Sini Saarinen
Opiskelijanumero: 015062088
MOOC-tunnus: sini.saarinen@hotmail.fi

Tietokantojen perusteet, kevät 2020

Tietokannan suunnittelu pakentinseurantajärjestelmälle

2. Harjoitustyön ja sen toimintojen toteutus

Harjoitustyö toteutettiin Javan ja SQL:n avulla ja siihen sisältyy 2 luokkaa: PaketinSeurantaFinal (käyttöliittymän käynnistämiseen) ja Käyttöliittymä. Työhön on toteutettu kaikki harjoitustyön vaatimusten mukaiset toiminnot, joista ohjelman käyttäjä pääsee valitsemaan komennoilla 1-9. Lisäksi ohjelman suorituksen voi päättää komennolla X. Komennot kutsuvat kukin eri metodeita.

Tietokanta luodaan komennolla 1. Jos tietokantaa ei ole aiemmin luotu, komento luo sovelluksen tarvitsemat neljä taulua tyhjään tietokantaan ja ilmoittaa käyttäjälle, kun toiminto on suoritettu. Jos komentoa yrittää toisen kerran, ohjelma ainoastaan ilmoittaa käyttäjälle, että tietokanta on jo lisätty.

Uusi paikka lisätään komennolla 2 ja uusi asiakas komennolla 3. Onnistuneesta lisäyksestä ilmoitetaan käyttäjälle. Jos samaa paikkaa tai asiakasta yritetään lisätä toisen kerran, ilmoittaa ohjelma tästä käyttäjälle eikä tee mitään muita toimenpiteitä.

Uusi paketti lisätään komennolla 4. Onnistuneesta lisäyksestä ilmoitetaan käyttäjälle. Jos asiakasta ei löydy tai paketti on jo olemassa, tulee tästä virheviesti käyttäjälle, eikä ohjelma suorita muita toimenpiteitä.

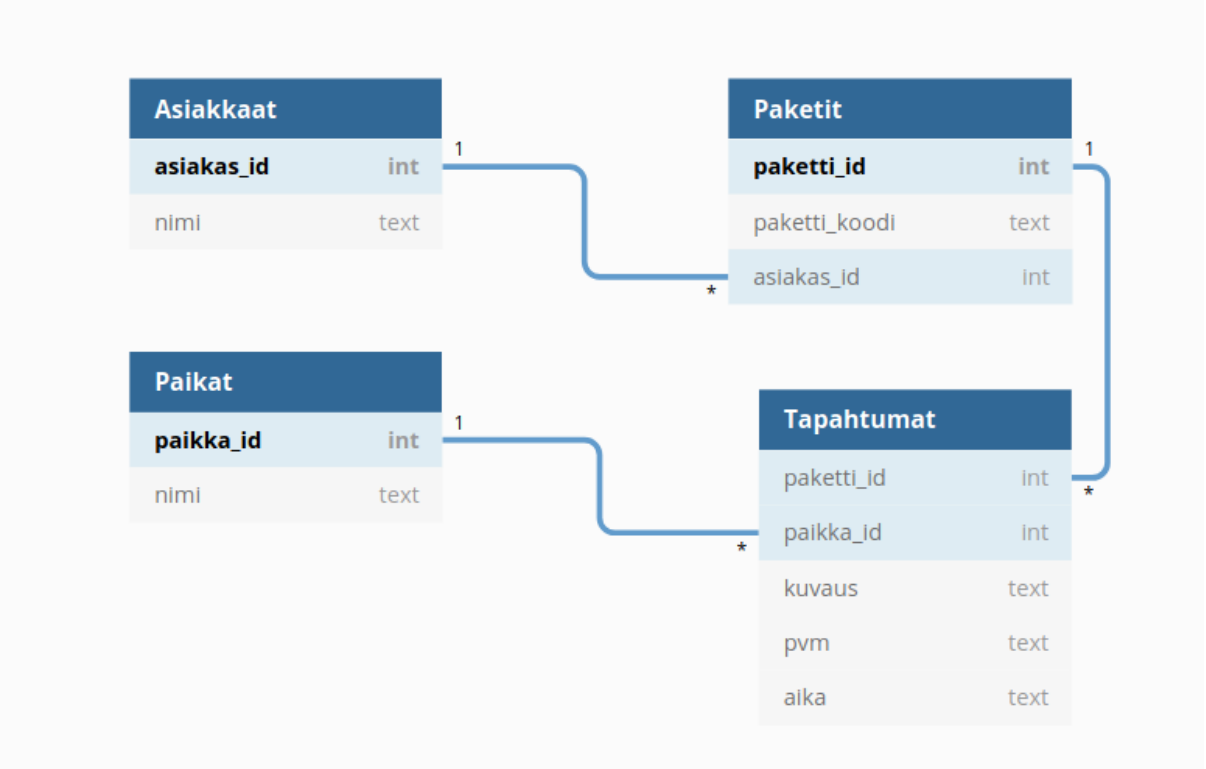
Uusi tapahtuma lisätään komennolla 5 ja onnistuneesta lisäyksestä ilmoitetaan käyttäjälle. Tapahtumaan lisätään käyttäjän ilmoittamien tietojen lisäksi senhetkinen päivämäärä ja kellonaika. Jos paikkaa tai seurantakoodia ei löydy, tästä ilmoitetaan käyttäjälle eikä ohjelma suorita muita toimenpiteitä.

