MongoDB – Es geht auch ohne SELECT

Ein Blick auf das in Kürze erscheinende neue DAM

Klaus Schrödl

Warum NoSQL

Wie kam es zu dem Bedürfnis, NoSQL-Datenbanken zu entwickeln?

Mehrere Faktoren

1. **Skalierbarkeit**
Traditionelle relationale Datenbanken waren oft nicht in der Lage, mit dem Wachstum der Datenmengen und der Anzahl von Benutzern Schritt zu halten.
2. **Flexibles Datenmodell**
Relationale Datenbanken erfordern ein starres Schema, das im Voraus definiert werden muss, was zu Problemen bei variablen Datenstrukturen führen kann.
3. **Big Data und Echtzeitanwendungen**
Die Notwendigkeit, große Datenmengen schnell zu verarbeiten und Analysen in Echtzeit durchzuführen, erforderte leistungsfähigere Datenbanklösungen.
4. **Verteilte Datenbanken**
Die zunehmende Globalisierung erforderte Datenbanken, die über mehrere Standorte oder Rechenzentren verteilt werden können.
5. **Moderne Anwendungsanforderungen und wachsende Bedeutung**
Die steigenden Anforderungen moderner Anwendungen erforderten Datenbanken, die flexibler, skalierbarer und leistungsfähiger sind als traditionelle relationale Datenbanken.

NoSQL-Datenbanken wurden entwickelt, um diese Herausforderungen anzugehen und neue Möglichkeiten für Datenmanagement und -verarbeitung in einer zunehmend digitalen Welt zu schaffen.

Relationale versus NoSQL Datenbanken
Schlüsselunterschiede

1. Datenmodell
SQL-Datenbanken verwenden ein strukturiertes Datenmodell, das aus Tabellen mit vordefinierten Schema besteht. Daten werden in Zeilen und Spalten organisiert.

NoSQL-Datenbanken verwenden ein flexibleres Datenmodell. Sie können dokumentenorientiert sein, Schlüssel-Wert-Paare, spaltenbasiert oder grafenorientiert speichern.

2. Skalierbarkeit
SQL-Datenbanken skalieren typischerweise vertikal, indem sie leistungsstärkere Hardware verwenden.

NoSQL-Datenbanken sind für horizontale Skalierbarkeit ausgelegt. Sie können problemlos auf mehrere Server verteilt werden, um die Last zu verteilen.

3. Konsistenz und Konsistenzmodelle
SQL-Datenbanken betonen die Konsistenz der Daten. Transaktionen werden verwendet, um sicherzustellen, dass Daten stets in einem konsistenten Zustand bleiben.

NoSQL-Datenbanken bieten oft flexible Konsistenzmodelle, die je nach Anforderungen der Anwendung angepasst werden können. Einige NoSQL-Datenbanken priorisieren Verfügbarkeit und Partitionstoleranz über Konsistenz.

4. Schema-Design
SQL-Datenbanken erfordern ein vorgängiges Schema-Design, in dem die Struktur der Daten im Voraus definiert werden muss.

NoSQL-Datenbanken bieten oft ein dynamisches Schema, das es ermöglicht, Daten flexibel zu speichern, ohne ein festes Schema zu benötigen.

MongoDB und NoSQL

Merkmale I

Hintergrund & Facts
Beginn der Entwicklung 2007 in C++ durch 10gen, heute mongoDB Inc., seit 2018 als freie Software unter der SSPL verfügbar, NASDAQ: MDB
Website: https://www.mongodb.com/

Verfügbare OS:
Linux, Windows, MacOS, OpenBSD

Dokumentenorientiert
MongoDB speichert Daten in flexiblen JSON-ähnlichen Dokumenten, die als BSON (Binary JSON) bezeichnet werden. Diese Dokumente können komplexe Strukturen enthalten und ermöglichen eine einfache Darstellung von Hierarchien und verschachtelten Datenstrukturen.

Skalierbarkeit
MongoDB ist für horizontale Skalierbarkeit ausgelegt und kann problemlos auf mehrere Server verteilt werden, um große Datenmengen zu verarbeiten.

MongoDB und NoSQL
Merkmale II

Flexibles Schema
MongoDB bietet ein dynamisches Schema, das es erlaubt, Daten flexibel zu speichern, ohne ein festes Schema vorher definieren zu müssen. Dies ermöglicht eine schnelle Iteration und Entwicklung von Anwendungen.

Leistungsstarke Abfragesprache
MongoDB bietet eine leistungsstarke Abfragesprache, die es erlaubt, komplexe Abfragen über die Daten zu formulieren, einschließlich Aggregationen, Sortierungen und Indizierung.

