Advanced Database Techniques - Aufgabe 1 - Entwicklung eines Datenbankschemas

Jochen Schmidt
Student, Informatik
FH Würzburg-Schweinfurt
Fakultät Informatik
Matrikelnummer: 511xxx
Email: jochen.schmidt1@fhws.student.de

Lennard Rose
Student, Informatik
FH Würzburg-Schweinfurt
Fakultät Informatik
Matrikelnummer: 511xxx
Email: lennard.rose@fhws.student.de

Moritz Zeitler
Student, Informatik
FH Würzburg-Schweinfurt
Fakultät Informatik
Matrikelnummer: 5118094
Email: moritz.zeitler@fhws.student.de

Im Kurs 'Advanced Database Techniques' an der FHWS soll im Laufe des Kurses eine Datenbank, von den Studenten gewählt auf Herz und Nieren geprueft werden. Die Gruppe um Lennard Rose, Jochen Schmidt und Moritz Zeitler entschied sich daraufhin Elasticsearch als ebene diese zu verwenden, unter dem Bewusstsein das Elasticsearch eigentlich keine klassische Datenbank ist sondern eher eine Suchmaschine. Bei der ersten Aufgabe soll nun für ein freigewähltes Thema ein Datenbankschema erstellt werden. Dies wird im folgenden genauer beleuchtet.

Um das bereist vorgestellte Datenbankschema besser verwendbar zu machen, in Bezug auf Elasticsearch und Kibana, müssen einige Anpassungen vorgenommen werden. Die aus Relationalen Datenbanken bekannten 'joins', und somit das erstellen von vielschichtigen Queries über mehrere Tabellen, vergleichbar mit Indizes in Elasticsearch, müssen groesstenteils entfernt werden. SQL-Style joins sind in Elasticsearch so kostenintensiv das sie eigentlich grundsätzlich verboten sind [1]. Elasticsearch betrachtet die Welt daher auf einer eher horizontalen Ebene [2, Chapter 40], Objekte die in Elasticsearch gespeichert werden sollten daher geebnet werden (aus dem englischen 'flatten').
Hier gibt es verschiedene Optionen um diese flache Welt zu realisieren. Unter anderem könnte zum Beispiel auf Applikationsseite eine Art 'join' über mehrer Indizes implementiert werden. Da sich der Kurs aber nur im die Datenbank drehen soll, stellt dies keine ernsthafte Option dar. Hingegen sinnvoller sind hier entweder sogenannte 'nested objects' [2, Chapter 41] oder eine Art 'Parent-Child-Beziehung' zwischen den Objekten [2, Chapter 42]. Grundsätzlich gilt aber für beide Optionen das die Daten im gleichen Index abgelegt werden. Dies bedeutet das zwar die Abfragen an sich komplexer werden, aber nicht über mehrere Indizes hinweg gesucht wird.
An einigen stellen ist aber eine Denormalisierung, und somit eine redundante Datenhaltung sinnvoll. Dies ist die Beste, weil die schnellste Lösung, um relationale Modell in Elasticsearch darzustellen.
Jeder dieser Möglichkeiten bringt seine Vor- und Nachteile mit sich. Hier sind drei Metriken zu beachten: Indizierungs Zeit, Dauer der Suche und Datenmenge. Solange der Speicherverbrauch nicht entscheidend ist, kommt eine komplette Denormalisierung durchaus in Frage. Da bisher aber keine Hardware vorgaben bestehen nehmen wir an, dass diese begrenzt ist. Daher besteht nun die Frage ob fuer die Auflösung eben nun 'Parent-Child-Beziehungen' oder 'Nested-objects' verwendet werden soll. Grundsätzlich ist die 'Parent-Child-Beziehung' besser geeignet wenn die Indizierungszeit [2, Chapter 42] ausschlaggebend ist und vice versa 'Nested-Objects' besser für Suchzeiten.
Dies wird aber nun anhand von festen Beispielen Individuell bewertet:

1. Alle Verbindungstabellen die im relationalen Modell n:m Beziehungen auflösen werden ohne Ausnahme denormalisiert und 'redundant' abgelegt. Zum Beispiel kann man hier alle Entitäten im Bezug auf die Übungen nehmen wie 'exercise', 'exercise-entry' oder 'workout-

entry'.

2. Aufgrund von Geschwindigkeit der Indizierung ist für die Entitäten die im Bezug auf die Mitgliederverwaltung stehen eine 'Parent-Child-Beziehung' von Vortail. Dadurch können Änderungen an der Mitgliedschaft schneller verarbeitet werden. Dies ist durch direkten Kundenkontakt durchaus sinnvoll, da der Kunde so möglichst schnell seine Änderungen einsehen kann.
3. Im Bereich der Trainingspläne wird dann auf nested objects gesetzt. Hier sind vor allem schnelle Suchen nach passenden Workouts für den einzelnen Sportler von nöten.

2.1 Kibana spezifische Änderungen

3 Fazit

dynamic mapping muessen wir fuer nested objects ueberschreiben, type nested drauss machen sonst waere es type object

4 Lessons Learned

References

[1] Reference, E., 2021. Joining queries.

[2] Gormley, C., and Tong, Z., 2015. *Elasticsearch: The Definitive Guide - A Distributed Real-Time Search and Analytics Engine*. "O'Reilly Media, Inc.", Sebastopol.