Komennolla 6 haetaan paketin tapahtumat seurantakoodin perusteella. Jos seurantakoodia ei löydy, tästä ilmoitetaan käyttäjälle. Komento 7 hakee asiakkaan paketit ja niihin liittyvien tapahtumien määrän. Jos asiakasta ei löydy, tästä ilmoitetaan käyttäjälle. Komento 8 puolestaan ilmoittaa paikan tapahtumien lukumäärän tiettynä päivämääränä. Jos paikkaa tai päivämäärää ei löydy, tästä ilmoitetaan käyttäjälle.

Tehokkuustestit (komento 9) toteutettiin kahden eri metodin avulla: ensin kutsutaan ensimmäistä metodia, joka luo tarvittavat taulut ja suorittaa tehokkuustestin vaiheet 1-6 ilman indeksejä. Tämän jälkeen kutsutaan toista metodia, joka poistaa tauluista tehokkuustestin ensimmäisessä vaiheessa tietokantaan lisätyt tiedot, jonka jälkeen se lisää tauluihin sopivat indeksit. Tämän jälkeen kutsutaan uudelleen ensimmäistä metodia, joka suorittaa tehokkuustestin 1-6 vaiheet nyt nopeammin lisättyjen indeksien ansiosta.

Tehokkuustestin vaiheissa 3-6 satunnaisuutta luotiin tietokantaan Javan Random-funktion avulla. Kohdassa 3 kullekin lisättävälle paketille arvottiin asiakas väliltä A1-A1000 ja kohdassa 4 kullekin lisättävälle tapahtumalle paketti väliltä PP1-PP1000. Kohdassa 5 puolestaan haettiin pakettien määrä satunnaisesti valituilta asiakkailta (asiakas-id väliltä 1-1000) ja kohdassa 6 tapahtumien määrä satunnaisesti valituilta paketeilta (paketti-id väliltä 1-1000).