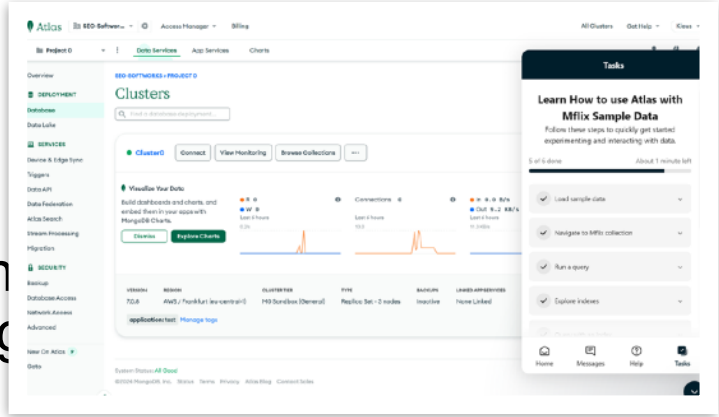
Verwendung in der Praxis
MongoDB wird häufig in Webanwendungen, Content-Management-Systemen, Big-Data-Anwendungen und anderen Szenarien eingesetzt, in denen flexible Datenmodellierung und horizontale Skalierbarkeit erforderlich sind.

Unterstützung & Treiber
Für alle gängigen Sprachen und Entwicklungsumgebungen vefügbar: C#, C, C++, Java, PHP, TypeScript, Python, Ruby etc.

Omnis Studio & MongoDB

Neues DAM - Erweiterung auf NoSQL

Verfügbarkeit
Voraussichtlich im Herbst 2024, wird auch in **Omnis Studio Now** zur Verfügung stehen.

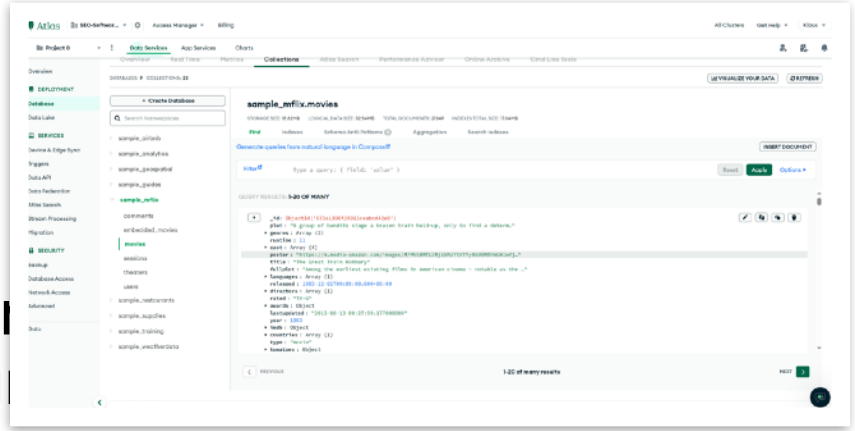


Einstieg in MongoDB
Möglichkeit des Downloads und Installation auf eigenem Server. Für erste Schritte – auch mit Omnis Studio – ist jedoch die Verwendung der MongoDB-eigenen Cloud-Lösung **Atlas** zu empfehlen

https://www.mongodb.com/products/platform/cloud

MongoDB - Atlas

Einstieg, Vorbereitung



Anlegen Atlas-Account

Neben den üblichen Informationen muss auch eine (am besten feste) IP-Adresse für den Zugriff registrier werden, nur von dieser aus ist Zug auf die Datenbank möglich.

