input.nextLine();
            System.out.print("Anna paivays (muodossa pp.kk.vvvv): ");
            String paiva = input.nextLine();
            t.paikkatapahtumat(pnimi,paiva);
        }

        if(toiminto.equals("9")) {
            long alku1 = System.nanoTime();
            t.t1paikka();
            long loppu1 = System.nanoTime();
            System.out.println("Tehokkuustestin 1 tulos: "+((loppu1-alku1)/1e9)+" s, kun lisättiin 1000 paikkaa");

            long alku2 = System.nanoTime();
            t.t2asiakas();
            long loppu2 = System.nanoTime();
            System.out.println("Tehokkuustestin 2 tulos: "+((loppu2-alku2)/1e9)+" s, kun lisättiin 1000 asiakasta");

            long alku3 = System.nanoTime();
            t.t3paketti();
            long loppu3 = System.nanoTime();
            System.out.println("Tehokkuustestin 3 tulos: "+((loppu3-alku3)/1e9)+" s, kun lisättiin 1000 pakettia");

            long alku4 = System.nanoTime();
            t.t4tapahtuma();
            long loppu4 = System.nanoTime();
            System.out.println("Tehokkuustestin 4 tulos: "+((loppu4-alku4)/1e9)+" s, kun lisättiin 1000000 tapahtumaa");

            long alku5 = System.nanoTime();
            t.t5asiakaspaketit();
            long loppu5 = System.nanoTime();
            System.out.println("Tehokkuustestin 5 tulos: "+((loppu5-alku5)/1e9)+" s, kun suoritettiin 1000 kyselyä asiakkaan
pakettimääristä");

            long alku6 = System.nanoTime();
            t.t6pakettitapahtumat();
            long loppu6 = System.nanoTime();
            System.out.println("Tehokkuustestin 6 tulos: "+((loppu6-alku6)/1e9)+" s, kun suoritettiin 1000 kyselyä pakettien
tapahtumamääristä");
        }

        if(toiminto.equals("12")) {
        long alku5 = System.nanoTime();
            t.t6pakettitapahtumat();
            long loppu5 = System.nanoTime();
            System.out.println("Aikaa kului: "+((loppu5-alku5)/1e9)+" s, kun suoritettiin 1000 kyselyä asiakkaan pakettien määristä");
        }
      }
   }
}
```

Tietokantaluokka:

```
import java.sql.*;
import java.util.Scanner;
import static javax.swing.JFormattedTextField.COMMIT;

public class Tietokanta {
   Connection db;

   public Tietokanta() throws SQLException {
      db = DriverManager.getConnection("jdbc:sqlite:tietokanta.db");
   }

   public void luonti() throws SQLException {
      try {
         Statement s = db.createStatement();
         s.execute("PRAGMA foreign_keys = ON");
         s.execute("CREATE TABLE Paikat (id INTEGER PRIMARY KEY, nimi TEXT UNIQUE NOT NULL)");
         s.execute("CREATE TABLE Asiakkaat (id INTEGER PRIMARY KEY, nimi TEXT UNIQUE NOT NULL)");
         s.execute("CREATE TABLE Paketit (id INTEGER PRIMARY KEY, seurantakoodi TEXT UNIQUE NOT NULL, asiakas_id
INTEGER NOT NULL REFERENCES Asiakkaat ON DELETE CASCADE)");
         s.execute("CREATE TABLE Tapahtumat (id INTEGER PRIMARY KEY, paivays TEXT, seurantakoodi_id INTEGER NOT
NULL REFERENCES Paketit ON DELETE CASCADE, paikka_id INTEGER NOT NULL REFERENCES Paikat ON DELETE
CASCADE, kuvaus TEXT NOT NULL)");
         s.execute("CREATE INDEX idx_seurantakoodiid ON Tapahtumat (seurantakoodi_id)");
         System.out.println("Tietokanta luotu");
      } catch (SQLException e) {
         System.out.println("VIRHE: Tietokanta on jo luotu");
      }
   }

   public void paikka(String nimi) throws SQLException {
      try {
         PreparedStatement p1 = db.prepareStatement("INSERT INTO Paikat(nimi) VALUES (?) ");
         p1.setString(1,nimi);
         p1.executeUpdate();
         System.out.println("Paikka lisätty");