3. Tietokantakaavio ja SQL-skeema



Yllä laadittu tietokantakaavio ja alla käytetty SQL-skeema:

```
CREATE TABLE Paikat (paikka_id INTEGER PRIMARY KEY, nimi TEXT UNIQUE);
CREATE TABLE Asiakkaat (asiakas_id INTEGER PRIMARY KEY, nimi TEXT UNIQUE);
CREATE TABLE Paketit (paketti_id INTEGER PRIMARY KEY, paketti_koodi TEXT UNIQUE, asiakas_id INTEGER REFERENCES Asiakkaat);
CREATE TABLE Tapahtumat (paketti_id INTEGER REFERENCES Paketit, paikka_id INTEGER REFERENCES Paikat, kuvaus TEXT, pvm TEXT, aika TEXT);
```

4. Tehokkuustestit

Tehokkuustesteissä kului aikaa ilman indeksejä yhteensä 46,58s ja indekseillä yhteensä 12,24s. Suurin ero oli vaiheessa 6: ilman indeksejä vaiheessa 6 kesti yli 44 sekuntia, kun taas indekseillä aikaa kului ainoastaan 0,06 sekuntia. Vaihe 4 toisaalta kesti ilman indeksejä reilu 2 sekuntia, kun taas indekseillä aikaa kului lähes kuusi kertaa enemmän. Kaikenkaikkiaan tehokkuustestit hoituivat kuitenkin huomattavasti nopeammin indekseillä. Tarkemmat ajat vaihteittain alla:

Ilman indeksejä:

Vaihe 1: 0.042685843 sekuntia
Vaihe 2: 0.006302368 sekuntia
Vaihe 3: 0.005564965 sekuntia
Vaihe 4: 2.345001454 sekuntia
Vaihe 5: 0.050964038 sekuntia
Vaihe 6: 44.13071277 sekuntia

Indekseillä:

Vaihe 1: 0.005213244 sekuntia
Vaihe 2: 0.005026154 sekuntia
Vaihe 3: 0.007534902 sekuntia
Vaihe 4: 12.091949142 sekuntia
Vaihe 5: 0.062768481 sekuntia
Vaihe 6: 0.068068849 sekuntia

5. Tiedon eheys ja transaktiot

Sovelluksessa varmistettiin UNIQUE-ehdon avulla, ettei samannimistä asiakasta tai paikkaa eikä pakettia samalla seurantakoodilla pystytä lisäämään tietokantaan. Jos jotain edellisistä yrittää lisätä aiemmin lisätyllä nimellä tai seurantakoodilla, tulee tästä virheilmoitus eikä ohjelma suorita muita toimenpiteitä. Lisäksi käytettiin viiteavaimia ja REFERENCES-määrettä tauluja luodessa varmistamaan, että tauluissa olevat viittaukset viittaavat todellisiin riveihin. Myös “PRAGMA foreign_keys = ON” -komento suoritettiin ohjeiden mukaisesti. Ohjelman laatija myös testasi ohjelmaa asianmukaisesti ja riittävästi varmistuakseen aiemmin lueteltujen toimenpiteiden toimivuudesta.

Jos käyttäjiä on useampia samaan aikaan, niin SQLitessä transaktiot ovat täysin eristettyjä ja komennot käyttäytyvät aivan kuin transaktiot olisi suoritettu peräkkäin yksi kerrallaan jossain järjestyksessä. Jos toinen käyttäjä esimerkiksi muuttaa joitain tietoja tietokannassa sen jälkeen, kun toinen käyttäjä on hakenut tietoja samoista tauluista, eivät muutokset näy toiselle käyttäjälle. Tämä voi toisaalta hidastaa tai estää transaktioiden suorittamista. Samanaikaiset transaktiot aiheuttavat tietokannan lukkiutumisen, josta tulee käyttäjälle virheviesti “database is locked”.

SQLitessa ei ole mahdollista, että eri käyttäjät pääsisivät samaan aikaan lisäämään esimerkiksi samannimisen asiakkaan tai paketin. SQLite mahdollistaa saman tietokannan käytön usealle käyttäjälle samanaikaisesti, ja esimerkiksi hakuja on mahdollista toteuttaa samaan aikaan, mutta vain yksi käyttäjä voi tehdä muutoksia tietokantaan tietyllä hetkellä.

