Tehtävä 3: Indeksin tehokkuus

Lomakkeen yläreuna

Tee ohjelma, jonka avulla voidaan suorittaa tietokannan tehokkuustesti. Tehokkuustestin osat ovat:

- Ohjelma luo taulun Elokuvat, jossa on sarakkeet id, nimi ja vuosi.
- Ohjelma lisää tauluun miljoona riviä. Jokaisen rivin kohdalla nimeksi valitaan satunnainen merkkijono ja vuodeksi valitaan satunnainen kokonaisluku väliltä 1900–2000.
- Ohjelma suorittaa tuhat kertaa kyselyn, jossa haetaan elokuvien määrä vuonna x. Jokaisessa kyselyssä x valitaan satunnaisesti väliltä 1900–2000.

Toteuta ohjelma niin, että kaikki rivit lisätään saman transaktion sisällä (esimerkiksi alussa komento BEGIN ja lopussa komento COMMIT), jotta rivien lisääminen ei vie liikaa aikaa.

Elokuvien määrän laskemisessa käytä kyselyä, jossa haetaan COUNT(*) ehtoon täsmäävistä riveistä.

Suorita ohjelman avulla kolme testiä seuraavasti:

1. Tauluun ei lisätä kyselyitä tehostavaa indeksiä.
2. Tauluun lisätään kyselyitä tehostava indeksi ennen rivien lisäämistä.
3. Tauluun lisätään kyselyitä tehostava indeksi ennen kyselyiden suoritusta.

Ilmoita jokaisesta testistä rivien lisäämiseen ja kyselyiden suoritukseen kuluva aika sekä tietokantatiedoston koko testin jälkeen. Miten voit selittää testin tulokset?

Varmista ennen varsinaisia testejä, että kyselysi antavat järkeviä tuloksia: indeksin tulisi nopeuttaa kyselyjä merkittävästi testeissä 2 ja 3. Älä kuitenkaan tulosta varsinaisissa testeissä mitään muuta kuin ajankäyttö, jotta tulostaminen ei vääristä testin tulosta.

Testi 1

Rivien lisäämiseen kuluva aika: sekuntia

Kyselyiden suoritukseen kuluva aika: sekuntia

Tietokantatiedoston koko testin lopuksi: megatavua

Testi 2

Rivien lisäämiseen kuluva aika: sekuntia

Kyselyiden suoritukseen kuluva aika: sekuntia

Tietokantatiedoston koko testin lopuksi: megatavua

Testi 3

Rivien lisäämiseen kuluva aika: sekuntia

Kyselyiden suoritukseen kuluva aika: sekuntia

Tietokantatiedoston koko testin lopuksi: megatavua

#Tehtävä 1: Tauluun ei lisätä kyselyitä tehostavaa indeksiä

- Exclude the index information from the table

- By using transaction 'Begin/Commit', random string to 'nimi' and random integer from 1990-2000 to 'vuosi' are added to the table 'Elo27'. It took 19.042953968048096 second for the year 1999 to execute the table insertion for the year 1999. The running time varies differently, depending on each time the insertion is executed.

- The query COUNT function calculating that, 91081 movies for the year 1999 took 154.62019896507263 seconds and the execution time also differs each time the query and different year input are executed.

- The total amount of megabytes was calculated based on count(*) multiple with 8 bytes for random string and 8 bytes for random integer. For this execution, it took 15 megabytes.

Anna vuosi: 1999
Elokuvien maaran vuonna 1999 91081
Tietokantatiedoston koko testin lopuksi (megatavua) 15
Rivien lisaamiseen kuluva aika 19.042953968048096
Kyselyiden suoritukseen kuluva aika 154.62019896507263

#Tehtävä 2: Tauluun lisätään kyselyitä tehostava indeksi ennen rivien lisäämistä.

- Include the index information to the table before adding the rows

- By using transaction 'Begin/Commit', random string to 'nimi' and random integer from 1990-2000 to 'vuosi' are added to the table 'Elo1'. It took 18.6104519367218 second for the year 19999 to execute the table insertion. The running time varies differently, depending on each time the insertion is executed.

- The query COUNT function calculating that, 91145 movies for the year 1999 took 139.71149802207947 seconds and the execution time also differs each time the query and different year input are executed.

- The total amount of megabytes was calculated based on count(*) multiple with 8 bytes for index, 8 bytes for random string and 8 bytes for random integer. For this execution, it took 22 megabytes.

Anna vuosi: 1999
Elokuvien maaran vuonna 1999 91145
Tietokantatiedoston koko testin lopuksi (megatavua) 15
Rivien lisaamiseen kuluva aika 18.6104519367218
Kyselyiden suoritukseen kuluva aika 139.71149802207947

#Tehtävä 3: Tauluun lisätään kyselyitä tehostava indeksi ennen kyselyiden suoritusta.

- Include the index information to the table before the query

- By using transaction 'Begin/Commit', random string to 'nimi' and random integer from 1990-2000 to 'vuosi' are added to the table 'Elo27'. It took 18.46564793586731 seconds for the year 1999 to execute the table insertion. The running time varies differently, depending on each time the insertion is executed.

- The query COUNT function calculating that, 90737 movies for the year 1999 took 28.70013999938965 seconds and the execution time also differs each time the query and different year input are executed.

