Veebiarendus

Back-End arendus

Martti Raavel

martti.raavel@tlu.ee

Tänased teemad

- Meenutame eelmist loengut
- Relatsiooniline andmebaas
- MySQL
- MySQL Dockeris
- Andmebaasi loomine
- Tabelite loomine
- Andmete sisestamine
- Päringute tegemine
- Andmebaasi ühendamine NodeJS-iga

Millest rääkisime eelmine kord?

Relatsiooniline andmebaas

Relatsiooniline andmebaas - 1

Relatsioonilised andmebaasid on laialdaselt kasutatav andmehaldusvahend, mis võimaldab andmeid salvestada struktureeritud ja korraldatud kujul. Need andmebaasid kasutavad tabelite süsteemi, et andmeid talletada ja hallata, ning tuginevad relatsioonilisele mudelile.

Relatsioonilise andmebaasi põhikomponendid

- Tabelid
- Primaarvõti (Primary Key)
- Võõrvõti (Foreign Key)
- Indeksid
- ...

Tabelid

Tabelid on relatsiooniliste andmebaaside põhielemendid, mis sisaldavad ridu ja veerge. Iga tabel esindab kindlat andmekogumit (nt kliendid, tellimused, tooted).

- **Rida (Row):** Tabeli andmekirje. Iga rida sisaldab andmeid vastavalt tabeli veergudele. Rida võib olla ka nimetatud kui kirje või olem.
- **Veerg (Column):** Tabeli omadus või atribuut, mida nimetatakse ka väljadeks.

Näide tabelist:

ID	Nimi	Vanus
1	Jaan	25
2	Mari	30

Primaarvõti (Primary Key)

Primaarvõti on unikaalne identifikaator, mis eristab iga tabeli rida. Primaarvõti tagab, et iga kirje on unikaalne.

Võõrvõti (Foreign Key)

Võõrvõti on veerg või veergude kombinatsioon, mis loob seose kahe või enama tabeli vahel. Võõrvõti viitab primaarvõtmele teises tabelis.

Indeksid

Indeks on andmestruktuur, mis võimaldab kiiremat andmete otsimist ja päringute täitmist tabelis. Indeksid luuakse ühe või mitme veeru põhjal ja need loovad viiteid, mis võimaldavad kiiremini leida konkreetseid ridu tabelist.

MySQL

MySQL on avatud lähtekoodiga relatsiooniline andmebaasihaldussüsteem (**RDBMS** - *Relational Database Management System*), mis on laialdaselt kasutatav veebirakenduste arendamisel. MySQL kasutab SQL-i (Structured Query Language) andmete haldamiseks ja päringute tegemiseks.

Docker

Docker on avatud lähtekoodiga konteinerite virtualiseerimise platvorm, mis võimaldab arendajatel luua, käivitada ja jagada rakendusi konteinerites. Docker kasutab Linuxi konteinerite tehnoloogiat, et pakendada rakenduse kood, sõltuvused ja konfiguratsioon ühtsesse konteinerisse.

Docker konteinerid

Docker konteinerid on isoleeritud ja kergkaalulised *virtuaalmasinad*, mis käivitatakse Dockeri platvormil. Iga konteiner sisaldab rakenduse koodi, sõltuvusi ja konfiguratsiooni, mis on pakendatud konteineri imageks.

Docker Hub

Docker Hub on Dockeri ametlik registriplatvorm, mis võimaldab arendajatel leida, jagada ja käivitada Dockeri konteinereid. Docker Hub sisaldab tuhandeid avalikke konteinereid, mida saab kasutada erinevate rakenduste ja teenuste käivitamiseks.

MySQL Dockeris

Kursuse käigus kasutame MySQL-i konteinerit Dockeris, et luua ja hallata andmebaase arenduskeskkonnas, kuna see on kiire ja lihtne viis MySQL-i käivitamiseks ja kasutamiseks. Lisaks ei pea me otseselt paigaldama MySQL-i oma arvutisse.

MySQL Dockeri käivitamine

MySQL-i keskkonnamuutujad:

- `MYSQL_ROOT_PASSWORD` : MySQL-i juurkasutaja parool.
- `MYSQL_DATABASE` : Andmebaasi nimi, mida soovite luua.
- `MYSQL_USER` : Andmebaasi kasutajanimi.
- `MYSQL_PASSWORD` : Andmebaasi kasutaja parool.

SQLTools VS Code lisandmoodul

SQLTools on Visual Studio Code lisandmoodul, mis võimaldab arendajatel ühendada ja hallata SQL-andmebaase otse VS Code keskkonnas. SQLTools toetab mitmeid andmebaasisüsteeme, sealhulgas MySQL, PostgreSQL, SQLite, SQL Server ja paljud teised.

Andmebaasi loomine

Andmebaasi loomiseks kasutame SQL-i käsku `CREATE DATABASE`, millele järgneb andmebaasi nimi.

```
CREATE DATABASE mydatabase;
```

> Kui kasutame MySQL-i Dockeris, siis loome andmebaasi juba Dockeri konteineri käivitamisel kasutades keskkonnamuutujat `MYSQL_DATABASE`.

