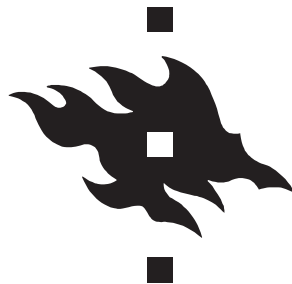


Tietokantojen perusteet, kevät 2020
Harjoitustyön raportti

Ville Manninen

014922187
macville

12. helmikuuta 2020



HELSINGIN YLIOPISTO

Sisältö

1	Esittely	3
2	Kaaviot ja Skeemat	5
3	Tehokkuustesti	6
4	Tietokannan eheys	7
5	Lähdekoodi	8

1 Esittely

Yleistä

Harjoitustyö on toteutettu Java-ohjelmana käyttäen NetBeans ympäristöä, ohjelmalla on seuraavat ominaisuudet.

- Tietokannan ja puuttuvien taulukoiden luominen.
- Asiakkaiden lisääminen tietokantaan nimellä.
- Paikan lisääminen tietokantaan nimellä.
- Paketin lisääminen tietokantaan käyttäen asiakkaan nimeä.
- Tapahtuman lisääminen tietokantaan käyttäen paikannimeä ja paketin seurantakoodia.
- Paketin tapahtumien hakeminen
- Asiakkaan pakettien hakeminen ja yksittäisten pakettien tapahtumien lukumäärä'
- Tapahtumien määrän hakeminen paikasta tietyllä päivämäärällä

Komennot

Ohjelmalle annetaan komentoja komentoriviltä käyttäen pelkästään komennon numeroa, komennon jälkeen ohjelma kysyy käyttäjältä tarvittavia jatko-syötteitä. Ohjelmalla on seuraavat komennot jotka esiintyvät myös ohjelman valikossa.

- 1** Uuden asiakkaan lisääminen
- 2** Uuden paikan lisääminen
- 3** Paketin lisääminen
- 4** Uuden tapahtuman lisääminen
- 5** Tulostaa paketin tapahtumat
- 6** Tulostaa asiakkaan paketit ja niihin liittyvien tapahtumien määrän
- 7** Tapahtumien määrän haku paikasta tietyllä päivämäärällä
- 0** Sulkee ohjelman

2 Kaaviot ja Skeemat

Tietokantakaavio

SQL-skeema

3 Tehokkuustesti

Ilman indeksejä

Indeksien lisäämisen jälkeen

4 Tietokannan eheys

Ohjelman tietokannan eheyttä pidetään yllä käyttämällä SQLiten sekä Javan tarjoamia ominaisuuksia. Tietokantaan ei ole mahdollista luoda samannimisiä asiakkaita tai paikkoja, eikä tietokantaan luoduille paketeille saa antaa samaa seurantakoodia. Samannimisiä syötteitä valvotaan SQLiten rajoitteella UNIQUE, joka varmistaa ettei samaan taulukkoon lisätä samannimisiä arvoja. Lisäksi SQLiten rajoitteella NOT NULL valvotaan että sarakkeisiin ei lisätä tyhjiä arvoja.

Java-ohjelma tarkistaa käyttäjän antamat syötteet ja ilmoittaa syyn jos syöte on puutteellinen, esimerkiksi jos käyttäjän syöte on tyhjä. Tietokantaan lisättäessä paketteja tai tapahtumia jotka viittaavat toisiin taulukoihin, sovellus varmistaa että annetut viitteet löytyvät tietokannasta.

5 Lähdekoodi

<https://github.com/Viltska/sql-app-training>.

Main.java

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

public class Main {

    public static void main(String[] args) throws SQLException {
        Scanner lukija = new Scanner(System.in);
        String databaseName = "tikape.db";
        System.out.println("Current database: " + databaseName);
        DatabaseManager manager = new DatabaseManager(databaseName);

        //Komentorivi ohjelma
        System.out.println("Komennot:");
        System.out.println("");
        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 paketin tapahtumat");
        System.out.println("6: Hae asiakkaan paketit");
        System.out.println("7: Hae paikan tapahtumien määrä
            päivämäärällä");
        System.out.println("9: Luo tietokannan ja/tai puuttuvat
            taulukot");
        System.out.println("0: Lopettaa ohjelman");
        System.out.println("");

        while (true) {
            System.out.print("Syötä komento: ");
            String komento = lukija.nextLine();
            if (komento.equals("0")) {
                manager.tikapePrint();
                System.out.println("Ohjelma suljetaan.");
                break;
            }
            if (komento.equals("1")) {
                System.out.print("Syötä nimi: ");
                String nimi = lukija.nextLine();
                manager.uusiAsiakas(nimi);
            }
            if (komento.equals("2")) {
                System.out.print("Syötä paikannimi: ");
```



