Big data Technologies

Opdracht 2



|  |  |
| --- | --- |
| Klas | EIN3VBDa |
| Namen | Vincent Beltman & Mike Holtkamp |
| Versie | 1.0 |
| Leraar | Evert Duipmans |

Inhoudsopgave

[1 Opdracht 3](#_Toc412979393)

[1.1.1 Achtergrond 3](#_Toc412979394)

[1.1.2 OPDRACHT 1 ONTWERP 3](#_Toc412979395)

[1.1.3 OPDRACHT 2: IMPLEMENTATIE 3](#_Toc412979396)

[1.1.4 OPDRACHT 3: QUERIES SCHRIJVEN 3](#_Toc412979397)

[2 Database ontwerp 4](#_Toc412979398)

[2.1 Verantwoording keuzes 4](#_Toc412979399)

[2.1.1 Album\_tracks 4](#_Toc412979400)

[2.1.2 Postcode + Huisnummer 4](#_Toc412979401)

[2.1.3 Persoon 4](#_Toc412979402)

[2.1.4 Artiest album & track 5](#_Toc412979403)

[2.1.5 Copy 5](#_Toc412979404)

[2.1.6 Reservering / Uitlenen 5](#_Toc412979405)

[2.2 Mogelijke (Toekomstige) verbeteringen 5](#_Toc412979406)

[2.2.1 Boete systeem 5](#_Toc412979407)

[2.2.2 Variabel tabel 6](#_Toc412979408)

[2.2.3 Zoeken op kopieën 6](#_Toc412979409)

[3 Uitvoering 6](#_Toc412979410)

[3.1 Verantwoording keuzes 6](#_Toc412979411)

[3.1.1 Connectie Database 6](#_Toc412979412)

[3.1.2 Data classes 6](#_Toc412979413)

[3.1.3 Resevering / Uitlening 6](#_Toc412979414)

[3.1.4 Terug brengen 7](#_Toc412979415)

[3.1.5 Query’s zonder procedure 7](#_Toc412979416)

[3.1.6 Types 7](#_Toc412979417)

[3.2 Queries 7](#_Toc412979418)

# Opdracht

### Achtergrond

Het bedrijf Thuiseten wil een database hebben waarmee recepten en klantgegevens kunnen worden opgeslagen en opgehaald. De volgende dingen moeten apart ingevoerd kunnen worden:

* Een klant moet een beoordeling kunnen geven aan een recept;
* Een klant moet zich kunnen registreren;
* Een klant moet een recept kunnen liken;
* Een klant moet een recept kunnen toevoegen;
* Een klant moet een reactie op een recept kunnen geven, maar ook op een andere reactie;

Op de volgende manieren moet kunnen worden gezocht:

* Een klant moet een recept kunnen vinden door ingrediënten in te voeren;
* Een klant moet een recept op naam kunnen vinden;
* Een klant moet zijn eigen of die van iemand anders kunnen vinden;
* Een klant moet een lijst krijgen met recepten die hij wel leuk zou kunnen vinden;
* Een klant moet de beste recepten kunnen zien;

Verder moeten er indexen geplaatst worden. En er moet nagedacht worden over eventuele uitbreidingen aan het systeem. Het systeem hoeft geen updates, als in het wijzigen van namen en ingrediënten, uit te voeren.

# Database ontwerp

De gehele database bestaat uit twee collecties. De user-collectie en de recepten-collectie. De usercollectie ziet er als volgt uit:

|  |  |  |  |
| --- | --- | --- | --- |
| Attribuut | | Voorbeeld | Verantwoording |
| \_id | | Vincent1995 | In \_id wordt een username opgeslagen. Hiervoor hebben wij gekozen, omdat de username al uniek moet zijn per user. Hierdoor kunnen we een user makkelijk ophalen aan de hand van zijn name in plaats van een id.  Ook is dit een voordeel, omdat mongo automatisch een index maakt voor de \_id velden. Hierdoor wordt het zoeken op username snel. |
| likes | **commentID** | 54fc59831822ad7f85691953 | Deze verwijst naar de reactie die de user heeft geliked. We slaan dit op in user, zodat het mogelijk is om, bij het zichtbaar maken van de reacties, te zien welke reacties hij al geliked heeft.  Dit is beter dan het opslaan van usernames in een comment, omdat het userobject nog vrij klein is en dat van een recept niet. Zo kun je een recept sneller op halen. |
|  | **like** | true | Dit geeft aan of de like positief of negatief was. We slaan dit op in een variabele in plaats van een aparte lijst. |
| recipes | | [54ff266ea4c1adc39e8fbdda,  54f88f063674acb7bfabf910] | Dit is een lijst van de recepten die een user heeft toegevoegd. Op deze manier is het makkelijk om de recepten van een gebruiker op te halen. |

De recepten collectie ziet er als volgt uit.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribuut | | | Voorbeeld | Verantwoording |
| \_id | | | 54f88f063674acb7bfabf910 | Een recept heeft wel een naam, maar we hebben ervoor gekozen om dit niet als primary key te gebruiken. Er kunnen namelijk twee kipsalades toegevoegd worden en het zou dan lelijk zijn om er kipsalade2 van te moeten maken. |
| name | | | Kipsalade | De naam van het recept. |
| personCount | | | 5 | Het aantal personen waarvoor dit recept bedoelt is. |
| courses | | | [Toetje] | Een lijst met gangen waar dit recept voor bedoelt is. |
| difficulty | | | 5 | De moeilijkheidsgraad van het recept |
| preparationTime | | | 20 | Het aantal minuten om het recept te kunnen bereiden. We hadden hier ook seconden kunnen opslaan, maar als je dan een erwtensoep van anderhalf uur wilt opslaan, wordt het onoverzichtelijk en moeilijker om in te voeren. Recepten worden ook meestal afgebeeld in minuten. |
| types | | | [Frans,Engels] | De keuken waar dit recept vandaan komt. Dit is een lijst, omdat het mogelijk is dat een recept bijvoorbeeld Chinees-Indisch is. |
| ingredients | | **name** | Kip | De naam van het ingrediënt. |
| **amount** | 3 | De benodigdheden. |
| **unit** | stuks | De eenheid. Bijvoorbeeld gram of theelepels. |
| description | | | Een salade met kip mayonaise en als geheim ingredient: ananas. | Een beschrijving van wat het recept inhoud. |
| procedures | | | [Stap1,Stap2,Stap3] | Een lijst met stappen. We hebben ervoor gekozen om dit gescheiden te houden van de beschrijving, omdat het op deze manier gemakkelijk is om het gescheiden af te beelden. |
| reviews | **username** | | Vincent1995 | De naam van de gebruiker die de review gegeven heeft. We slaan reviews niet op in de user, omdat de user alleen gebruik maakt van de reviews als deze een recept ophaalt. Dan kunnen we dus beter de reviews in de recepten opslaan. |
| **review** | | 4 | Het aantal sterren dat deze user geeft aan dit recept. |
| comments | **\_id** | | 54fc541218225a705103e521 | Het ID van deze reactie. |
| **username** | | Vincent1995 | De eigenaar van deze reactie. |
| **body** | | Wat een geweldig recept | De body van deze reactie. |
| **path** | |  | \* |
| **depth** | | 0 | \* |
| **likes** | | 1 | Het aantal likes voor deze reactie. De likes worden hier ook opgeslagen, zodat de database bij het ophalen van de likes niet alle users langs moet gaan. De likes worden hier echter wel als een getal opgeslagen en niet als een lijst. De usernames hebben we hier toch niet nodig. |
| **dislikes** | | 0 | Zie likes |

