Database Performance

#INSY #STO

> Getestete Datenbanksysteme: MongoDB, MySQL

1. Daten generieren

MongoDB

Verwendet wurde JavaScript. Quelle: https://www.mongodb.com/docs/atlas/synthetic-data/
1 Mio. synthetische Datensätze wurden generiert.

```
// require the necessary libraries
const { faker } = require("@faker-js/faker");
const MongoClient = require("mongodb").MongoClient;

function randomIntFromInterval(min, max) { // min and max included
    return Math.floor(Math.random() * (max - min + 1) + min);
}

async function seedDB() {
    // Connection URL
    const uri = "mongodb://localhost:27017/";

    const client = new MongoClient(uri);

    try {
        await client.connect();
        console.log("Connected correctly to server");

        const collection = client.db("testdata").collection("data");

        // make a bunch of time series data
        let timeSeriesData = [];

        for (let i = 0; i < 1000000; i++) {
            const firstName = faker.person.firstName();
            const lastName = faker.person.lastName();
            console.log("Dataset: "+i);
            let newDay = {
                timestamp_day: faker.date.past(),
                cat: faker.lorem.word(),
                owner: {
                    email: faker.internet.email({firstName, lastName}),
                    firstName,
                    lastName,
                },
                events: [],
            };

            for (let j = 0; j < randomIntFromInterval(1, 6); j++) {
                let newEvent = {
                    timestamp_event: faker.date.past(),
                    weight: randomIntFromInterval(14,16),
                }
```

```
                newDay.events.push(newEvent);
            }
            timeSeriesData.push(newDay);
        }
        await collection.insertMany(timeSeriesData);

        console.log("Database seeded with synthetic data! :)");
    } catch (err) {
        console.log(err.stack);
    }
}

seedDB();
```

MySQL

Verwendet wurde JavaScript. Code wurde an MySQL angepasst.
1 Mio. synthetische Datensätze wurden generiert.

```
// require the necessary libraries
const faker = require("faker");
const mysql = require("mysql");

function randomIntFromInterval(min, max) { // min and max included
    return Math.floor(Math.random() * (max - min + 1) + min);
}

async function seedDB() {
    // Connection configuration
    const connection = mysql.createConnection({
        host: 'localhost',
        user: 'root',
        password: 'XXXX',
        database: 'testdata'
    });

    // Connect to MySQL
    connection.connect(function(err) {
        if (err) {
            console.error('Error connecting to MySQL database: ' + err.stack);
            return;
        }
        console.log("Connected correctly to MySQL database");

        // Create table if not exists
        const createTableQuery = `
            CREATE TABLE IF NOT EXISTS data (
                id INT AUTO_INCREMENT PRIMARY KEY,
                timestamp_day TIMESTAMP,
                cat VARCHAR(255),
                email VARCHAR(255),
                firstName VARCHAR(255),
                lastName VARCHAR(255),
                timestamp_event TIMESTAMP,
                weight INT
            )
        `;
        connection.query(createTableQuery, function(err, results, fields) {
            if (err) {
```

```
                console.error('Error creating table: ' + err.stack);
                return;
            }
            console.log("Table created successfully");
        });

        // Seed the database with synthetic data
        let inserts = [];
        for (let i = 0; i < 300000; i++) {
            const firstName = faker.name.firstName();
            const lastName = faker.name.lastName();
            console.log("Dataset: " + i);
            const timestamp_day = faker.date.past();
            const cat = faker.lorem.word();
            const email = faker.internet.email(firstName, lastName);
            const timestamp_event = faker.date.past();
            const weight = randomIntFromInterval(14, 16);

            inserts.push([
                timestamp_day,
                cat,
                email,
                firstName,
                lastName,
                timestamp_event,
                weight
            ]);
        }

        // Insert data into the database
        const insertQuery = "INSERT INTO data (timestamp_day, cat, email, firstName,
lastName, timestamp_event, weight) VALUES ?";
        connection.query(insertQuery, [inserts], function(err, results, fields) {
            if (err) {
                console.error('Error inserting data: ' + err.stack);
                return;
            }
            console.log("Database seeded with synthetic data! :)");
        });

        // Close connection
        connection.end(function(err) {
            if (err) {
                console.error('Error closing connection: ' + err.stack);
                return;
            }
            console.log("Connection closed");
        });
    });
}

seedDB();
```