      } catch (SQLException e) {
         if(e.getErrorCode()==1) {
            System.out.println("VIRHE: Tietokantaa ei ole luotu");
         } else {
            System.out.println("VIRHE: Paikka on jo olemassa");
         }
      }
   }

   public void asiakas(String nimi) throws SQLException {
      try{
         PreparedStatement p1 = db.prepareStatement("INSERT INTO Asiakkaat(nimi) VALUES (?)");
         p1.setString(1,nimi);
         p1.executeUpdate();
         System.out.println("Asiakas lisätty");

      } catch (SQLException e) {
         if(e.getErrorCode()==1) {
            System.out.println("VIRHE: Tietokantaa ei ole luotu");
         } else {
            System.out.println("VIRHE: Asiakas on jo olemassa");
         }
      }
   }
```

```
    public void paketti(String seuranta, String asiakas) throws SQLException {
        try {
            PreparedStatement p1 = db.prepareStatement("SELECT id FROM Asiakkaat WHERE nimi=?");
            p1.setString(1,asiakas);
            ResultSet r1 = p1.executeQuery();

            PreparedStatement p2 = db.prepareStatement("INSERT INTO Paketit(seurantakoodi,asiakas_id) VALUES (?,?)");
            p2.setString(1,seuranta);
            p2.setString(2,r1.getString("id"));
            p2.executeUpdate();
            System.out.println("Paketti lisätty");
        }
        catch (SQLException e) {
            if(e.getErrorCode()==1) {
                System.out.println("VIRHE: Tietokantaa ei ole luotu");
            }
            else if(e.getErrorCode()==19) {
                System.out.println("VIRHE: Seurantakoodi on jo olemassa");
            }
            else if(e.getErrorCode()==0) {
                System.out.println("VIRHE: Asiakasta ei ole olemassa");
            }
        }
    }

    public void tapahtuma(String seurantakoodi2, String paikka, String kuvaus) throws SQLException {
        try {
            PreparedStatement p1 = db.prepareStatement("SELECT id FROM Paketit WHERE seurantakoodi=?");
            p1.setString(1,seurantakoodi2);
            ResultSet r1 = p1.executeQuery();

            PreparedStatement p2 = db.prepareStatement("SELECT id FROM Paikat WHERE nimi=?");
            p2.setString(1,paikka);
            ResultSet r2 = p2.executeQuery();

            PreparedStatement p4 = db.prepareStatement("SELECT STRFTIME('%d.%m.%Y %H:%M', 'now','localtime');");
            ResultSet r4 = p4.executeQuery();

            PreparedStatement p3 = db.prepareStatement("INSERT INTO Tapahtumat(paivays,seurantakoodi_id,paikka_id,kuvaus) VALUES
(?,?,?,?)");
            p3.setString(1,r4.getString("STRFTIME('%d.%m.%Y %H:%M', 'now','localtime')"));
            p3.setString(2,r1.getString("id"));
            p3.setString(3,r2.getString("id"));
            p3.setString(4,kuvaus);
            p3.executeUpdate();
            System.out.println("Tapahtuma lisätty");

        } catch (SQLException e) {
            if(e.getErrorCode()==1) {
              System.out.println("VIRHE: Tietokantaa ei ole luotu");
            }
            else {
              System.out.println("VIRHE: Seurantakoodia tai paikkaa ei ole olemassa");
            }
        }
    }

    public void haku(String seuranta) throws SQLException {
        try{
            PreparedStatement p1 = db.prepareStatement("SELECT id FROM Paketit WHERE seurantakoodi=?");
            p1.setString(1,seuranta);
            ResultSet r1 = p1.executeQuery();
            PreparedStatement p2 = db.prepareStatement("SELECT T.paivays, P.nimi, T.kuvaus FROM Tapahtumat T, Paikat P WHERE
T.paikka_id=P.id AND T.seurantakoodi_id=?");
            p2.setString(1,r1.getString("id"));
```

```
            ResultSet r2 = p2.executeQuery();
               while(r2.next()) {
                  System.out.println(r2.getString("paivays")+", "+r2.getString("nimi")+", "+r2.getString("kuvaus"));
               }

         } catch (SQLException e) {
            if(e.getErrorCode()==1) {
               System.out.println("VIRHE: Tietokantaa ei ole luotu");
            } else{
               System.out.println("VIRHE: Seurantakoodia ei ole olemassa");
            }
         }
      }

      public void asiakastapahtumat(String asiakas) throws SQLException {
         try{
            PreparedStatement p1 = db.prepareStatement("SELECT id FROM Asiakkaat WHERE nimi=?");
            p1.setString(1,asiakas);
            ResultSet r1 = p1.executeQuery();
            PreparedStatement p2 = db.prepareStatement("SELECT P.seurantakoodi, count(T.id) FROM Paketit P LEFT JOIN Tapahtumat T
ON T.seurantakoodi_id=P.id WHERE P.asiakas_id=? GROUP BY P.seurantakoodi");
            p2.setString(1,r1.getString("id"));