\* Wij hebben ervoor gekozen om de reacties niet in een boomstructuur op te slaan. Het nadeel van het opslaan in een boomstructuur, is namelijk dat het zoeken dan veel moeilijker en intensiever wordt. Daarom hebben we ervoor gekozen om de reacties in één grote lijst op te slaan. Hierdoor is het zoeken wel makkelijk. Ons eerste idee was om alleen het ID van de bovenliggende reactie op te slaan in een reactie. Het probleem hiervan is, dat je dan moeilijk kan vinden wie de parent is van de parent van deze reactie. Ons tweede idee was om een lijst te maken met alle parentID’s. Maar ook dit is weer lastig zoeken. Uiteindelijk vonden we een idee uit het boek MongoDB in action. Hierbij worden alle parentID’s van een reactie opgeslagen in één string, met als separator een dubbele punt. Ook wel het pad genoemd. Aan de hand van regex of andere Mongo functies is het makkelijk om te zoeken naar een subtree en zijn parents. Tijdens het opbouwen van de boom kun je alle sub-reacties ophalen door alle paths langs te gaan en te kijken of een kind bij deze reactie hoort. En als je dan alle kinderen hebt en voor die kinderen hetzelfde doet, heb je een boom opgebouwd.

Verder geeft de depth aan hoe diep de reactie zit. De depth is gelijk aan het aantal parentID’s in het pad. Aan de hand van de depth is het makkelijker om de boomstructuur uit te printen.

## Mogelijke (Toekomstige) verbeteringen

Bij het maken van een database schema is er altijd ruimte voor discussie/verbeteringen. Om het behapbaar houden moeten er keuzes gemaakt worden welke situaties we wel en niet moeten oplossen. In deze paragraaf staan een aantal mogelijke verbeteringen/uitbreidingen voor in de toekomst.

### Reactie tijdstip van creatie

### User tijdstip van creatie

### Indexen op comments

# Uitvoering

## Het systeem

Het systeem bestaat uit aantal delen:

* Een invoer-apl. In deze class worden alle invoerquery’s uitgevoerd. Met andere woorden deze apl moet voor het zoeken worden uitgevoerd, want anders is er geen test data om op te zoeken.
* Een zoek-apl. Deze class voert alle zoekquery’s uit. De invoer-apl moet eerst worden uitgevoerd.
* Een Main-class. Deze zorgt ervoor dat de invoer-apl en de zoek-apl worden uitgevoerd.
* Een databasehelper. Hierin wordt de database connectie aangemaakt en worden query’s uitgevoerd. De opbouw van de query’s wordt zoveel mogelijk gedaan in de invoer-apl en de zoek-apl.

## Query’s

### Invoer query’s

### Zoek query’s

## Indexen

In de tabel hieronder staan alle indexen die gebruikt worden om het zoeken sneller te maken. Ook de verantwoording wordt gegeven.

|  |  |
| --- | --- |
| recipe.\_id | Een recept wordt het vaakst aan de hand van \_id opgehaald. Ook staat er automatisch een index op \_id, dus een verantwoording is hier eigenlijk niet nodig. |
| recipe.ingredient.name | Omdat het de bedoeling is dat het zoeken op ingrediënten veel gebruikt moet worden, staat er een index op de naam van het ingrediënt. De index staat in alfabetische volgorde, zodat deze makkelijk gesorteerd opgehaald kunnen worden. |
| recipe.name | Een recept zal ook via de naam worden opgehaald. Denk bijvoorbeeld aan user die wil zoeken naar een schotel. Een schotel is over het algemeen geen ingrediënt. Daarom wordt er ook een index gezet op de naam van het recept. Ook deze staat in alfabetische volgorde. |
| user.\_id | Hier staat automatisch een index op. Wij hebben er echter voor gekozen om hierin de username op te slaan in plaats van een Object id. Als dit wel het geval was, zouden de users opgehaald worden aan de hand van \_id en de username. Dit kan net zo goed samengevoegd worden aangezien het id maar één waarde toevoegt aan de user. En dat is, dat een Object id een tijdstip bevat. Wij hebben hier niet voor gekozen, omdat dit ook in een apart veld kan als er behoefte aan is. |