



mongoDB®

MongoDB

Hannover, 29.09..2023



































Agenda

- Einführung NoSQL
- MongoDB
- Praktische Übung

Was ist NoSQL ?



Was ist NoSQL ?

Rang			DBMS	Datenbankmodell
Sep 2023	Aug 2023	Sep 2022		
1.	1.	1.	Oracle 	Relational, Multi-Model 
2.	2.	2.	MySQL 	Relational, Multi-Model 
3.	3.	3.	Microsoft SQL Server 	Relational, Multi-Model 
4.	4.	4.	PostgreSQL 	Relational, Multi-Model 
5.	5.	5.	MongoDB 	Document, Multi-Model 
6.	6.	6.	Redis 	Key-value, Multi-Model 
7.	7.	7.	Elasticsearch	Suchmaschine, Multi-Model 
8.	8.	8.	IBM Db2	Relational, Multi-Model 
9.	 10.	 10.	SQLite 	Relational
10.	 9.	 9.	Microsoft Access	Relational
11.	11.	 13.	Snowflake 	Relational
12.	12.	 11.	Cassandra 	Wide column, Multi-Model 
13.	13.	 12.	MariaDB 	Relational, Multi-Model 
14.	14.	14.	Splunk	Suchmaschine
15.	 16.	 16.	Microsoft Azure SQL Database	Relational, Multi-Model 
16.	 15.	 15.	Amazon DynamoDB 	Multi-Model 

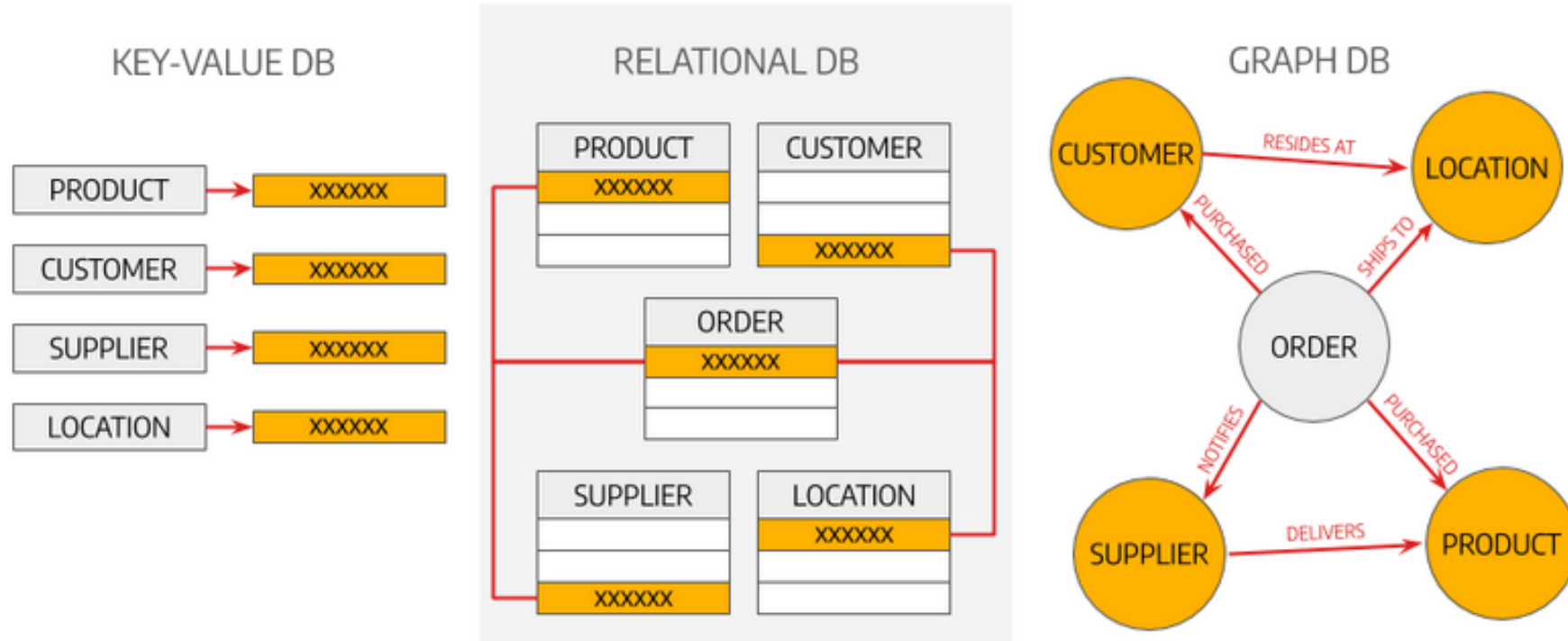
NoSQL ist ein Oberbegriff für eine Vielzahl von Datenbanken, die nicht das relationale Modell von SQL verwenden.

Relationale Systeme machen aufgrund von über 50 Jahren Erfahrung und konstanter Entwicklung immernoch einen Großteil aller eingesetzten Lösungen aus.

NoSQL-Datenbanken sind häufig für Anwendungen mit großen Datenmengen und/oder komplexen Datenstrukturen geeignet.