6. Ohjelman lähdekoodi

```
package paketinseurantafinal;

import java.sql.SQLException;
import java.util.Scanner;
/**
 *
 * @author saasini
 */
public class PaketinSeurantaFinal {
    /**
     * @param args the command line arguments
     * @throws java.sql.SQLException
     */
    public static void main(String[] args) throws SQLException {

        Käyttöliittymä kayttis = new Käyttöliittymä();
        Scanner lukija = new Scanner(System.in);
        kayttis.aloita(lukija);
    }
}
```

```
package paketinseurantafinal;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Random;
import java.util.Scanner;

public class Käyttöliittymä {

    private Connection db;

    public Käyttöliittymä() throws SQLException {
        this.db = DriverManager.getConnection("jdbc:sqlite:testi.db");
    }
```

```
    public void aloita(Scanner lukija) throws SQLException {

        Statement s = this.db.createStatement();

        while (true) {
            System.out.println("Valitse toiminto (1-9 tai X): ");
            System.out.println(" X - lopeta");
            System.out.println(" 1 - luo tietokanta");
            System.out.println(" 2 - lisää paikka");
            System.out.println(" 3 - lisää asiakas");
            System.out.println(" 4 - lisää paketti");
            System.out.println(" 5 - lisää tapahtuma");
            System.out.println(" 6 - hae paketin tapahtumat");
            System.out.println(" 7 - hae asiakkaan paketit");
            System.out.println(" 8 - hae paikan tapahtumat");
            System.out.println(" 9 - suorita tehokkuustesti");
            System.out.println("");

            String komento = lukija.nextLine();

            if (komento.equals("X")) {
                break;
            } else if (komento.equals("1")) {
                lisaaTietokanta(lukija, s);
            } else if (komento.equals("2")) {
                lisaaPaikka(lukija, s);
            } else if (komento.equals("3")) {
                lisaaAsiakas(lukija, s);
            } else if (komento.equals("4")) {
                lisaaPaketti(lukija, s);
            } else if (komento.equals("5")) {
                lisaaTapahtuma(lukija, s);
            } else if (komento.equals("6")) {
                haePaketinTapahtumat(lukija, s);
            } else if (komento.equals("7")) {
                haeAsiakkaanPaketit(lukija, s);
            } else if (komento.equals("8")) {
                haePaikanTapahtumat(lukija, s);
            } else if (komento.equals("9")) {
                tehokkuusTesti(s);
                poistaTiedotJaLuoIndeksit(s);
                tehokkuusTesti(s);
            } else {
                System.out.println("Tuntematon komento.");
            }
        }
    }

    public void lisaaTietokanta(Scanner lukija, Statement s) {
        try {
            s.execute("PRAGMA foreign_keys = ON");
            s.execute("CREATE TABLE Paikat (paikka_id INTEGER PRIMARY KEY, nimi TEXT UNIQUE)");
            s.execute("CREATE TABLE Asiakkaat (asiakas_id INTEGER PRIMARY KEY, nimi TEXT UNIQUE)");
            s.execute("CREATE TABLE Paketit (paketti_id INTEGER PRIMARY KEY, paketti_koodi TEXT UNIQUE, asiakas_id INTEGER
REFERENCES Asiakkaat)");
            s.execute("CREATE TABLE Tapahtumat (paketti_id INTEGER REFERENCES Paketit, paikka_id INTEGER REFERENCES
Paikat, kuvaus TEXT, pvm TEXT, aika TEXT)");
            System.out.println("Tietokanta luotu.");
        } catch (SQLException e) {
            System.out.println("VIRHE: Tietokanta on jo lisätty.");
        }
    }

    public void lisaaPaikka(Scanner lukija, Statement s) {
        System.out.println("Anna paikan nimi: ");