            ResultSet r2 = p2.executeQuery();
               while(r2.next()) {
                  System.out.println(r2.getString("seurantakoodi")+", "+r2.getString("count(T.id)") +" tapahtumaa");
               }

         } catch (SQLException e) {
            if(e.getErrorCode()==1) {
               System.out.println("VIRHE: Tietokantaa ei ole luotu");
            } else{
               System.out.println("VIRHE: Seurantakoodia ei ole olemassa");
            }
         }
      }

      public void paikkatapahtumat(String paikka, String paiva) throws SQLException {
         try{
            PreparedStatement p1 = db.prepareStatement("SELECT id FROM Paikat WHERE nimi=?");
            p1.setString(1,paikka);
            ResultSet r1 = p1.executeQuery();

            PreparedStatement p2 = db.prepareStatement("SELECT COUNT(kuvaus) FROM Tapahtumat WHERE paikka_id=? AND paivays
LIKE ?");
            p2.setString(1,r1.getString("id"));
            p2.setString(2,paiva+ "%");

            ResultSet r2 = p2.executeQuery();
               while(r2.next()) {
                  System.out.println("Tapahtumien määrä: "+r2.getString("COUNT(kuvaus)"));
               }

         } catch (SQLException e) {
            if(e.getErrorCode()==1) {
               System.out.println("VIRHE: Tietokantaa ei ole luotu");
            } else{
               System.out.println("VIRHE: Paikkaa ei ole olemassa");
            }
         }
      }

      public void t1paikka() throws SQLException {
         Statement s = db.createStatement();
```

```
    s.execute("BEGIN TRANSACTION");
    try {
        PreparedStatement p1 = db.prepareStatement("INSERT INTO Paikat(nimi) VALUES (?) ");
        for(int i=1;i<=1000;i++) {
            String nimi = "P"+i;
            p1.setString(1,nimi);
            p1.executeUpdate();
            //System.out.println("Paikka lisätty");
        }
    } catch (SQLException e) {
        if(e.getErrorCode()==1) {
          System.out.println("VIRHE: Tietokantaa ei ole luotu");
        } else {
            System.out.println("VIRHE: Paikka on jo olemassa");
        }
    }
    s.execute("COMMIT");

}

public void t2asiakas() throws SQLException {
    Statement s = db.createStatement();
    s.execute("BEGIN TRANSACTION");
    try{
        PreparedStatement p1 = db.prepareStatement("INSERT INTO Asiakkaat(nimi) VALUES (?)");
        for(int i=1;i<=1000;i++) {
            String nimi = "A"+i;
            p1.setString(1,nimi);
            p1.executeUpdate();
            //System.out.println("Asiakas lisätty");
       }
    } catch (SQLException e) {
        if(e.getErrorCode()==1) {
            System.out.println("VIRHE: Tietokantaa ei ole luotu");
        } else {
            System.out.println("VIRHE: Asiakas on jo olemassa");
        }
    }
    s.execute("COMMIT");
}

public void t3paketti() throws SQLException {
    Statement s = db.createStatement();
    s.execute("BEGIN TRANSACTION");
    try {
        PreparedStatement p1 = db.prepareStatement("SELECT id FROM Asiakkaat WHERE nimi=?");
        PreparedStatement p2 = db.prepareStatement("INSERT INTO Paketit(seurantakoodi,asiakas_id) VALUES (?,?)");
        for(int i=1;i<=1000;i++) {
            String asiakas = "A"+i;
            String seuranta = "S"+i;
            p1.setString(1,asiakas);
            ResultSet r1 = p1.executeQuery();

            p2.setString(1,seuranta);
            p2.setString(2,r1.getString("id"));
            p2.executeUpdate();
            //System.out.println("Paketti lisätty");
        }
    }
    catch (SQLException e) {
        if(e.getErrorCode()==1) {
            System.out.println("VIRHE: Tietokantaa ei ole luotu");
        }
        else if(e.getErrorCode()==19) {
            System.out.println("VIRHE: Seurantakoodi on jo olemassa");
```