Sinnvolle Ergänzung zu relationalen DB

Ein riesen Vorteil an NoSQL-DB ggü relationalen ist die Skalierbarkeit. Eher ein FiSi-Thema aber es sollte dennoch erwähnt werden, wir können Relationale DB potentiell endlos skalieren. Bei Relationalen: muss 1 Server das komplette System tragen, bei den moderneren kann man einfach endlos neue Datenbankserver hinzufügen.



Im Gegensatz zu relationalen DB, können NoSQL-DB Daten losgelöst von Tabellenschemata speichern. Dies erhöht die Performance ungemein. Das heißt allerdings auch, dass Daten an kein Format gebunden sind. So können in diesen DB die Informationen als JSON oder XML-Dokument gespeichert werden, als Key-Value-Paar oder als ganze Graphen mit Knoten & Kanten.

ID	Nachname	Vorname	Geburtsdatum	Gehalt
1	Müller	Hans	21.04.1998	2700
2	Maier	Thea	03.11.2001	2220
3	Schulz	Simone	12.06.2002	1970

Zeilenorientierte Speicherung

1, Müller, Hans, 21.04.1998, 2700; 2, Maier, Thea, 03.11.2001, 2220; 3, Schulz, Simone, 12.06.2002, 1970;

Spaltenorientierte Speicherung

1, 2, 3; Müller, Maier, Schulz; Hans, Thea, Simone; 21.04.1998; 03.11.2001, 12.06.2002; 2700, 2220, 1970;

Gehört haben sollte man im Zusammenhang mit relationalen Datenbanken ja zumindest schonmal “ACID”

Zur Auffrischung: ATOMICITY, CONSISTENCY, ISOLATION, DURABILITY

Was SQL-Datenbanken ja umsetzen müssen bzw. Sollten. Bei NoSQL wird jetzt gesagt, dass ACID-Regeln per se erstmal nicht unterstützt werden, sondern das BASE-Modell, “Basically Available, Soft State, Eventually Consistent” Das heißt grob zsm gefasst: Verfügbarkeit vor Konsistenz.

Bei Isolation bspw. Werden die Bearbeitungen von Daten nicht abgekapselt durchgeführt, sondern der Letzte d.d.Daten.bearbeitet gewinnt.

SOFT STATE besagt, dass wegen der „Eventual Consistency“, also der nur mäßigen Konsistenz der Daten, diese, ohne zutun von Nutzern, sich ändern können. Irgendwann sind die Daten konsistent, nur halt nicht immer.

Atomicity
Consistency
Isolation
Durability

vs

Basically **A**vailable
Soft State
Eventually Consistent

Wir können also Effizient, schnell eine große Menge an Daten behandeln, ohne dass wir irgendwelche Tabellen brauchen. Dieser performance-boost kostet allerdings, wie erwähnt regelmäßig viele der eben genannten ACID-Punkte.

Ich habe ja eben gezeigt, dass es bei NoSQL-DBs verschiedenste Modelle gibt und demnach auch unterschiedliche Wege, wie Daten gespeichert werden. Bei MongoDB werden sie im BSON-Format gespeichert, was eine Binäre Umwandlung o. Variante von JSON ist. Effektiv kann man aber beim Programmieren von einem JSON-Objekt ausgehen.

The screenshot shows the MongoDB Compass web interface. The top bar indicates the connection to 'localhost:27017/nuxeo.default' and the version 'MongoDB 3.4.4 Community'. The main panel displays the 'nuxeo.default' database with 189.9k documents and 19 indexes. The 'DOCUMENTS' tab is selected, showing a filter bar with the query '{ "filter" : "example" }'. Below the filter, it states 'Query returned 189933 documents. Displaying documents 1-40'. A green 'INSERT DOCUMENT' button is visible. The document view shows a JSON-like structure with fields such as 'ecm:primaryType', 'ecm:id', 'ecm:lifeCycleState', 'dc:contributors', '_id', 'ecm:racl', 'icon', 'dc:creator', 'ecm:parentId', 'ecm:ancestorIds', 'dc:modified', 'ecm:minorVersion', 'dc:lastContributor', 'content', 'data', 'name', 'mime-type', 'length', 'ecm:name', 'ecm:majorVersion', 'filename', 'ecm:lifeCyclePolicy', 'size', 'dc:created', 'dc:title', 'ecm:primaryType', 'ecm:id', and 'ecm:lifeCycleState'.

Basic Mongo DB

<code>db</code>	Show name of current database
<code>mongod</code>	Start database
<code>mongo</code>	Connect to database
<code>show dbs</code>	Show databases
<code>use db</code>	Switch to database <code>db</code>
<code>show collections</code>	Display current database collections

Create

<code>insert(data)</code>	insert document(s) returns write result
<code>insertOne (data, options)</code>	insert one document
<code>insertMany(data, options)</code>	insert many documents
<code>insertMany([{}],{},{})</code>	needs square brackets

Read

<code>db.collection.find()</code>	Display documents from <code>collection</code>
<code>find(filter, options)</code>	find all matching documents
<code>findOne(filter, options)</code>	find first matching document

Update

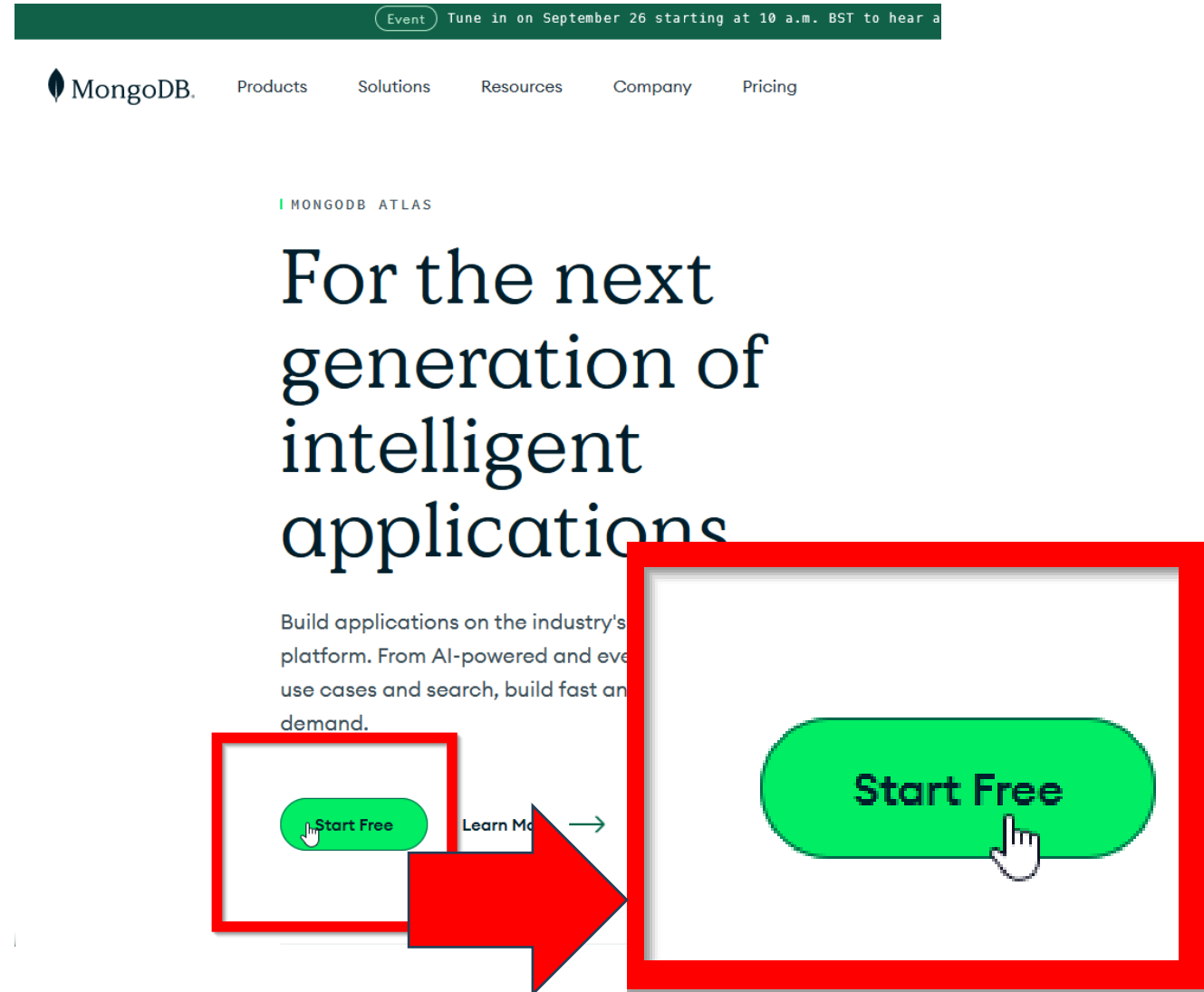
<code>updateOne(filter, data, options)</code>	Change one document
<code>updateMany(filter, data, options)</code>	Change many documents
<code>replaceOne(filter, data, options)</code>	Replace document entirely

Delete

<code>deleteOne(filter, options)</code>	Delete one document
<code>deleteMany(filter, options)</code>	Delete many documents

Praktische Übung

<https://account.mongodb.com/account/register>



Quelle: <https://d2slcw3kip6qmk.cloudfront.net/marketing/pages/chart/activity-diagram-for-login-UML/activity-diagram-for-login-UML-650x797.png>

Password*



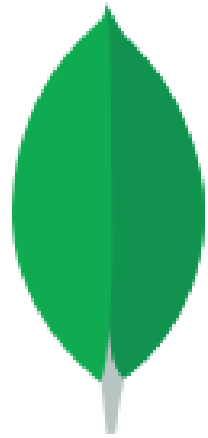
☐ I agree to the **Terms of Service** and **Privacy Policy**.

Create your Atlas account

or

 Sign up with Google

Sign in



mongoDB®