2. Perfomance testen

Ein Skript wurde entworfen um die Geschwindigkeit der Datenbanken zu testen. Gemessen wird die Zeit die benötigt wird alle Datensätze auszugeben

```
const MongoClient = require('mongodb').MongoClient;
const mysql = require('mysql');

// MongoDB Verbindung URI
const mongoURI = "mongodb://localhost:27017/testdata";

// MySQL Verbindungsdetails
const mysqlConfig = {
    host: 'localhost',
    user: 'root',
    password: 'XXXX',
    database: 'testdata'
};

// Name der MongoDB-Sammlung und MySQL-Tabelle
const collectionName = "data";
const tableName = "data";

// Funktion zur Messung der Performance für MongoDB
async function measureMongoDBPerformance() {
    try {
        // Verbindung zur MongoDB herstellen
        const client = await MongoClient.connect(mongoURI, { useNewUrlParser: true,
useUnifiedTopology: true });
        const db = client.db();

        // Startzeit messen
        const startTime = new Date();

        // Alle Datensätze aus der MongoDB-Sammlung abrufen
        const collection = db.collection(collectionName);
        const documents = await collection.find({}).toArray();

        // Endzeit messen
        const endTime = new Date();
        const elapsedTime = endTime - startTime; // Zeitdifferenz berechnen

        // Ausgabe der Zeit
        console.log("MongoDB Performance:");
        console.log(`Anzahl der Datensätze: ${documents.length}`);
        console.log(`Dauer: ${elapsedTime} Millisekunden`);
        console.log();

        // Verbindung schließen
        await client.close();
    } catch (error) {
        console.error("Fehler beim Abrufen der MongoDB-Daten:", error);
    }
}

// Funktion zur Messung der Performance für MySQL
function measureMySQLPerformance() {
    // Verbindung zur MySQL-Datenbank herstellen
    const connection = mysql.createConnection(mysqlConfig);

    // Startzeit messen
    const startTime = new Date();

    // Alle Datensätze aus der MySQL-Tabelle abrufen
    connection.query(`SELECT * FROM ${tableName}`, (error, results) => {
```

```
        if (error) {
            console.error("Fehler beim Abrufen der MySQL-Daten:", error);
            return;
        }

        // Endzeit messen
        const endTime = new Date();
        const elapsedTime = endTime - startTime; // Zeitdifferenz berechnen

        // Ausgabe der Zeit
        console.log("\nMySQL Performance:");
        console.log(`Anzahl der Datensätze: ${results.length}`);
        console.log(`Dauer: ${elapsedTime} Millisekunden`);
        console.log();

        // Verbindung schließen
        connection.end();
    });
}

// Funktionen aufrufen
measureMongoDBPerformance();
measureMySQLPerformance();
```

Ausgabe:

```
MySQL Performance:
Anzahl der Datensätze: 1000000
Dauer: 4111 Millisekunden

MongoDB Performance:
Anzahl der Datensätze: 1000000
Dauer: 14950 Millisekunden
```

Das Ergebnis zeigt, um alle Datensätze auszugeben ist die MySQL Datenbank erheblich schneller.

Nach spezifischen Daten suchen

Ebenso wurde nach spezifischen Daten gesucht. In diesem Fall nach den Personen mit dem Nachnamen 'Tromp'.

```
MySQL Performance:
Anzahl der Datensätze: 2119
Dauer: 799 Millisekunden

MongoDB Performance:
Anzahl der Datensätze: 2066
Dauer: 841 Millisekunden
```

Obwohl in der MySQL Datenbank mehr Datensätze gefunden wurden, ist diese um 42ms schneller als MongoDB.

MongoDB indexieren

Nach dem indexieren des Feldes `"owner.lastName": "Tromp"` sah das Ergebnis wie folgt aus:

```
MongoDB Performance:
Anzahl der Datensätze: 2066
Dauer: 75 Millisekunden

MySQL Performance:
Anzahl der Datensätze: 2119
Dauer: 747 Millisekunden
```

Nach dem indexieren ist die MongoDB um **766ms** schneller.