```

        String paikka = lukija.nextLine();
        manager.uusiPaikka(paikka);
    }

    if (komento.equals("3")) {
        System.out.print("Syötä paketin asiakas: ");
        String asiakas = lukija.nextLine();
        System.out.print("Syötä paketin seurantakoodi: ");
        String koodi = lukija.nextLine();

        manager.uusiPaketti(asiakas, koodi);
    }
    if (komento.equals("4")) {
        System.out.print("Syötä tapahtuman paikka: ");
        String paikka = lukija.nextLine();
        System.out.print("Syötä paketin seurantakoodi: ");
        String seurantaKoodi = lukija.nextLine();
        System.out.print("Syötä tapahtuman kuvaus: ");
        String kuvaus = lukija.nextLine();
        manager.uusiTapahtuma(paikka, seurantaKoodi, kuvaus);
    }
    if (komento.equals("5")) {
        System.out.print("Syötä paketin seurantakoodi: ");
        String seurantaKoodi = lukija.nextLine();
        manager.haePaketinTapahtumat(seurantaKoodi);
    }
    if (komento.equals("6")) {
        System.out.print("Syötä asiakkaan nimi: ");
        String asiakas = lukija.nextLine();
        manager.haeAsiakkaanPaketit(asiakas);
    }
    if (komento.equals("7")) {
        System.out.println("Syötä päivämäärä muodossa
        'YYYY-MM-DD'");
        System.out.print("Päivämäärä: ");
        String pvm = lukija.nextLine();
        System.out.print("Syötä paikannimi: ");
        String paikannimi = lukija.nextLine();
        manager.haePaikanTapahtumatPaivamaaralla(pvm,
        paikannimi);
    }

    if (komento.equals("9")) {
        System.out.println("Tarkistetaan tietokantaa..");
        manager.createTables();
    }
}

```

```
    }  
}
```

DatabaseManager.java

```
import java.sql.Connection;  
import java.sql.*;  
  
public class DatabaseManager {  
  
    //Taulukko luokat  
    private final Asiakkaat asiakkaat;  
    private final Paikat paikat;  
    private final Paketit paketit;  
    private final Tapahtumat tapahtumat;  
    // SQL  
    private final String connection;  
    private final Connection db;  
  
    public DatabaseManager(String database) throws SQLException {  
        this.connection = "jdbc:sqlite:" + database;  
        this.db = DriverManager.getConnection(connection);  
  
        this.asiakkaat = new Asiakkaat(this.db);  
        this.paikat = new Paikat(this.db);  
        this.paketit = new Paketit(this.db);  
        this.tapahtumat = new Tapahtumat(this.db);  
    }  
  
    public void uusiAsiakas(String nimi) throws SQLException {  
        //Tarkistetaan että syöte ei ole tyhjä  
        if (!nimi.isEmpty()) {  
            this.asiakkaat.uusiAsiakas(nimi);  
        } else {  
            System.out.println("Asiakkaan nimi ei saa olla tyhjä");  
        }  
    }  
  
    public void uusiPaikka(String nimi) throws SQLException {  
        //Tarkistetaan että syöte ei ole tyhjä  
        if (!nimi.isEmpty()) {  
            this.paikat.uusiPaikka(nimi);  
        } else {  
            System.out.println("Paikannimi ei saa olla tyhjä");  
        }  
    }  
  
    public void uusiPaketti(String asiakas, String koodi) throws  
        SQLException {
```

```

        // .getID(asiakas) hakee SQL tietokannasta asiakkaan id, jos
        // asiakasta ei löydy palautetaan arvo -1
        //Tarkistetaan että syötteet eivät ole tyhjä
        if (!asiakas.isEmpty() && !koodi.isEmpty()) {
            int asiakas_id = asiakkaat.getID(asiakas);
            //Jos asiakasta ei löydy tietokannasta ei pakettia lisätä
            if (asiakas_id != -1) {
                paketit.uusiPaketti(asiakas_id, koodi);
            } else {
                System.out.println("Asiakasta ei löytynyt");
            }
        } else {
            System.out.println("Asiakkaan nimi tai seurantakoodi
                eivät saa olla tyhiä");
        }
    }

