Document store naar rationele database

Wytze Ketel 1797080
Kenny van den Berg 1777503
Izabelle Auriaux 1761808
Ceyhun Cakir 1784480

Maart 2021

1 Probleem

Om een soepel lopende recommendation engine te maken is het best om de gegevens te downloaden en direct te lezen van een speciale analyse database. Hiervoor maken wij een connectie tussen een MongoDB en MySQL. De gewenste data wordt uit MongoDB gehaald en via verschillende algoritmes in een Python programma verwerkt op de gewenste manier. Zodra de data uit MongoDB op de gewenste manier is verwerkt wordt deze in die structuur geplaatst in onze analyse database. Door het op deze manier te doen is de data makkelijker te analyseren en te gebruiken voor onze recommendation engine.
Als we de data niet eerst zouden behandelen zou het heel lastig worden om de juiste data op te vragen voor het recommendation engine. Ook is zonder de juiste specificatie en labeling van de data het moeilijk om goede algoritmes op te zetten om recommendations te maken. Evenals wanneer de data niet op een speciale analyse database wordt geladen. Door de data om de zoveel tijd opnieuw te behandelen en te uploaden is de data altijd up to date en kan de recommendation engine ook het best de data verwerken en de beste recommendations geven.

1.1 Onderdelen

Om het probleem op te lossen we stapsgewijs de volgende onderdelen bouwen en functionerend krijgen, wanneer deze onderdelen dan verenigt worden in een programma is het probleem opgelost.

1. Rationele database opzetten (Extern)
2. Python connectie met de databases
3. Inlezen van data uit een MongoDB in programma
4. Omzetten ingelezen data naar gewenste vorm.
5. Database invullen met gewenste vorm.
6. Omgezette data naar database schrijven.

1.2 Problemen

We hebben ervoor gekozen om al direct wat extra functionaliteit en complexiteit toe te passen in onze rationele database, ter voorbereiding van de nog te komen algoritmes.
Alhoewel het simpel inladen van de data, en de omzetting hiervan van MongoDB naar MySQLdb al gauw gerealizeerd was, was het omzetten van de data nog wel wat truckenwerk. We liepen vooral tegen de volgende problemen aan.

- Zinnigheid data controleren

- Verbanden opzetten in de data
- Complexiteit van het verwerkings algoritmes

1.3 Oplossingen

Voor de bovengenoemde problemen hebben we al enige oplossingen verzonnen, die ook deel uitmaken van de pseudocode.
Zinnigheid controle: Kolommen waar helemaal geen data instaan of tertiaire subonderdelen die allemaal leeg zijn of gewoon data dat zo weinig voorkomt dat er weinig mee te doen valt worden allemaal uitgefilterd op selectie we halen ze wel op maar stoppen ze niet meer terug in de rationele database.
Verbanden opzetten: zie onderdeel 3, database ontwerp voor een uitgebreide uitleg. We gebruiken keys en ondergeschiktheid om in onze rationele database een web van connecties op te zetten.
Complexiteit van het verwerkingsalgoritme: Het werdt al gauw duidelijk dat het grootte probleem niet eenvoudig was op te delen in een parallelle werksfeer, de kleine en makelijkere onderdelen waren al gauw af echter het algoritme zelf moest toen nog gedaan worden, om de werkdruk toch te verdelen hebben we gebruik gemaakt van de atom teletype plugin hiermee konden we met z'n allen in 1 document samenwerken.

2 Conversie algoritme

Ons Python programma begint door een connectie te maken met zowel MongoDB als MySQL. Het programma haalt alle gewenste data op uit de MongoDB database en deze wordt vervolgens in ons Python programma verwerkt. We organiseren de data op verschillende kenmerken en slaan deze op in een library met andere data met dezelfde kenmerken. Zodra alle data op de juiste manier is gesorteerd schrijft het programma de verwerkte data over op de MySQL database. Verder in dit hoofdstuk gaan we de pseudo code laten zien en daarbij aandacht geven aan hoe we sommige overwegingen hebben gedaan en hoe onze keuzes tot de definitieve code hebben geleid.

2.1 Pseudo code

Pseudocode voor het omzetten van data vanuit de MongoDB naar onze gewenste vorm voor de rationele database. De eerste def: deze controleert of we een database hebben, en zoja dan haalt hij alle data op en stuurt deze dan voor verwerking naar de volgende def. Zo nee dan krijg je een error.