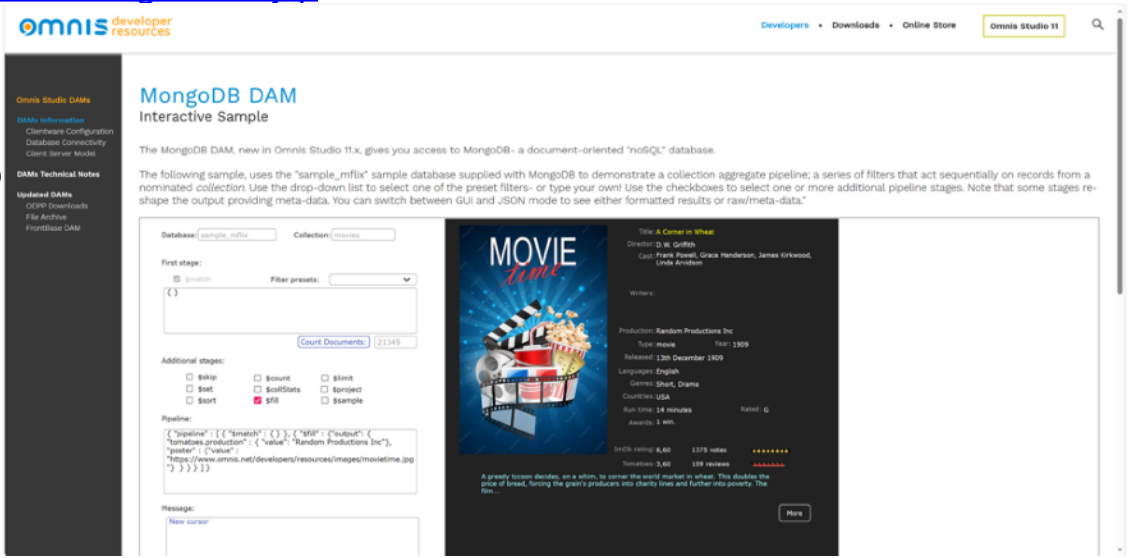
Cloud-Administration

Zahlreiche Demo-Datenbanken verfügbar. Ebenso Tutorials und eine umfangreiche, klar strukturierte Dokumentation

Omnis Studio – MongoDB Sample Application
Live Demo

URL:

https://www.omnis.net/developers/resources/dams/mongodemo.jsp



Verwendet wird die sample_mflix/movies Datenbank.

Aufbau der Verbindung über das Omnis Studio Objektmodell:
DAM-Object, Session-Object & Session-Pool
Verwendung eines Connection-Strings zum Aufbau der Verbindung:

```
$extobjects.MONGODBDAM.$objects.MONGODBSESS.$makepool('pool1',12,'mongodb+srv://myname:mypassword@clusterX.oojamxX.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0','','','Startup_Task.$myinit') Returns lPoolRef
```

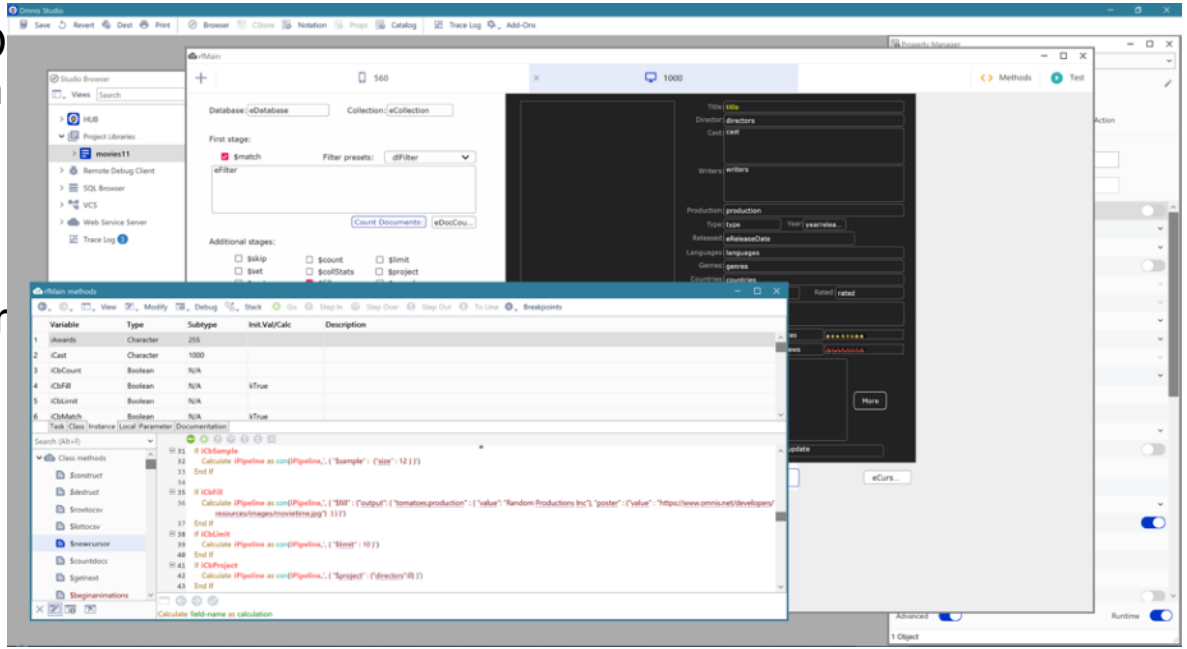
Omnis Studio – MongoDB Sample Application

Live Demo

Syntax und Zugriff auf die MongoD
Datenbank deutlich anders als von
SQL her gewohnt.
Stichworte:
- Pipeline
- Cursor

Queries müssen in JSON formulier
werden, z.B.

```
{genres : ["Drama", "Fantasy"]}
```



omnis studio

Besteht Bedarf für weitere NoSQL-Dams? Anregungen?

Vielen Dank!