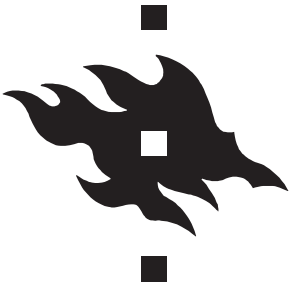
Tietokantojen perusteet, kevät 2020
Harjoitustyön raportti

Ville Manninen *†

24. helmikuuta 2020



HELSINGIN YLIOPISTO

*mooc.fi: vw.manninen@gmail.com
†opiskelijanumero: 014922187

Sisältö

1 Esittely

1.1 Tavoite

Harjoitustyön tavoitteena on luoda esimerkkisovellus tietokannan käytöstä sovelluksessa. Harjoitustyön sovelluksen ohjelmointi on toteutettu Java kielellä käyttäen NetBeans kehitysympäristöä. Ohjelman tietokantaa pitää yllä SQLite ja sitä tukeva ajuri.

1.2 Ohjelman ominaisuudet

Ohjelmalla on seuraavat ominaisuudet.

- Tietokannan luominen kun tietokanta puuttuu tai on puutteellinen.
- Asiakkaiden lisääminen tietokantaan.
- Paikkojen lisäämien tietokantaan.
- Pakettien lisääminen tietokantaan.
- Tapahtumien lisääminen tietokantaan.
- Pakettien tapahtumien hakeminen.
- Asiakkaan pakettien hakeminen ja yksittäisten pakettien tapahtumien lukumäärä.
- Tapahtumien määrän hakeminen halutusta paikasta päivämäärällä.
- Tehokkuustestin suorittaminen.

1.3 Komentorivi komennot

Ohjelmalle voidaan antaa komentoja syöttmällä komentoa kuvaava numero komentoriville. Ohjelmalla on seuraavat, numeroidut komennot.

1 lisää uuden asiakkaan tietokantaan.

2 lisää uuden paikan tietokantaan.

3 lisää uuden paketin tietokantaan.

4 lisää uuden tapahtuman tietokantaan.

5 tulostaa paketin tapahtumat tietokannasta.

6 tulostaa asiakkaan paketit ja niihin liittyvien tapahtumien määrän tietokannasta.

7 tulostaa tapahtumien määrän haku paikasta tietyllä päivämäärällä tietokannasta.

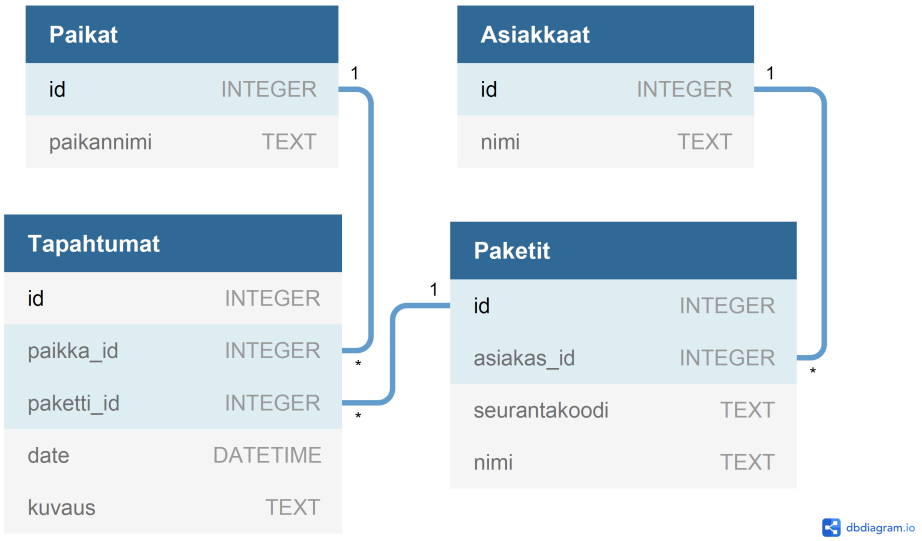
8 suorittaa tehokkuustestin ja tulostaa testin tulokset käyttäjälle.

9 luo tietokannan.