        String paikka = lukija.nextLine();
        try {
            PreparedStatement p = db.prepareStatement("INSERT INTO Paikat(nimi) VALUES (?)");
            p.setString(1,paikka);
            p.executeUpdate();
            System.out.println("Paikka lisätty.");
        } catch (SQLException e) {
            System.out.println("VIRHE: Paikka on jo olemassa.");
        }
    }

    public void lisaaAsiakas(Scanner lukija, Statement s) {
        System.out.println("Anna asiakkaan nimi: ");
```

```
        String asiakas = lukija.nextLine();
        try {
            PreparedStatement p = db.prepareStatement("INSERT INTO Asiakkaat(nimi) VALUES (?)");
            p.setString(1,asiakas);
            p.executeUpdate();
            System.out.println("Asiakas lisätty.");
        } catch (SQLException e) {
            System.out.println("VIRHE: Asiakas on jo olemassa.");
        }
    }

    public void lisaaPaketti(Scanner lukija, Statement s) {
        System.out.println("Anna paketin seurantakoodi: ");
        String seurantakoodi = lukija.nextLine();
        System.out.println("Anna asiakkaan nimi: ");
        String asiakas = lukija.nextLine();
        try {
            PreparedStatement p = db.prepareStatement("INSERT INTO Paketit(paketti_koodi, asiakas_id) VALUES (?,?)");
            p.setString(1,seurantakoodi);
            PreparedStatement a = db.prepareStatement("SELECT asiakas_id FROM Asiakkaat WHERE nimi=?");
            a.setString(1,asiakas);
            ResultSet r = a.executeQuery();
            int id = r.getInt("asiakas_id");
            p.setInt(2,id);
            p.executeUpdate();
            System.out.println("Paketti lisätty.");
        } catch (SQLException e) {
            System.out.println("VIRHE: Asiakasta ei löydy tai paketti on jo olemassa.");
        }
    }

    public void lisaaTapahtuma(Scanner lukija, Statement s) {
        System.out.println("Anna paketin seurantakoodi: ");
        String seurantakoodi = lukija.nextLine();
        System.out.println("Anna tapahtuman paikka: ");
        String paikka = lukija.nextLine();
        System.out.println("Anna tapahtuman kuvaus: ");
        String kuvaus = lukija.nextLine();
        try {
            PreparedStatement p = db.prepareStatement("INSERT INTO Tapahtumat(paketti_id, paikka_id, kuvaus, pvm, aika) VALUES
(?,?,?,?,?)");
            PreparedStatement b = db.prepareStatement("SELECT paketti_id FROM Paketit WHERE paketti_koodi=?");
            b.setString(1,seurantakoodi);
            ResultSet re = b.executeQuery();
            int id1 = re.getInt("paketti_id");
            p.setInt(1, id1);
            PreparedStatement a = db.prepareStatement("SELECT paikka_id FROM Paikat WHERE nimi=?");
            a.setString(1,paikka);
            ResultSet r = a.executeQuery();
            int id = r.getInt("paikka_id");
            p.setInt(2, id);
            p.setString(3, kuvaus);
            DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");
            DateTimeFormatter dtf2 = DateTimeFormatter.ofPattern("HH:mm:ss");
            LocalDateTime now = LocalDateTime.now();
            String pvm = dtf.format(now);
            String aika = dtf2.format(now);
            p.setString(4, pvm);
            p.setString(5, aika);
            p.executeUpdate();
            System.out.println("Tapahtuma lisätty.");
        } catch (SQLException e) {
            System.out.println("VIRHE: Paikkaa tai seurantakoodia ei löydy.");
        }
    }

    public void haePaketinTapahtumat(Scanner lukija, Statement s) {
        System.out.println("Anna paketin seurantakoodi: ");