    public void uusiTapahtuma(String paikka, String seurantaKoodi,
        String kuvaus) throws SQLException {
        //Tarkistetaan että syötteet eivät ole tyhjä
        if (!paikka.isEmpty() && !seurantaKoodi.isEmpty() &&
            !kuvaus.isEmpty()) {
            int paikka_id = paikat.getPaikkaID(paikka);
            int paketti_id = paketit.getID(seurantaKoodi);
            //Tarkistetaan että paikka ja paketti löytyvät
            //tietokannasta
            if (paikka_id != -1) {
                if (paketti_id != -1) {
                    tapahtumat.uusiTapahtuma(paikka_id, paketti_id,
                        kuvaus);
                } else {
                    System.out.println("Seurantakoodilla ei löytynyt
                        pakettia");
                }
            } else {
                System.out.println("Paikkaa ei löytynyt");
            }
        } else {
            System.out.println("Syötteet eivät saa olla tyhjä");
        }
    }

    public void haePaketinTapahtumat(String seurantaKoodi) throws
        SQLException {
        //Tarkistetaan että syöte ei ole tyhjä
        if (!seurantaKoodi.isEmpty()) {
            int paketti_id = paketit.getID(seurantaKoodi);

```

```

        //Tarkistetaan että paketti löytyy tietokannasta
        if (paketti_id != -1) {
            try {
                paketit.haePaketinTapahtumat(paketti_id);
            } catch (SQLException e) {
                System.out.println(e);
            }
        } else {
            System.out.println("Pakettia ei löytynyt");
        }
    } else {
        System.out.println("Syöte ei saa olla tyhjä");
    }
}

public void haeAsiakkaanPaketit(String asiakas) throws
    SQLException {
    int asiakkaan_id = asiakkaat.getID(asiakas);
    //Tarkistetaan että syöte ei ole tyhjä
    if (!asiakas.isEmpty()) {
        //Tarkistetaan että asiakas löytyy tietokannasta
        if (asiakkaan_id != -1) {
            try {
                asiakkaat.haeAsiakkaanPaketit(asiakkaan_id);
            } catch (SQLException e) {
                System.out.println(e);
            }
        } else {
            System.out.println("Asiakasta ei löytynyt
                tietokannasta");
        }
    } else {
        System.out.println("Syöte ei saa olla tyhjä");
    }
}

public void haePaikanTapahtumatPaivamaaralla(String paivamaara,
    String paikannimi) throws SQLException {
    int paikka_id = paikat.getPaikkaID(paikannimi);
    if (paikka_id != -1) {
        try {
            tapahtumat.haeTapahtumatPaikasta(paivamaara,
                paikannimi);

        } catch (SQLException e) {
            System.out.println(e);
        }
    }
}

```

```

    } else {
        System.out.println("Paikkaa ei löytynyt");
    }
}

public void createTables() throws SQLException {
    Statement s = db.createStatement();
    try {
        s.execute("CREATE TABLE Asiakkaat (id INTEGER PRIMARY
            KEY, nimi TEXT NOT NULL UNIQUE)");
        System.out.println("Luotu taulukko 'Asiakkaat'");
    } catch (SQLException e) {
        System.out.println("Löytyi taulukko 'Asiakkaat'");
    }
    try {
        s.execute("CREATE TABLE Paikat (id INTEGER PRIMARY KEY,
            paikannimi TEXT NOT NULL UNIQUE)");
        System.out.println("Luotu taulukko 'Paikat'");
    } catch (SQLException e) {
        System.out.println("Löytyi taulukko 'Paikat'");
    }
    try {
        s.execute("CREATE TABLE Paketit (id INTEGER PRIMARY KEY,
            asiakas_id INTEGER NOT NULL, seurantakoodi TEXT NOT
            NULL UNIQUE)");
        System.out.println("Luotu taulukko 'Paketit'");
    } catch (SQLException e) {
        System.out.println("Löytyi taulukko 'Paketit'");
    }
    try {
        s.execute("CREATE TABLE Tapahtumat (id INTEGER PRIMARY
            KEY, paikka_id INTEGER NOT NULL, paketti_id INTEGER
            NOT NULL, date DATETIME, kuvaus TEXT NOT NULL)");
        System.out.println("Luotu taulukko 'Tapahtumat'");
    } catch (SQLException e) {
        System.out.println("Löytyi taulukko 'Tapahtumat'");
    }
    System.out.println("Tietokanta valmis.");
}