```
        }
        else if(e.getErrorCode()==0) {
            System.out.println("VIRHE: Asiakasta ei ole olemassa");
        }
    }
    s.execute("COMMIT");

}

public void t4tapahtuma() throws SQLException {
    Statement s = db.createStatement();
    s.execute("BEGIN TRANSACTION");
    try {

        PreparedStatement p1 = db.prepareStatement("SELECT id FROM Paketit WHERE seurantakoodi=?");
        PreparedStatement p2 = db.prepareStatement("SELECT id FROM Paikat WHERE nimi=?");
        PreparedStatement p4 = db.prepareStatement("SELECT STRFTIME('%d.%m.%Y %H:%M', 'now','localtime');");
        PreparedStatement p3 = db.prepareStatement("INSERT INTO Tapahtumat(paivays,seurantakoodi_id,paikka_id,kuvaus) VALUES
(?,?,?,?)");
        for(int y=1;y<=1000;y++) {
            for(int i=1;i<=1000;i++) {
                String paikka = "P"+i;
                String seurantakoodi2 = "S"+i;
                String kuvaus = paikka+seurantakoodi2;
                p1.setString(1,seurantakoodi2);
                ResultSet r1 = p1.executeQuery();

                p2.setString(1,paikka);
                ResultSet r2 = p2.executeQuery();

                ResultSet r4 = p4.executeQuery();

                p3.setString(1,r4.getString("STRFTIME('%d.%m.%Y %H:%M', 'now','localtime')"));
                p3.setString(2,r1.getString("id"));
                p3.setString(3,r2.getString("id"));
                p3.setString(4,kuvaus);
                p3.executeUpdate();
                //System.out.println("Tapahtuma lisätty");
            }
        }

    } catch (SQLException e) {
        if(e.getErrorCode()==1) {
            System.out.println("VIRHE: Tietokantaa ei ole luotu");
        }
        else {
          System.out.println("VIRHE: Seurantakoodia tai paikkaa ei ole olemassa");
        }
    }
    s.execute("COMMIT");
}

public void t5asiakaspaketit() throws SQLException {
    Statement s = db.createStatement();
    s.execute("BEGIN TRANSACTION");
    try{
        PreparedStatement p1 = db.prepareStatement("SELECT id FROM Asiakkaat WHERE nimi=?");
        PreparedStatement p2 = db.prepareStatement("SELECT COUNT(*) FROM Paketit WHERE asiakas_id=?");
        for(int i=1;i<=1000;i++) {
            String asiakas = "A"+i;
            p1.setString(1,asiakas);
            ResultSet r1 = p1.executeQuery();
            p2.setString(1,r1.getString("id"));
            ResultSet r2 = p2.executeQuery();
            while(r2.next()) {
```

```
                // System.out.println(asiakas+ ": " +r2.getString("count(*)") +" pakettia");
              }
            }
        } catch (SQLException e) {
            if(e.getErrorCode()==1) {
                System.out.println("VIRHE: Tietokantaa ei ole luotu");
            } else{
                System.out.println("VIRHE: Seurantakoodia ei ole olemassa");
            }
        }
        s.execute("COMMIT");
    }

    public void t6pakettitapahtumat() throws SQLException {
        Statement s = db.createStatement();
        s.execute("BEGIN TRANSACTION");
        Statement s1 = db.createStatement();
        //s1.executeUpdate("CREATE INDEX idx_seuranta ON Paketit (seurantakoodi)");
        try{
            PreparedStatement p1 = db.prepareStatement("SELECT id FROM Paketit WHERE seurantakoodi=?");
            PreparedStatement p2 = db.prepareStatement("SELECT count(id) FROM Tapahtumat WHERE seurantakoodi_id=?");
            for(int i=1;i<=1000;i++) {
                String seuranta = "S"+i;
                p1.setString(1,seuranta);
                ResultSet r1 = p1.executeQuery();
                p2.setString(1,r1.getString("id"));
                ResultSet r2 = p2.executeQuery();
                    while(r2.next()) {
                     //   System.out.println(seuranta+ ": "+r2.getString("count(id)") +" tapahtumaa");
                    }
            }
        } catch (SQLException e) {
            if(e.getErrorCode()==1) {
                System.out.println("VIRHE: Tietokantaa ei ole luotu");
            } else{
                System.out.println("VIRHE: Seurantakoodia ei ole olemassa");
            }
        }
        s.execute("COMMIT");
    }

}
```