Tabelite loomine

Tabeli loomiseks kasutame SQL-i käsku `CREATE TABLE`, millele järgneb tabeli nimi ja veergude määratlused.

```
CREATE TABLE users (
    id INT PRIMARY KEY,
    username VARCHAR(50),
    email VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Andmetüübid

MySQL toetab mitmeid andmetüüpe, mis määravad veergude väärtuste tüübi ja piirangud. Mõned levinumad andmetüübid on:

- `INT` : Täisarvuline väärtus.
- `VARCHAR(n)` : Muutuvpikkusega tähemärkide ahel.
- `TEXT` : Pikad tekstilised andmed.
- `DATE` : Kuupäev.
- `DATETIME` : Kuupäev ja kellaaeg.
- `TIMESTAMP` : Kuupäev ja kellaaeg, mida uuendatakse automaatselt.
- `BOOLEAN` : Tõeväärtus (TRUE või FALSE).
- ...

> `BOOLEAN` andmetüübi tegelikult MySQL-is ei eksisteeri ja selle kasutamisel luuakse tegelikult `TINYINT(1)` veerg.

Tabelite muutmine

Tabelite muutmiseks kasutame SQL-i käsku `ALTER TABLE`, millele järgneb tabeli nimi ja muudatused.

```
ALTER TABLE users ADD COLUMN password VARCHAR(255);
```

Välja kustutamiseks kasutame `DROP COLUMN` käsku.

```
ALTER TABLE users DROP COLUMN password;
```

Tabeli kustutamine

Tabeli kustutamiseks kasutame SQL-i käsku `DROP TABLE`, millele järgneb tabeli nimi.

```
DROP TABLE users;
```

> Olge ettevaatlik, kui kasutate `DROP` käsku, kuna see kustutab andmed ja struktuuri pöördumatult.

Seosed (Relationships)

Andmebaasides kasutatakse seoseid, et luua seosed kahe või enama tabeli vahel.
Seosed võimaldavad andmeid tõhusalt pärida ja hallata, luues seoseid võõrvõtmete abil.

Võõrvõtme lisamine

Võõrvõtme lisamiseks kasutame SQL-i käsku `FOREIGN KEY`, millele järgneb veeru nimi ja viide teise tabeli primaarvõtmele.

```
CREATE TABLE posts (
    id INT PRIMARY KEY,
    user_id INT,
    title VARCHAR(255),
    content TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Seoste tüübid

- **Üks-ühele (One-to-One):** Iga kirje esimeses tabelis vastab täpselt ühele kirjele teises tabelis.
- **Üks-paljudele (One-to-Many):** Iga kirje esimeses tabelis vastab mitmele kirjele teises tabelis.
- **Palju-paljudele (Many-to-Many):** Mitu kirjet esimeses tabelis vastab mitmele kirjele teises tabelis ja vastupidi.

Andmete sisestamine

Andmete sisestamiseks kasutame SQL-i käsku `INSERT INTO`, millele järgneb tabeli nimi ja veergude väärtused.

```
INSERT INTO users (username, email) VALUES ('alice', 'alice@alice.ee');
```

Päringute tegemine

Andmete lugemiseks ja pärimiseks tabelitest kasutame SQL-i käsku `SELECT`, millele järgneb veergude nimed või `*` märge kõigi veergude valimiseks.

```
SELECT * FROM users;
```

Täpsustatud päringud

Päringute täpsustamiseks kasutame `WHERE` tingimust, mis piirab päringu tulemusi vastavalt määratud tingimustele.

```
SELECT * FROM users WHERE username = 'alice';
```

```
SELECT * FROM users WHERE id = 1;
```

```
SELECT * FROM users WHERE age > 18;
```

Määrame, mis andmeid soovime kuvada

```sql
SELECT username, email FROM users;
```

```sql
SELECT username, email FROM users WHERE age > 18;
```

Andmebaasi ühendamine NodeJS-iga

MySQL NodeJS-iga

Andmebaasi ühendamine NodeJS-iga toimub MySQL mooduli abil, mis võimaldab luua ühenduse MySQL andmebaasiga, saata päringuid ja saada vastuseid.

```
const mysql = require('mysql2');

const pool = mysql.createPool({
  host: 'localhost',
  user: 'mysql-user',
  password: 'mysql-password',
  database: 'mydatabase'
});

const promisePool = pool.promise();

module.exports = promisePool;
```

Andmebaasi päringud teenustes

Andmebaasi päringute tegemiseks loome teenused, mis kasutavad andmebaasi ühendust ja saadavad päringuid.

```
const db = require('../db');

const getAllUsers = async () => {
  const [rows] = await db.query('SELECT * FROM users');
  return rows;
};

const getUserById = async (id) => {
  const [rows] = await db.query('SELECT * FROM users WHERE id = ?', [id]);
  return rows[0];
};

const createUser = async (username, email) => {
  const [result] = await db.query('INSERT INTO users (username, email) VALUES (?, ?)', [username, email]);
  return result.insertId;
};
```

SQL-injection

SQL-injection on turvarünnak, mis võimaldab ründajal süstida pahatahtlikku SQL-koodi veebirakendusse, et manipuleerida andmebaasi päringuid ja saada juurdepääs konfidentsiaalsetele andmetele.

Parameetrilised päringud

Parameetrilised päringud võimaldavad kasutada muutujaid ja väärtusi päringutes, mis muudavad päringud dünaamiliseks ja turvalisemaks. ,

Parameetrilised päringud aitavad vältida SQL-injection rünnakuid.

```
const getUserById = async (id) => {
  const [rows] = await db.query('SELECT * FROM users WHERE id = ?', [id]);
  return rows[0];
};
```

> Pane tähele, et päringu muutuja asendatakse küsimärgiga `?` ja väärtused antakse massiivina.

Kodutöö

- Loo MySQL andmebaas ja tabelid, mis sisaldavad kasutajaid, postitusi ja kommentaare.
- Lisa tabelitesse andmeid, et oleks võimalik oma API-s neid kasutada.
- Tee päringuid, mis tagastavad kasutajate, postituste ja kommentaaride andmeid.
- Ühenda andmebaas NodeJS-iga ja tee päringuid kasutades teenuseid.