public void tikapePrint() {
    System.out.println("-----");
}

```

```

        System.out.println(" ----- ");
        System.out.println("    ");
        System.out.println("/\\_ _ \\/\\_ _ \\ /\\ \\/\\ \\ /\\ _
        \\/\\_ _ '\\ /\\_ _ '\\ ");
        System.out.println("\\/_/\\ \\/\\/_/\\ \\/ \\ \\ \\/\\/'/'\\ \\
        \\L\\ \\ \\ \\L\\ \\ \\ \\ \\L\\_\\ ");
        System.out.println(" \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ , < \\ \\ _
        \\ \\ ,_/_/\\ \\ _\\L ");
        System.out.println(" \\ \\ \\ \\ _\\_\\ \\ \\ \\ \\ \\ '\\ \\
        \\ \\/\\ \\ \\ \\/ \\ \\ \\L\\ \\ ");
        System.out.println(" \\ \\ _\\_ /\\_\\_\\_\\ \\_\\ \\ _\\_\\_\\
        _\\_\\ _\\_\\ _\\_\\ \\ \\ _\\_\\_\\");
        System.out.println(" \\/_/ \\/_\\_\\_\\/_/_/_/_/
        _/_/_/_/_/_/_/ \\/_\\_\\_\\ ");
        System.out.println("
        ");
        System.out.println("-----");
    }
}

```

Asiakkaat.java

```

import java.sql.*;

public class Asiakkaat {

    private final Connection db;

    public Asiakkaat(Connection db) throws SQLException {
        this.db = db;
    }

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

    public int getID(String asiakas) throws SQLException {
        //Palauttaa -1 jos asiakasta ei löydy tietokannasta
        try {
            PreparedStatement p = db.prepareStatement("SELECT id

```

```

        FROM Asiakkaat WHERE nimi=?");
p.setString(1, asiakas);

ResultSet r = p.executeQuery();

if (r.next()) {
    return r.getInt("id");

} else {
    return -1;
}
} catch (SQLException e) {
    System.out.println(e);
}
return -1;
}

public void haeAsiakkaanPaketit(int asiakas_id) throws
SQLException {
    try {
        PreparedStatement p = db.prepareStatement("SELECT
            seurantakoodi, COUNT(Tapahtumat.paketti_id) FROM
            Paketit\n"
            + "LEFT JOIN Tapahtumat ON Tapahtumat.paketti_id
            = Paketit.id\n"
            + "LEFT JOIN Asiakkaat ON Asiakkaat.id =
            Paketit.asiakas_id\n"
            + "GROUP BY Paketit.seurantakoodi HAVING
            Paketit.asiakas_id = ?;");
        p.setInt(1, asiakas_id);
        ResultSet r = p.executeQuery();

        if (r.next() == false) {
            System.out.println("Asiakkaalla ei ollut paketteja");
        } else {
            System.out.println("Asiakkaan paketit");
            System.out.println("-----");
            do {
                System.out.println("Paketti (" +
                    r.getString("seurantakoodi") + ")");
                System.out.println("Tapahtumia(" +
                    r.getString("COUNT(Tapahtumat.paketti_id)") +
                    ")");
                System.out.println("-----");
            } while (r.next());
        }
    } catch (SQLException e) {
        System.out.println(e);
    }
}

```

```

    }

    }

}

```

Paikat.java

```

import java.sql.*;

public class Paikat {

    private final Connection db;

    public Paikat(Connection db) throws SQLException {
        this.db = db;
    }

    public void uusiPaikka(String nimi) throws SQLException {

        try {
            PreparedStatement p = db.prepareStatement("INSERT INTO
                Paikat (paikannimi) VALUES (?)");
            p.setString(1, nimi);
            p.executeUpdate();
            System.out.println("Paikka lisätty");
        } catch (SQLException e) {
            System.out.println(e);
        }
    }

    public int getPaikkaID(String paikka) throws SQLException {
        //Palauttaa -1 jos paikkaa ei löydy tietokannasta
        try {
            PreparedStatement p = db.prepareStatement("SELECT id
                FROM Paikat WHERE paikannimi=?");
            p.setString(1, paikka);

            ResultSet r = p.executeQuery();

            if (r.next()) {
                return r.getInt("id");
            } else {
                return -1;
            }
        } catch (SQLException e) {
            System.out.println(e);
        }
    }
}

```