        String seurantakoodi = lukija.nextLine();
        try {
            PreparedStatement b = db.prepareStatement("SELECT paketti_id FROM Paketit WHERE paketti_koodi=?");
            b.setString(1,seurantakoodi);
            ResultSet re = b.executeQuery();
            int id1 = re.getInt("paketti_id");
            PreparedStatement p = db.prepareStatement("SELECT Tapahtumat.pvm, Tapahtumat.aika, Paikat.nimi, Tapahtumat.kuvaus FROM
Tapahtumat JOIN Paikat ON Tapahtumat.paikka_id=Paikat.paikka_id WHERE Tapahtumat.paketti_id=?");
            p.setInt(1, id1);
            ResultSet r = p.executeQuery();
```

```
            while (r.next()) {
                System.out.println(r.getString("pvm")+" "+r.getString("aika")+", "+r.getString("nimi")+", "+r.getString("kuvaus"));
            }
        } catch (SQLException e) {
            System.out.println("VIRHE: Seurantakoodia ei löydy.");
        }
    }

    public void haeAsiakkaanPaketit(Scanner lukija, Statement s) {
        System.out.println("Anna asiakkaan nimi: ");
        String asiakas = lukija.nextLine();
        try {
            PreparedStatement p = db.prepareStatement("SELECT Paketit.paketti_koodi, COALESCE(COUNT(Tapahtumat.kuvaus), 0) as lkm
FROM Paketit LEFT JOIN Tapahtumat ON Paketit.paketti_id=Tapahtumat.paketti_id WHERE Paketit.asiakas_id=? GROUP BY
Paketit.paketti_koodi");
            PreparedStatement c = db.prepareStatement("SELECT asiakas_id FROM Asiakkaat WHERE nimi=?");
            c.setString(1,asiakas);
            ResultSet res = c.executeQuery();
            int id2 = res.getInt("asiakas_id");
            p.setInt(1,id2);
            ResultSet r = p.executeQuery();
            while (r.next()) {
                System.out.println(r.getString("paketti_koodi")+", "+r.getInt("lkm")+" tapahtuma(a) ");
            }
        } catch (SQLException e) {
            System.out.println("VIRHE: Asiakasta ei löydy.");
        }
    }

    public void haePaikanTapahtumat(Scanner lukija, Statement s) {
        System.out.println("Anna paikan nimi: ");
        String paikka = lukija.nextLine();
        System.out.println("Anna päivämäärä (muodossa YYYY/MM/DD): ");
        String pvm = lukija.nextLine();
        try {
            PreparedStatement d = db.prepareStatement("SELECT paikka_id FROM Paikat WHERE nimi=?");
            d.setString(1,paikka);
            ResultSet resu = d.executeQuery();
            int id3 = resu.getInt("paikka_id");
            PreparedStatement p = db.prepareStatement("SELECT COALESCE(COUNT(Tapahtumat.kuvaus), 0) as lkm FROM Tapahtumat
WHERE Tapahtumat.paikka_id=? AND Tapahtumat.pvm=?");
            p.setInt(1, id3);
            p.setString(2, pvm);
            ResultSet r = p.executeQuery();
            while (r.next()) {
                System.out.println(r.getInt("lkm")+" tapahtumaa");
            }
        } catch (SQLException e) {
            System.out.println("VIRHE: Paikkaa tai valittua päivämäärää ei löydy.");
        }
    }

    public void tehokkuusTesti(Statement s) throws SQLException {
        System.out.println("Suoritetaan tehokkuustesti.");

        s.execute("BEGIN TRANSACTION");
        Random random = new Random();

        //1. vaihe
        long t1aika1 = System.nanoTime();
        PreparedStatement p1 = db.prepareStatement("INSERT INTO Paikat(nimi) VALUES (?)");
        for (int i=1; i<=1000; i++) {
            p1.setString(1, "P"+i);
            p1.executeUpdate();
        }
        long t1aika2 = System.nanoTime();
        System.out.println("Aikaa kului "+(t1aika2-t1aika1)/1e9+" sekuntia");

        //2. vaihe
        long t2aika1 = System.nanoTime();