0 Sulkee ohjelman.

2 Kaaviot ja Skeemat

2.1 Tietokantakaavio



2.2 SQL-skeema

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

CREATE TABLE Paikat (id INTEGER PRIMARY KEY, paikannimi TEXT NOT
    NULL UNIQUE);

CREATE TABLE Paketit (id INTEGER PRIMARY KEY, asiakas_id INTEGER
    NOT NULL, seurantakoodi TEXT NOT NULL UNIQUE);

CREATE TABLE Tapahtumat (id INTEGER PRIMARY KEY, paikka_id INTEGER
    NOT NULL, paketti_id INTEGER NOT NULL, date DATETIME NOT NULL,
    kuvaus TEXT NOT NULL);
```

3 Tehokkuustesti

3.1 Testin vaiheet

Ohjelmalla voidaan suorittaa tehokkuustesti jonka avulla voidaan testata tietokannan tehokkuutta. Tehokkuustesti suorittaa ennalta määriteltyjä muutoksia tietokantaan ja tulostaa testeihin käytetyn ajan. Tehokkuustesti luo uuden tietokannan johon testin aikana luodut arvot tallennetaan, testin jälkeen ohjelma palaa käyttämään vanhaa tietokantaa.

Testi koostuu kuudesta eri vaiheesta, testin vaiheissa yhdestä neljään tietokantaan lisätään tuhat asiakasta, paikkaa ja pakettia. Seuraavaksi tietokantaan lisätään miljoona tapahtumaa käyttäen satunnaisesti tietokantaan lisättyjä arvoja. Ohjelmaa testatessa tuhannen arvon lisääminen tapahtui alle sekunnin kymmenesosassa kun taas miljoonaa arvoa lisätessä aikaa kului noin minuutti, tästä nähdään että ohjelman tehokkuuteen vaikuttaa merkittävästi tietokantaan lisättävien arvojen määrä.

Viimeisissä vaiheissa viisi ja kuusi testi suorittaa kyselyitä joissa haetaan tuhannen asiakkaan pakettien määrä, sekä tuhannen paketin tapahtumien määrä. Mainituissa vaiheissa testin tehokkuuteen vaikutti eniten indeksien käyttö.

3.2 Alikyselyn vaikutus tehokkuutten

3.3 Ilman indeksiä

Ilman lisättyä indeksiä pakettien hakemiseen tietokannasta aikaa kului alle sekunnin, mutta tapahtumia haettaessa aikaa kuluikin yli minuutin. Tarkastelun jälkeen huomataan että tapahtumaa haettaessa SQLite suorittaa alikyselyn jokaisen tapahtuman kohdalla. Mainittu alikysely etsii paketit taulukosta nimellä haluttua pakettia, tämä käsittely hidastaa kyselyä merkittävästi.

3.4 Indeksin lisäämisen jälkeen

Tehokkuuden lisäämiseksi voidaan paketin arvolle nimi luoda hakemistorakenne paketin nimelle. Käyttämällä SQLiten CREATE INDEX toimintoa, indeksin lisäämisen jälkeen tapahtuma kysely hakee nimellä paketin id-numeron tehokkaasti. Indeksin lisäämisen jälkeen tapahtuma kysely vei aikaa noin sekunnin.

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 tauluun lisätä samannimisiä arvoja. Lisäksi SQLiten rajoitteella NOT NULL valvotaan että sarakkeisiin ei lisätä tyhjiä arvoja, tätä käytetään hyväksi esimerkiksi kun lisätään arvoja jotka viittaavat muihin sarakkeisiin, alikysely palauttaa arvon NULL jos arvoa johon viitataan ei löydy tietokannasta.

Ohjelma tarkistaa käyttäjän antamat syötteet ja hylkää syötteet jotka eivät kelpaa ohjelmalle. Tietokantaan lisättäessä paketteja tai tapahtumia jotka viittaavat toiseen taulukkoon tallennettuun tietoon, sovellus varmistaa että viitatut tiedot löytyvät tietokannassa.

Tapauksissa joissa ohjelmalle syötetään arvo jota se ei kelpuuta ohjelma tulostaa käyttäjälle virheilmoituksen ja jatkaa ohjelman suorittamista. Virheilmoitukset kertovat käyttäjälle miksi annettu syöte ei kelvannut. Tapauksissa joissa SQLite antaa virheilmoituksen tulostetaan myös tämä käyttäjälle.

5 Lähdekoodi

Esittely

Seuraavasta kappaleesta löytyy ohjelman lähdekoodi, sekä lyhyt esittely ohjelman rakenteesta:

Ohjelma koostuu kuudesta luokasta, luokka Main suorittaa komentorivisovellusta, kun tehdään muutoksia tietokantaan kutsutaan luokkaa DatabaseManager. DatabaseManager luokka luo ja muokkaa tietokanna taulukoita joista jokaisella on oma luokkansa. Ohjelman lähdekoodi ja raportti löytyvät myös alla olevasta osoitteesta.

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

Main.java

DatabaseManager.java

Asiakkaat.java

Paikat.java

Paketit.java

Tapahtumat.java