```

    }
    return -1;
}
}

```

Paketit.java

```

import java.sql.*;

public class Paketit {

    private final Connection db;

    public Paketit(Connection db) throws SQLException {
        this.db = db;
    }

    public void uusiPaketti(int asiakas_id, String seurantaKoodi)
        throws SQLException {
        try {
            PreparedStatement p = db.prepareStatement("INSERT INTO
                Paketit (asiakas_id,seurantakoodi) VALUES (?,?)");
            p.setInt(1, asiakas_id);
            p.setString(2, seurantaKoodi);
            p.executeUpdate();
            System.out.println("Paketti lisätty");
        } catch (SQLException e) {
            System.out.println(e);
        }
    }

    public int getID(String seurantaKoodi) throws SQLException {
        // Palauttaa arvon -1 jos pakettia ei löydy tietokannasta
        try {
            PreparedStatement p = db.prepareStatement("SELECT id
                FROM Paketit WHERE seurantaKoodi=?");
            p.setString(1, seurantaKoodi);

            ResultSet r = p.executeQuery();

            if (r.next()) {
                return r.getInt("id");
            } else {
                return -1;
            }
        } catch (SQLException e) {
            System.out.println(e);
        }
    }
}

```

```

    }
    return -1;
}

public void haePaketinTapahtumat(int paketti_id) throws
SQLException {
    try {
        PreparedStatement p = db.prepareStatement("SELECT
            paikannimi,kuvaus, seurantakoodi, date FROM
            Tapahtumat, Paikat Paketit WHERE
            Tapahtumat.paikka_id = Paikat.id AND
            Tapahtumat.paketti_id = Paketit.id AND Paketit.id =
            ?");
        p.setInt(1, paketti_id);
        ResultSet r = p.executeQuery();

        if (r.next() == false) {
            System.out.println("Paketilla ei ollut tapahtumia");
        } else {
            System.out.println("-----");
            do {
                System.out.println("TAPAHTUMA (" +
                    r.getString("date") + ")");
                System.out.println("Sijainti: " +
                    r.getString("paikannimi"));
                System.out.println("Kuvaus: " +
                    r.getString("kuvaus"));
                System.out.println("-----");
            } while (r.next());

        }

    } catch (SQLException e) {
        System.out.println(e);
    }
}

public int paketinTapahtumienMaara(int paketti_id) throws
SQLException {
    try {
        PreparedStatement p = db.prepareStatement("SELECT
            COUNT(*) FROM Tapahtumat, Paikat, Paketit WHERE
            Tapahtumat.paikka_id = Paikat.id AND
            Tapahtumat.paketti_id = Paketit.id AND Paketit.id =
            ?");
        p.setInt(1, paketti_id);
    }
}

```

```

        ResultSet r = p.executeQuery();

        while (r.next()) {
            return 0;
        }

    } catch (SQLException e) {
        System.out.println(e);
    }

    return -1;
}
}

```

Tapahtumat.java

```

import java.sql.*;

public class Tapahtumat {

    private final Connection db;
    private final Statement s;

    public Tapahtumat(Connection db) throws SQLException {
        this.db = db;
        this.s = db.createStatement();
    }

    public void uusiTapahtuma(int paikka_id, int paketti_id, String
        kuvaus) throws SQLException {
        try {
            PreparedStatement p = db.prepareStatement("INSERT INTO
                Tapahtumat (paikka_id,paketti_id,date,kuvaus) VALUES
                (?, ?,datetime(), ?)");
            p.setInt(1, paikka_id);
            p.setInt(2, paketti_id);
            p.setString(3, kuvaus);
            p.executeUpdate();
            System.out.println("Tapahtuma lisätty");

        } catch (SQLException e) {
            System.out.println(e);
        }
    }

    public void haeTapahtumatPaikasta(String paivamaara, String
        paikannimi) throws SQLException {

```

```

try {
    PreparedStatement p = db.prepareStatement("SELECT
        COUNT(Tapahtumat.id) AS Tapahtumia FROM Tapahtumat,
        Paikat WHERE Paikat.id = Tapahtumat.paikka_id AND
        DATE(Tapahtumat.date) = ? AND Paikat.paikannimi =
        ?");
    p.setString(1, paivamaara);
    p.setString(2, paikannimi);
    ResultSet r = p.executeQuery();

    if (r.next() == false) {
        System.out.println("Paikassa ei ollut tapahtumia
            annetulla päivämäärällä");
    } else {
        do {
            System.out.println("Tapahtumien määrä: " +
                r.getString("Tapahtumia"));

        } while (r.next());
    }

} catch (SQLException e) {
    System.out.println(e);
}

}

```