        PreparedStatement p2 = db.prepareStatement("INSERT INTO Asiakkaat(nimi) VALUES (?)");
        for (int i=1; i<=1000; i++) {
            p2.setString(1, "A"+i);
            p2.executeUpdate();
        }
        long t2aika2 = System.nanoTime();
        System.out.println("Aikaa kului "+(t2aika2-t2aika1)/1e9+" sekuntia");
```

```
        //3. vaihe
        long t3aika1 = System.nanoTime();
        PreparedStatement p3 = db.prepareStatement("INSERT INTO Paketit(paketti_koodi, asiakas_id) VALUES (?,(SELECT asiakas_id
FROM Asiakkaat WHERE nimi=?))");
        for (int i=1; i<=1000; i++) {
            int numero = random.nextInt(1000)+1;
            p3.setString(1, "PP"+i);
            p3.setString(2, "A"+numero);
            p3.executeUpdate();
        }
        long t3aika2 = System.nanoTime();
        System.out.println("Aikaa kului "+(t3aika2-t3aika1)/1e9+" sekuntia");

        //4. vaihe
        long t4aika1 = System.nanoTime();
        PreparedStatement p4 = db.prepareStatement("INSERT INTO Tapahtumat(paketti_id) VALUES ((SELECT paketti_id FROM Paketit
WHERE paketti_koodi=?))");
        for (int i=1; i<=1000000; i++) {
            int numero = random.nextInt(1000)+1;
            p4.setString(1, "PP"+numero);
            p4.executeUpdate();
        }
        long t4aika2 = System.nanoTime();
        System.out.println("Aikaa kului "+(t4aika2-t4aika1)/1e9+" sekuntia");
        s.execute("COMMIT");

        //5. vaihe
        s.execute("BEGIN TRANSACTION");
        long t5aika1 = System.nanoTime();
        PreparedStatement p5 = db.prepareStatement("SELECT COALESCE(COUNT(paketti_koodi), 0) FROM Paketit WHERE
asiakas_id=?");
        for (int i=1; i<=1000; i++) {
            int numero = random.nextInt(1000)+1;
            p5.setInt(1,i);
            p5.executeQuery();
        }
        long t5aika2 = System.nanoTime();
        System.out.println("Aikaa kului "+(t5aika2-t5aika1)/1e9+" sekuntia");
        s.execute("COMMIT");

        //6. vaihe
        s.execute("BEGIN TRANSACTION");
        long t6aika1 = System.nanoTime();
        PreparedStatement p6 = db.prepareStatement("SELECT COALESCE(COUNT(paketti_id), 0) FROM Tapahtumat WHERE
paketti_id=?");
        for (int i=1; i<=1000; i++) {
            int numero1 = random.nextInt(1000)+1;
            p6.setInt(1,numero1);
            p6.executeQuery();
        }
        long t6aika2 = System.nanoTime();
        System.out.println("Aikaa kului "+(t6aika2-t6aika1)/1e9+" sekuntia");
        s.execute("COMMIT");
    }

    public void poistaTiedotJaLuoIndeksit(Statement s) throws SQLException {
        PreparedStatement poisto1 = db.prepareStatement("DELETE FROM Tapahtumat");
        PreparedStatement poisto2 = db.prepareStatement("DELETE FROM Paketit");
        PreparedStatement poisto3 = db.prepareStatement("DELETE FROM Paikat");
        PreparedStatement poisto4 = db.prepareStatement("DELETE FROM Asiakkaat");
        poisto1.executeUpdate();
        poisto2.executeUpdate();
        poisto3.executeUpdate();
        poisto4.executeUpdate();
        PreparedStatement indeksit1 = db.prepareStatement("CREATE INDEX paikka_index ON Paikat(nimi)");
        PreparedStatement indeksit2 = db.prepareStatement("CREATE INDEX nimi_index ON Asiakkaat(nimi)");
        PreparedStatement indeksit3 = db.prepareStatement("CREATE INDEX koodi_index ON Paketit(paketti_koodi)");
        PreparedStatement indeksit4 = db.prepareStatement("CREATE INDEX paketti_id_index ON Tapahtumat(paketti_id)");
        indeksit1.executeUpdate();
        indeksit2.executeUpdate();
        indeksit3.executeUpdate();
        indeksit4.executeUpdate();
    }
}
```