Technische Hochschule Augsburg

Fakultät für Informatik

Dokumentation

für die Semesteraufgabe im Fach NoSQL

NoSQL: Umsetzung der Semesteraufgabe mit Google Firestore

erstellt von

Peter Fischer -

Leonelle Tifani Kommegne Kammegne -

Michael Mertl - 2209076

Gregor Pfister -

Jana Sophie Schweizer -

Technische Hochschule Augsburg

An der Hochschule 1
D-86161 Augsburg
T +49 821 5586-0
F +49 821 5586-3222
www.tha.de
info@tha.de

Inhaltsverzeichnis

1 Ausgangssituation 2

2 No-SQL Datenbank Entscheidung 3

3 Firestore-Emulator Aufsetzung 4

4 Abfragensprache-Entscheidung 5

5 Typsicherheit mit TypeScript 6

6 Ansatz-Struktur-Entscheidung 7

7 Herausforderungen bei den Read-Abfragen 8

8 Herausforderungen bei den Update-Abfragen 9

9 Herausforderungen bei den Delete-Abfragen 10

10 Fazit 11

1 Ausgangssituation

2 No-SQL Datenbank Entscheidung

3 Firestore-Emulator Aufsetzung

4 Abfragensprache-Entscheidung

5 Typsicherheit mit TypeScript

6 Ansatz-Struktur-Entscheidung

7 Herausforderungen bei den Read-Abfragen

Hier sind ein paar Einschränkungen, die wir bei der Umsetzung von Leseanfragen in Firestore hatten.

1. Kein JOIN: Wir können in SQL mehrere Tabellen mit JOIN miteinander verknüpfen. Das geht nicht direkt in Firestore. Der Leseanfrage **d** könnte man in SQL mit Join von Angebot und Kurs über die Kurs-Nr. abfragen. Wir müssen zuerst alle Angebote in Firestore laden. Danach laden wir für jedes Angebot das dazugehörige Kursdokument aus der Kurs-Sammlung einzeln nach. Dadurch gibt es viele einzelne Leseoperationen. Das hätte für eine umfangreichere Datenbank zu höhere Zeitkosten geführt.
2. Keine Aggregationen wie **COUNT, GROUP BY**: Firestore bietet keine Unterstützung für Aggregationen wie COUNT(*), AVG() oder GROUP BY. In Query **i** , in der alle Kurse mit mindestens zwei Teilnehmern gesucht werden, muss deshalb zunächst ein Zählerobjekt (teilnehmerCounter) im Code erstellt werden, das alle Teilnahmen durchläuft und pro Kurs-Angebot die Anzahl speichert. In einem relationalen Datenbank wäre dies eine einzige SQL-Zeile mit GROUP BY und HAVING COUNT(*) >= 2.

8 Herausforderungen bei den Update-Abfragen

9 Herausforderungen bei den Delete-Abfragen

10 Fazit