RethinkDB

# Vergleich zwischen NoSQL (Not only SQL) und SQL

|  |  |
| --- | --- |
| **NoSQL** | **SQL** |
| **Modell**   * Nicht-Relational * Daten werden in JSON Dokumenten gespeichert | * Relational * Daten werden in Tabellen gespeichert |
| **Daten**   * Hohe Flexibilität, weil nicht jeder Eintrag dieselben Eigenschaften speichern muss * Neue Eigenschaften können leicht später hinzugefügt werden * Gut für Semi-strukturierte, komplexe, oder verschachtelte Daten | * Jeder Eintrag muss den vorgegebenen Eigenschaften entsprechen * Eigenschaften später hinzufügen ist oft kritisch und das Schema muss vielleicht geändert werden * Gut für strukturierte Daten |
| **Schema**   * Dynamische oder Flexibles Schema * Schema ist von der APP vorgegeben | * Striktes Schema * Schema muss in Synchronisation zwischen APP und Datenbank stehen |
| **Konsistenz, Verfügbarkeit und Performance**   * Abhängig von der APP * Die 3 Faktoren können wie gewünscht realisiert werden um den Ansprüchen der APP gerecht zu werden (Nach der CAP Theorie) | * Strenge Konsistenz * Konsistenz ist streng gegenüber den anderen Faktoren priorisiert |
| **Skalierung** **(Datendarstellung)**   * Daten werden horizontal dargestellt | * Daten werden vertikal dargestellt |

RethinkDB verwendet **NoSQL(Not only SQL)** zur Datenspeicherung.

## CAP Theorie (Consistency, Availability, Partition Tolerance)

Besagt, dass es in einem verteilten System unmöglich ist, gleichzeitig alle 3 Eigenschaften Konsistenz, Verfügbarkeit und Partitionstoleranz zu garantieren

Eines der 3 Eigenschaften muss also geopfert werden, um die anderen 2 garantieren zu können.

# Warum RethinkDB verwenden?

Diese Datenbank ist sehr gut mit Node.js kompatibel und bietet aufgrund der NoSQL Variante, eine flexibles Schema. RethinkDB ist gut geeignet für schnelle Speicher- und Lesevorgängen, was von Vorteil ist für eine Echtzeit APP.

Außerdem erkennt diese Datenbank, ob Datensätze geändert werden. Durch dieses Feature ist es mögliche Datenänderungen sofort in Echtzeit zu veranschaulichen.

Die gesamte Datenbank kann auch über einen Webclient überwacht und verwaltet werden. Über diesen lassen sich Statistiken auslesen wie z.B.: die Momentane Auslastung oder ob Fehler aufgetreten sind. Aber auch Tabellen und Datenbanken selber lassen sich direkt erstellen, bearbeiten und löschen.

Besonders ist auch die einfache und mächtige Abfragesprache: ReQL

# ReQL Query Language

Wie eine traditionelle Datenbank, arbeitet RethinkDB mit Tabellen. Aber gespeichert werden sämtliche Dateneinträge in JSON Dokumenten.

Beispiel:



Ein JSON Dokument beginnt immer mit einer geschwungenen Klammer und endet auch mit einer. Dazwischen befinden sich **Keys** mir ihren zugeordneten **Werten**, durch Doppelpunkt getrennt.

Die **Keys** sind **Strings** und die einzelnen Werte können folgenden Typ annehmen:

* Nummer (jede reale Zahl)
* String (beliebige UTF-8 Zeichenkette)
* Binäres Objekt (z.B.: File oder Bild)
* Datum/Zeit (auf Millisekunden genau)
* Boolean (richtig/falsch)
* Null (kein Wert vorhanden)

Mithilfe von ReQL können also Abfragen in RethinkDB, mit JSON Dokumenten getätigt werden.

## Abfragen

Als erstes wird die Abfrage mit dem Buchstaben **“r“** begonnen und anschließend folgt direkt der   
**“.“** Operator.

Danach folgt die Datenbank, die man verwenden möchte z.B.: **db(“myDb“)**

Ist keine bestimmte Datenbank ausgewählt, so ist die Standardmäßige: **test**

Als nächstes folgt wieder der Operator und darauf die Tabelle: **table(“myTable“)**

Die komplette Abfrage sieht wie folgt aus:



Das Ergebnis sind alle Dokumente der ausgewählten Tabelle.

Mit dieser Schreibweise funktionieren sämtliche Datenbankabfragen. Folgend werden 2 Dokumente in die Datenbank eingetragen:

