MongoDB vs MySQL

MySQL – basis

MySQL is een managementsysteem voor relationele databases. Ze werd in 1995 ontwikkeld door MySQL AB en later overgenomen door Oracle Corporation. Ze is ontwikkeld met gebruik door internettoepassingen op het oog. MySQL is compatibel met en gratis beschikbaar voor alle gangbare besturingssystemen. Informatie wordt opgeslagen in tabellen met rijen en kolommen. De community edition (MySQL CE) wordt gezien als het meest gebruikte databasesysteem ter wereld.

MongoDB – basis

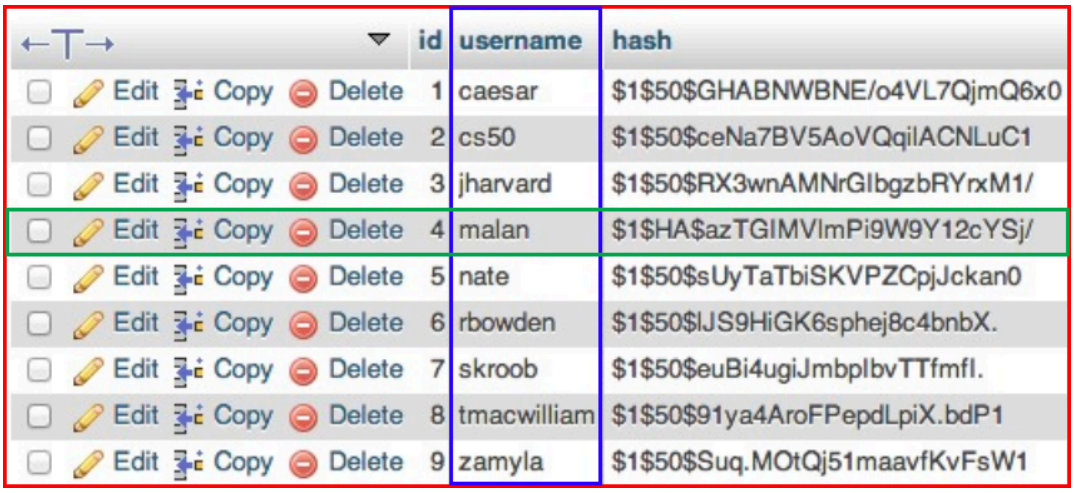
MongoDB is een NoSQL databasesysteem, ontwikkeld door MongoDB, Inc sinds 2007 met hun eerste openbare versie in 2009. Het is volledig open-source beschikbaar op GitHub. MongoDB wordt vooral gebruikt voor big-data en real-time toepassingen. Informatie wordt opgeslagen in collecties met documenten en velden. Ook MongoDB is compatibel met alle gangbare besturingssystemen, maar wordt vooral op UNIX-systemen gebruikt.

Jargon

MySQL en MongoDB gebruiken andere terminologie voor gelijkende concepten. De meest voorkomende vindt u hier terug, met nadien een uitleg:

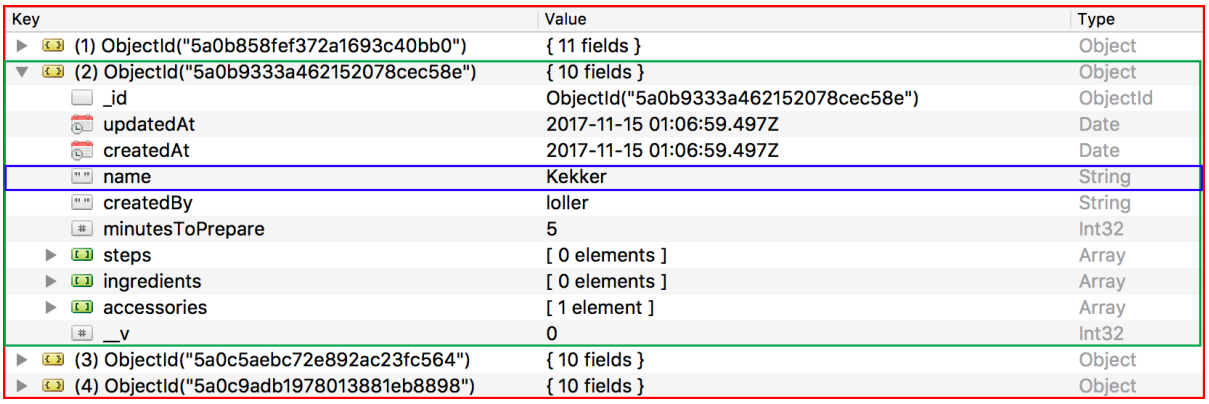
Wat men in MySQL een *Table* of *Tabel* noemt, heet in MongoDB een *Collection* of *Collectie*. Een *Row* of *Rij* wordt een *Document*. Ten laatste krijg de term *Column* of *Kolom* bij MongoDB de noemer *Field* of *Veld*. Hier vindt u nog een korte uitleg over elke term, aan de hand van foto's:

MySQL



	id	username	hash
Edit Copy Delete	1	caesar	$1$50$GHABNWBNE/o4VL7QjmQ6x0
Edit Copy Delete	2	cs50	$1$50$ceNa7BV5AoVQqilACNLuC1
Edit Copy Delete	3	jharvard	$1$50$RX3wnAMNrGIbgzbRYrxM1/
Edit Copy Delete	4	malan	1HA$azTGIMVlmPi9W9Y12cYSj/
Edit Copy Delete	5	nate	$1$50$sUyTaTbiSKVPZCpjJckan0
Edit Copy Delete	6	rbowden	$1$50$lJS9HiGK6sphej8c4bnbX.
Edit Copy Delete	7	skroob	$1$50$euBi4ugiJmbplbvTTfmfI.
Edit Copy Delete	8	tmacwilliam	$1$50$91ya4AroFPepdLpiX.bdP1
Edit Copy Delete	9	zamyla	$1$50$Suq.MOtQj51maavfKvFsW1

MongoDB



Key	Value	Type
(1) ObjectId("5a0b858fef372a1693c40bb0")	{ 11 fields }	Object
(2) ObjectId("5a0b9333a462152078cec58e")	{ 10 fields }	Object
_id	ObjectId("5a0b9333a462152078cec58e")	ObjectId
updatedAt	2017-11-15 01:06:59.497Z	Date
createdAt	2017-11-15 01:06:59.497Z	Date
name	Kekker	String
createdBy	loller	String
minutesToPrepare	5	Int32
steps	[0 elements]	Array
ingredients	[0 elements]	Array
accessories	[1 element]	Array
__v	0	Int32
(3) ObjectId("5a0c5aebc72e892ac23fc564")	{ 10 fields }	Object
(4) ObjectId("5a0c9adb1978013881eb8898")	{ 10 fields }	Object

Schaalbaarheid

MongoDB staat bekend voor zijn gebruik in big-data toepassingen. Dit wilt ook zeggen dat schaalbaarheid redelijk belangrijk is: als de hoeveelheid data de capaciteit van de server dreigt te bereiken, moet hier een oplossing voor zijn. Vaak is de eerste oplossing, zowel bij MySQL als bij MongoDB, vertical scaling. Dit doe je door meer kracht aan de database, of de server waarop die draait, te geven. Zo kan je de server krachtiger maken door er bijvoorbeeld een extra of een betere CPU bij te steken of de hoeveelheid RAM op te waarderen.

Vertical scaling heeft echter een limiet: een server kan maar zo krachtig worden voor hij niet meer verder kan. Wanneer je dit punt bereikt, ga je over op horizontal scaling: meer servers installeren, en de database over deze verschillende servers simultaan laten lopen.

Vertical scaling in MySQL is relatief complex en moet met het oog op het type data gebeuren. Vertical scaling in MongoDB is vrijwel ingebouwd door middel van *sharding*. Bij sharding wordt de database opgesplitst op basis van een index. Als je een database hebt met records met een id van 1-1000, kan deze bijvoorbeeld opgesplitst worden in 2 databases van 1 tot 500, en van 501 tot 1000. Als de applicatie of user document 300 nodig heeft, spreekt hij de eerste database aan enzovoort.

Querytaal

Bij MySQL worden database opgemaakt, data opgehaald en bewerkt door middel van de taal SQL. MongoDB gebruikt een querytaal die gebaseerd is op JavaScript. Als ik de gegevens van de gebruiker met naam *Adriaan* wil ophalen uit een database, zou ik volgende code gebruiken:

MySQL

```
SELECT * FROM users WHERE name = ‘Adriaan’;
```

MongoDB

```
db.users.find( { name : ‘Adriaan’ } );
```

Benchmarks (vanuit het opzicht van node.js)

Als je databasesystemen vergelijkt, mogen niet enkel de definities, het gebruik en de taal vergeleken worden, maar ook de snelheid en performance. Daarom schrijf ik enkele benchmarking tests die twee systemen in de praktijk tegen elkaar afwegen.

Ik schrijf deze tests in node.js met gebruik van de npm packages mysql voor MySQL en mongoose voor MongoDB. Lees- en schrijfsnelheden kunnen dus door het platform en deze packages beïnvloed worden, maar ik verwacht dat de tests toch nuttige informatie zullen opleveren, vooral voor node.js ontwikkelaars, maar ook voor ontwikkelaars die niet met node.js werken. Node.js is een redelijk goed platform om deze tests op te draaien, omdat ze alle beschikbare resources van de machine probeert te gebruiken, als ze niet gelimiteerd is.

De tests draaien op dezelfde machine, met dezelfde achtergrondprocessen draaiende (Microsoft Word, GitHub Desktop, Visual Studio Code, mysqld, mongod). De twee databases zullen allebei lokaal draaien, om foute voorstellingen in de resultaten door internet-latency te vermijden. Ook zorg ik ervoor dat de database en tabel of collectie volledig leeg is voor een schrijfopdracht.

De specificaties van de machine zijn als volgt:
MacBook Pro (13-inch, 2016)
CPU: 2,9 GHz Intel Core i5
RAM: 8 GB 2133 MHz LPDDR3
Opslag: SSD 256GB

Test_1 – Schrijfsnelheid: 500.000 documenten/rijen toevoegen

Ik schrijf een script dat nagenoeg dezelfde (randomised) data zowel naar een MongoDB als naar een MySQL database schrijft (niet tegelijk). Ik maak een Mongoose schema aan voor MongoDB, en creëer een MySQL database die voor dezelfde data past.

Ik houd de momenten bij dat het script eindigt en start in milliseconden, en geef het verschil op het einde weer. Ook toon ik hoe lang het duurt om bij elke 10% te geraken (dus bij 10%, 20%, 30%, …) om te checken of het toevoegen van data trager wordt als er al veel entries in de database zitten. Met deze informatie kan ik in Microsoft Excel een grafiek maken van de snelheid waarmee data wordt toegevoegd.

Ten laatste voer ik het script 5 keer uit per databasesysteem en neem het gemiddelde, om discrepanties te vermijden en een goed richtnummer te krijgen.

De data die ik aanmaak en doorstuur naar de database:

```
naam = getRandomString(10); //random string van 10 tekens
hoeveelheid = getRandomInt(0,1000); //random nummer tussen 0 en 1000
opmerking = getRandomString(150); //random string van 150 tekens
kleur = getRandomKleur(); /*random kleur uit ["rood","groen","blauw","paars",
"geel","oranje"]*/
```

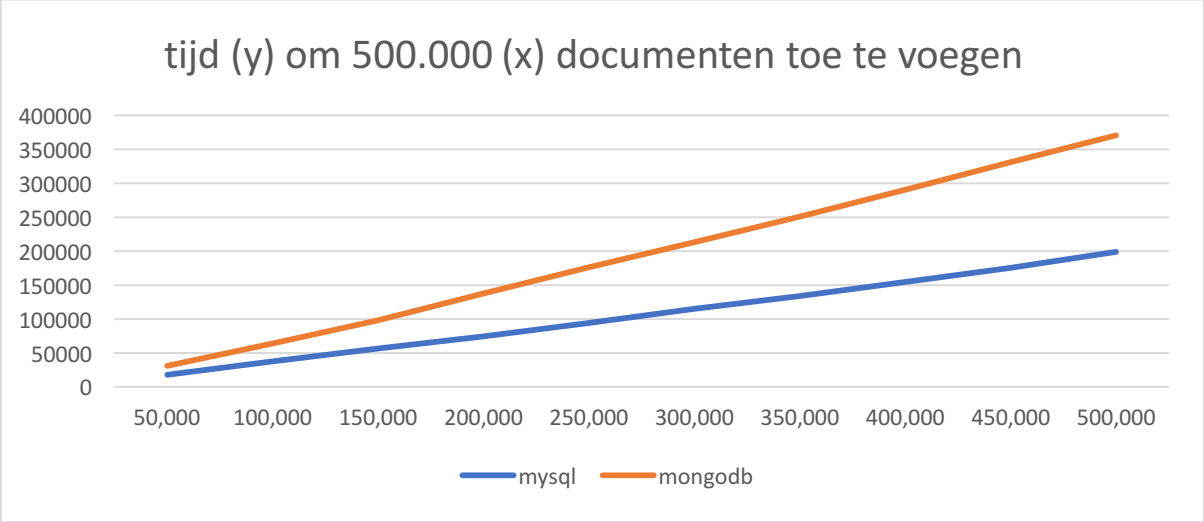
Ik gebruik deze random data zodat ik in verdere tests deze dataset verder kan gebruiken om performantietests op uit te voeren.

Tijden:

	MongoDB	**MySQL**
1	371146ms	201899ms
2	367990ms	199256ms
3	378392ms	208586ms
4	375207ms	203211ms
5	369431ms	200538ms
Gemiddelde	**372433ms**	**202698ms**

Zoals je ziet, is MySQL in deze use-case bijna 2 keer zo snel als MongoDB. MySQL doet er gemiddeld 3 minuten en 22 seconden over om 500.000 rijen toe te voegen aan een tabel. MongoDB doet over hetzelfde gemiddeld 6 minuten en 12 seconden. Dit wilt zeggen dat MySQL bijna 2.500 rijen per seconde schrijft, en MongoDB amper 1350.

Tijd om op tussenpunten te geraken:



In deze grafiek zie je ook dat MongoDB er na een tijd langer over doet om documenten toe te voegen, wanneer er dus al een groot aantal documenten in de database zit. Bij MySQL lijkt de schrijfsnelheid constant.

MySQL is dus sneller wanneer je veel informatie wilt toevoegen aan een database. Dit gaat tegen mijn oorspronkelijke verwachtingen in, aangezien MongoDB als een big data booster wordt beschreven.

Test_2 – Leessnelheid met en zonder indexering:

Na de vorige test zitten er zowel in de MongoDB database als in de MySQL database 500.000 documenten of rijen. Uit deze 500.000 zijn er ongeveer 500 waar de variabele '*hoeveelheid*' gelijk is aan één bepaald getal tussen 0 en 1000. Aangezien deze verspreid zitten, moet de database sowieso de volledige collectie of tabel doorlopen om bv alle entries te vinden met *hoeveelheid = 700 (bij MongoDB bestaan er 512, bij MySQL 506)*.

Daarom voer ik volgende queries uit om de leessnelheid van de databasesystemen te meten:

MongoDB

```
db.getCollection('datas').count({hoeveelheid:700})
```

MySQL

```
SELECT * FROM research_vgl_test_1.data WHERE hoeveelheid = 700;
```

Om deze query uit te voeren doet MongoDB er gemiddeld 330ms over. MySQL doet er gemiddeld 133ms over. MySQL lijkt dus een pak sneller. Maar, als ik in MongoDB met het commando `db.datas.createIndex({hoeveelheid:1})` een index op '*hoeveelheid*' zet, doet de MongoDB query er nog maar gemiddeld 2ms over.

De leessnelheid van MongoDB is dus, indien er gebruikt gemaakt wordt van indexering, ongelooflijk veel sneller dan die van MySQL. Wanneer geen indexering wordt gebruikt, is de MySQL query bijna 3 keer sneller. Wanneer er wel geïndexeerd is, is de MongoDB query ongeveer 60 keer sneller. Ik zie ook dat wanneer ik nog meer data aan de databases toevoeg door de eerste test opnieuw te laten lopen, de zoeksnelheid van MySQL nog trager wordt, en die van MongoDB nagenoeg hetzelfde blijft (enkel in geval van indexering).

Test_3 – Mass updating

Om de updatesnelheid van MySQL en MongoDB te testen, wil ik de naam van alle entries waar *'hoeveelheid'* = 700 aanpassen. De queries die ik hiervoor gebruik vindt u hieronder:

MongoDB

```
db.getCollection('datas').updateMany({hoeveelheid:500}, {$set: {naam: "adriaan"}})
```

MySQL

MongoDB doet er zonder indexering ongeveer 413ms over, met indexering gemiddeld 13ms. MySQL heeft gemiddeld 753ms nodig om deze query uit te voeren. Bij updates op veel data lijkt MongoDB zonder indexering dus iets sneller.

Conclusie

MongoDB, hét big-data databasesysteem, lijkt te worstelen met veel data snel te schrijven. MySQL, de gevestigde waarde, lijkt hier helemaal geen problemen mee te hebben. In leessnelheid is MongoDB sneller, als er juist gebruik gemaakt wordt van indices. Bij mass updates is MongoDB ook iets sneller wanneer er geen indexering is toegepast, en opnieuw veel sneller wanneer die er wel is.

Men moet er wel rekening mee houden dat de meeste van deze queries zeer simpel waren. Volgens onderzoek blijkt MySQL veel beter en sneller te zijn in het uitvoeren van complexere queries. Ook moet men onthouden dat het over 'maar' 500.000 entries gaat. Hiermee hebben we het bijlange na niet over *big data*.

Mijn persoonlijke mening over MySQL is na lange tijd opnieuw iets steviger. Ik schreef het vaak af als een oubollig systeem dat aan vervanging toe was, maar dit blijkt in veel gevallen toch fout. Ik wil wel verder gaan met het verkennen van de 2 systemen. Misschien heb ik door mijn beperkte kennis enkele foutjes gemaakt, en zou een expert volledig andere resultaten bekomen als alle code 100% performant zou zijn, of als er een betere testomgeving zou worden gebruikt dan mijn MacBook.