Open een connectie met mongoDB
Zolang het waar is
Haal de database informatie op uit MongoDB
Controleer of de opgehaalde database inderdaad bestaat
Zo ja dan:
Verwerk de data in de database
Zo nee dan:
geef de gebruiker een error message.
sluit connectie(s)

:De tweede def handelt het verwerken van de data, deze zet de opgehaalde data van de eerste def om naar daadwerkelijk
weg te schrijven data.
Verwerk de data:
Maak een opslag list aan voor elk element dat je wilt wegschrijven, en herhaal dit voor elke categorie.
Bijv: Products value item = []

Maak een verbinding met MongoDB
haal products op
haal sessions op
haal profile op

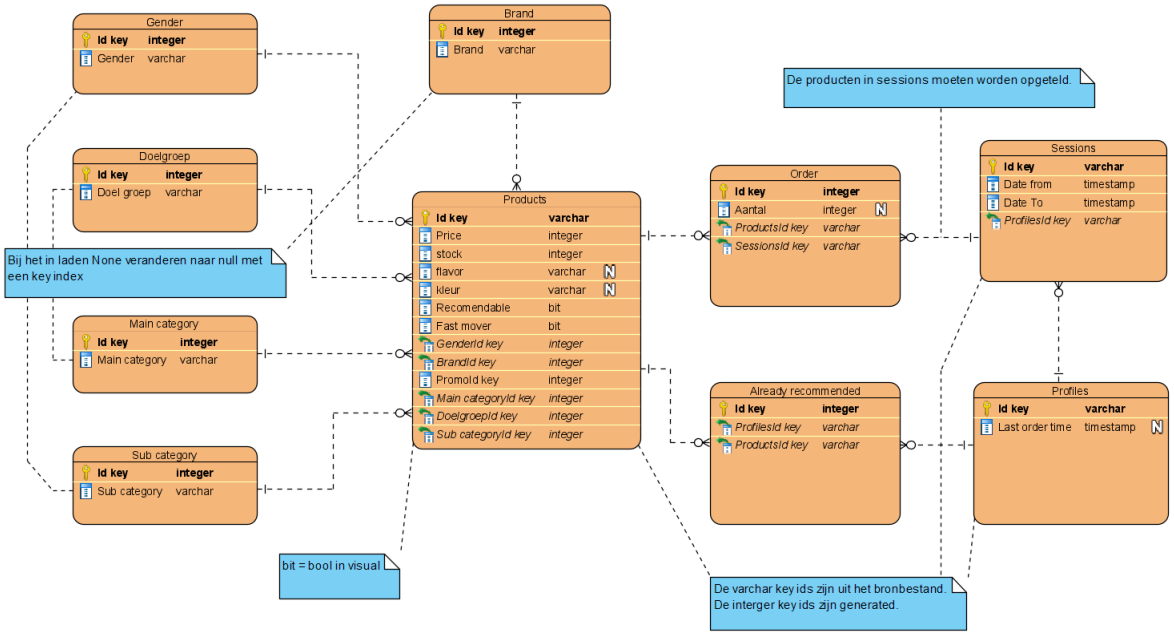
:Voor elk van deze drie doe het volgende.
voor waarde in categorie (products of sessions of profile) vindt,key
controleer of elke van deze opties wel bestaat, en data bevat.
Zo ja:
Schrijf de data in de desbetreffende lijst

Zo nee:
Schrijf "geenïn de desbetreffende lijst
Wanneer alle gewenste lijsten voor de categorie zijn aangemaakt, voeg ze samen met .append, in een lijst van lijsten.

Insert de gegenereerde lijsten in de Mysql database,categorie voor categorie.

3 Database ontwerp

Om een zinnige rationele database te ontwikkelen is het belangrijk om de gegeven data te bestuderen.
Bij het bestuderen van de data kwamen we al op onze eerste conclusies; er waren delen van de data die geen zinnige informatie bevatten, bijvoorbeeld, een groot aantal records bevatte helemaal geen informatie slechts een ID, en in de data die na die filtering overbleef waren er vaak veel kolommen die leeg waren, of waarin slechts een zeer beperkt aantal van de records data bevatte. Een voorbeeld was de kolom voor promotie, waar slechts één van de 33.000 records een true had. Aangezien we op dit soort punten geen verkoop voorstel gaan maken hebben we besloten een groot deel van die minder zinnige data niet meer op te nemen in de nieuwe rationele database.
Vervolgens hebben we gekeken naar de andere kant, wat zouden we wel willen hebben in onze nieuwe database en van wat we willen hebben waar willen we snel bijkomen, en wat willen we ondergeschikt maken aan wat het resultaat van dit onderzoek heeft na enige iteraties het model van figuur 1 hieronder voortgebracht.



Figuur 1: MySQL rationeel database

In het model van figuur 1 hebben we de derde normaalvorm toegepast, we willen immers elke stuk data maar eenmaal wordt opgeslagen. We vullen dus de gewenste data in de juiste tabel in en verbinden dezen vervolgens met keys

weer terug naar de products tabel. Op deze manier hebben we een balans tussen opslag en de bereikbaarheid van de belangrijkste stukken data.
We hebben de rond liggende categorieën voornamelijk gekozen op basis van toekomstige verkoop voorstellen, (zie gender en doelgroep bijvoorbeeld), door de keys vanuit die tabellen te gebruiken kunnen we dan later snel en effectief enige voorstellen genereren vanuit de products tabel.