- The total amount of megabytes was calculated based on count(*) multiple with 8 bytes for index, 8 bytes for random string and 8 bytes for random integer. For this execution, it took 15 megabytes.

Valitaan vuosi: 1999
Elokuvien maaran vuonna 1999 90737
Rivien lisaamiseen kuluva aika 18.46564793586731
Kyselyiden suoritukseen kuluva aika 28.70013999938965
Tietokantatiedoston koko testin lopuksi (megatavua) 15

CODE

```
#TEHTÄVÄ 1:

import sqlite3
import random
from random import randint
import string
import time
import secrets

import os

os.system("rm movies.db")


db = sqlite3.connect("movies.db")
db.isolation_level = None

# Tauluun ei lisätä kyselyitä tehostavaa indeksiä

db.execute("CREATE TABLE Elo27 (nimi TEXT, vuosi INTEGER)")
n=10**6

start1 = time.time()
db.execute('BEGIN')
for i in range(n):
        db.execute('INSERT INTO Elo27 (nimi, vuosi) VALUES (?, ?)',
[secrets.token_bytes(8), random.randint(1990, 2000)])
db.execute('COMMIT')
interval1 = time.time() - start1



start2 = time.time()
vuosi = input('Anna vuosi: ')
for i in range(1000):
        rivit = db.execute('SELECT vuosi, COUNT(*) FROM Elo27 WHERE vuosi = ?
GROUP BY vuosi ORDER BY vuosi', [vuosi]).fetchall()
for rivi in rivit:
        print('Elokuvien maaran vuonna', rivi[0], rivi[1])
interval2= time.time()-start2


megatavua = db.execute('SELECT ((count(*)*(8+8))/1024/ 1024) FROM
Elo27').fetchone()
print('Tietokantatiedoston koko testin lopuksi (megatavua)', megatavua[0])
```

```
print('Rivien lisaamiseen kuluva aika', interval1)
print('Kyselyiden suoritukseen kuluva aika', interval2)


------------------------------------------------------------------------------
---------------

#TEHTÄVÄ 2:

import sqlite3
import random
from random import randint
import string
import time
import secrets

import os

os.system("rm movies.db")

db = sqlite3.connect("movies.db")
db.isolation_level = None

# Tauluun lisätään kyselyitä tehostava indeksi ennen rivien lisäämistä.

db.execute("CREATE TABLE Elo1 (id INTEGER PRIMARY KEY, nimi TEXT, vuosi
INTEGER)")

n=10**6
start1 = time.time()

db.execute('BEGIN')
for i in range(n):
        db.execute('INSERT INTO Elo1 (nimi, vuosi) VALUES (?, ?)',
[secrets.token_hex(8), random.randint(1990, 2000)])
db.execute('COMMIT')

interval1 = time.time() - start1

start2 = time.time()
vuosi = input('Anna vuosi: ')
for i in range(1000):
        rivit = db.execute('SELECT vuosi, COUNT(*) FROM Elo1 WHERE vuosi = ?
GROUP BY vuosi ORDER BY vuosi', [vuosi]).fetchall()
for rivi in rivit:
        print('Elokuvien maaran vuonna', rivi[0], rivi[1])
interval2= time.time()-start2
print('Rivien lisaamiseen kuluva aika', interval1)
print('Kyselyiden suoritukseen kuluva aika', interval2)

megatavua = db.execute('SELECT ((count(*)*(8+8+8))/1024 / 1024) FROM
Elo1').fetchone()
print('Tietokantatiedoston koko testin lopuksi', megatavua[0])


------------------------------------------------------------------------------
------------------


#TEHTÄVÄ 3:
```

```
import sqlite3
import random
from random import randint
import string
import time
import secrets
import os

os.system("rm movies.db")


db = sqlite3.connect("movies.db")
db.isolation_level = None

# Tauluun ei lisätä kyselyitä tehostavaa indeksiä

db.execute("CREATE TABLE Elo27 (nimi TEXT, vuosi INTEGER)")
n=10**6

start1 = time.time()
db.execute('BEGIN')
for i in range(n):
        db.execute('INSERT INTO Elo27 (nimi, vuosi) VALUES (?, ?)',
[secrets.token_bytes(8), random.randint(1990, 2000)])
db.execute('COMMIT')
interval1 = time.time() - start1



start2 = time.time()
db.execute('CREATE INDEX idx_vuosi ON Elo27 (vuosi)')

idx_vuosi = input('Anna vuosi: ')
for i in range(1000):
        rivit = db.execute('SELECT vuosi, COUNT(*) FROM Elo27 WHERE vuosi = ?
GROUP BY vuosi ORDER BY vuosi', [idx_vuosi]).fetchall()
for rivi in rivit:
        print('Elokuvien maaran vuonna', rivi[0], rivi[1])
interval2= time.time()-start2


megatavua = db.execute('SELECT ((count(*)*(8+8))/1024/ 1024) FROM
Elo27').fetchone()
print('Tietokantatiedoston koko testin lopuksi (megatavua)', megatavua[0])

print('Rivien lisaamiseen kuluva aika', interval1)
print('Kyselyiden suoritukseen kuluva aika', interval2)
```