Darstellung der Datenbank Firma in MongoDB

Lucia Vega Resto und Leon Lilje

1 Normalisierung versus Denormalisierung

MongoDB verfolgt grundsätzlich andere Designprinzipien als relationale Datenbanken. Während in der relationalen Welt die Normalisierung zur Vermeidung von Redundanz angestrebt wird, favorisiert MongoDB oft eine bewusste Denormalisierung zur Optimierung der Abfrageperformance.

Die Antwort auf die eingangs gestellte Frage lautet daher eindeutig: **Nein, die Datenbank muss nicht in Normalform stehen und Zusammenhänge müssen nicht über Fremdschlüsselbeziehungen repräsentiert werden.** Ein zentraler Unterschied zum relationalen Design liegt darin, dass MongoDB-Schemas primär von den erwarteten Abfragemustern bestimmt werden sollten. Während relationale Datenbanken ihre Struktur aus den Entitäten und deren logischen Beziehungen ableiten, steht bei MongoDB die Optimierung für spezifische Anwendungsfälle im Vordergrund.

2 Entwicklung alternativer Designansätze

2.1 Ansatz 1: Vollständige Denormalisierung

Der erste Ansatz verfolgt eine vollständige Denormalisierung, bei der alle Informationen zu einem Mitarbeiter in einem einzigen Dokument zusammengefasst werden. Gehaltsstufen und Abteilungsinformationen würden direkt in das Mitarbeiterdokument eingebettet, ebenso wie alle Kinder, Prämien und Maschinen als Arrays von Subdokumenten.

Vorteile: Dieser Ansatz ermöglicht es, alle Informationen zu einem Mitarbeiter mit einer einzigen Abfrage zu erhalten. Die Performance für Lesezugriffe wird dadurch optimal, da keine JOIN-ähnlichen Operationen erforderlich sind. Zudem entspricht diese Struktur der natürlichen Datengruppierung, da alle Informationen logisch zu einer Person gehören.

Nachteile: Die vollständige Denormalisierung führt zu erheblicher Datenredundanz. Abteilungs- und Gehaltsinformationen werden für jeden Mitarbeiter dupliziert. Änderungen an Stammdaten erfordern Updates in möglicherweise vielen Dokumenten, was zu Konsistenzproblemen führen kann.

2.2 Ansatz 2: Hybridmodell mit selektiver Referenzierung

Der zweite Ansatz kombiniert Einbettung und Referenzierung. Häufig geänderte Stammdaten wie Gehaltsstufen und Abteilungsinformationen werden in separaten Collections gespeichert und nur referenziert. Personenbezogene Daten wie Kinder, Prämien und Maschinen werden hingegen eingebettet.

Vorteile: Dieser Ansatz vermeidet die Nachteile beider Extreme. Stammdaten können zentral verwaltet und aktualisiert werden, während personenbezogene Daten weiterhin

effizient abgefragt werden können. Die Dokumentgröße bleibt überschaubar, und Konsistenzprobleme werden minimiert.

Nachteile: Vollständige Mitarbeiterinformationen erfordern mehrere Abfragen oder die Verwendung von Aggregation-Pipelines mit $lookup-Operationen. Dies kann die Anwendungslogik komplexer machen.

2.3 Ansatz 3: Vollständige Referenzierung

Ein dritter Ansatz würde die relationale Struktur weitgehend beibehalten und alle Beziehungen durch Referenzen abbilden. Dies entspricht am ehesten dem gewohnten relationalen Paradigma.

Vorteile: Maximale Flexibilität und Konsistenz, keine Datenredundanz.

Nachteile: Verlust der MongoDB-spezifischen Vorteile, komplexe Abfragen weiterhin erforderlich.

3 Bewertung anhand CRUD-Operationen

3.1 Create-Operationen

Das Anlegen neuer Mitarbeiter erfolgt in allen Ansätzen relativ unkompliziert.

3.2 Read-Operationen

Für das Anzeigen aller Mitarbeiter bietet der denormalisierte Ansatz die beste Performance, da alle Informationen in einem Dokument verfügbar sind. Der Hybridansatz erfordert zusätzliche Lookup-Operationen, bietet aber dennoch akzeptable Performance.

3.3 Update-Operationen

Hier zeigen sich die Schwächen der vollständigen Denormalisierung deutlich. Eine Gehaltserhöhung für eine bestimmte Gehaltsstufe würde Updates in allen betroffenen Mitarbeiterdokumenten erfordern. Dies ist nicht nur ineffizient, sondern birgt auch Risiken für die Datenkonsistenz. Der Hybridansatz ermöglicht hingegen zentrale Updates der Stammdaten.

3.4 Delete-Operationen

Das Löschen von Mitarbeitern ist in allen Ansätzen unkompliziert, da alle personenbezogenen Daten in einem Dokument gekapselt sind.

4 Entscheidung und Begründung

Ich wähle den **Hybridansatz** mit selektiver Referenzierung, da er die beste Balance zwischen Performance und Wartbarkeit bietet. Häufig geänderte Stammdaten können zentral verwaltet werden, während personenbezogene Daten effizient eingebettet bleiben. Der Ansatz skaliert besser als vollständige Denormalisierung und nutzt MongoDB-Funktionalitäten wie $lookup optimal aus.