Onderzoek: databases

Team Data Dumpsters:

Niek Smets

Lorenzo Elias

Nicolas Van Dyck

Matthias Van Rooy

Ward Boeckx

Inhoud

1. Inleiding

We willen in kaart brengen hoe we de data van beide projecten zullen verwerken. De dataregistratie enerzijds en het bekerproject anderzijds. Aangezien beide projecten toch iets andere noden hebben qua database, maken we hier een duidelijke splitsing.

Omdat we voor het dataregistratieproject meerdere tabellen gebruiken die onderlinge met elkaar in verbinding staan, hebben we hier nood aan een relationele database. Voor de bekerrecyclage ligt dit anders. Dat project heeft vooral baat bij het gebruik van een time-series database. Dat zijn databases die geoptimaliseerd zijn voor het werken met 'time-stamped' data. Omdat de bekerrecyclage een specifiekere nood heeft, onderzoeken we eerst dit project.

1.1 Bekerrecyclage

Voor de bekerrecyclage krijgen we telkens dezelfde soort data doorgestuurd via MQTT. Dit dataobject zal volgende parameters bevatten: label of id van de vuilbak, de batterijstatus, de vullingsgraad, de longitude en de latitude. Telkens we dit ontvangen, zullen we die data wegschrijven in een database en er een timestamp aan toevoegen, zodat we telkens kunnen filteren op de laatst binnengekomen data.

Dit heeft als gevolg dat we eigenlijk maar één tabel nodig hebben, die we vlot moeten kunnen filteren en sorteren op de verschillende parameters. De database moet ook vlot vele requests tegelijkertijd kunnen verwerken en goed geïntegreerd kunnen worden in onze frameworks.

Na het bekijken van enkele opties kwamen we uit op de twee kanshebbers. MongoDB en TimescaleDB. Beide databases worden vaak gebruikt voor IoT-projecten. Aanvankelijk kwam ook InfluxDB als kanshebber naar boven, maar die bleek over minder krachtige filterfuncties te beschikken dan de andere twee.

1.2 Beschrijving beide databases:

1.2.1 TimescaleDB

TimescaleDB is een database gespecialiseerd in time-series die gebouwd werd op PostgreSQL. Belangrijk zijn de optimalisatie voor snelle invoer en efficiënte queries, zeker met het oog op de uitbreiding van het bekerrecyclageproject. Bovendien biedt het mogelijkheden om oude data automatisch te verwijderen om opslagruimte te besparen. De aanwezigheid van MySQL maakt het ook nog eens zeer toegankelijk voor andere developers die het project na verloop van tijd zullen overnemen.

1.2.2 MongoDB

MongoDB is een document-store database die ook gebruikt kan worden voor time-series data, maar gebruik maakt van NoSQL. Er is echter ook ondersteuning voor time-series aanwezig. MongoDB is horizontaal schaalbaar en kan grote hoeveelheden time-series data verwerken zonder schema-aanpassingen (aangezien het niet met een schema werkt). Voor ons project zijn schema-aanpassingen niet nodig, want elke vuilbak zal telkens dezelfde parameters doorsturen.

1.3 Vergelijking

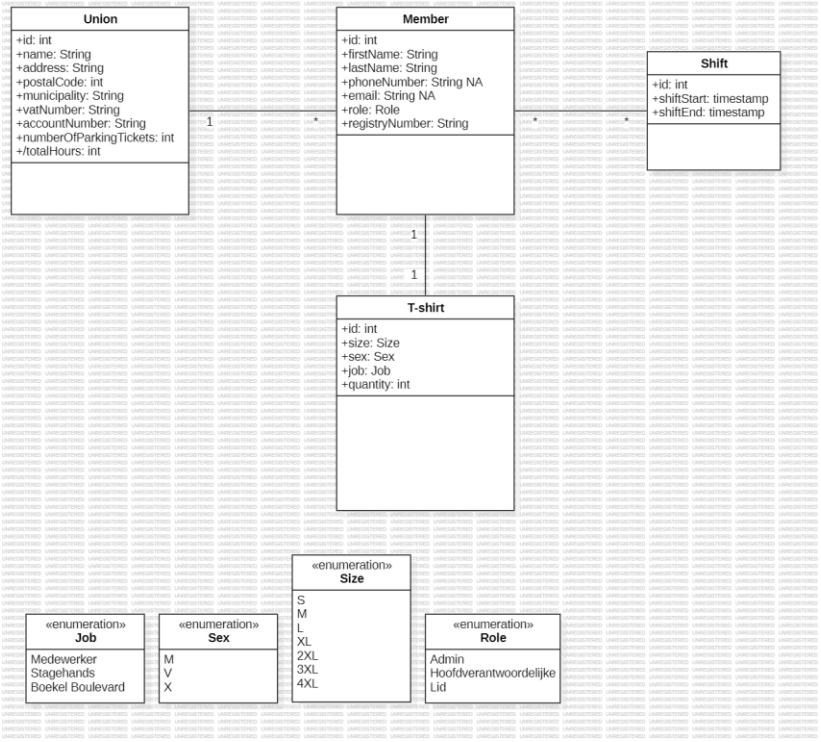
Wat meteen opvalt tijdens het onderzoek is hoe we op de website van TimescaleDB resultaten kunnen vinden van een vergelijking in performantie tussen TimescaleDB en MongoDB, waaruit duidelijk blijkt dat TimescaleDB een vele hogere performantie haalt op vele vlakken, zowel qua wegschrijven van data als qua queries. Twee criteria die toch zeer belangrijk zijn voor het bekerrecyclageproject. Uiteraard moeten we voorzichtig zijn, aangezien we ervan uit kunnen gaan dat het bedrijf achter TimescaleDB zichzelf uiteraard niet in de voet wil schieten. Toch vinden we op de website van MongoDB helemaal niets dat deze claims enigszins weerlegt. Over een vergelijking wordt zelfs niet gesproken.

Zelfs wanneer we dit niet meenemen in onze overweging lijkt TimescaleDB de betere keuze voor dit project, omdat de focus ligt op de analyse na verloop van tijd en omdat er telkens dezelfde parameters worden doorgestuurd, wat het gebruikt van een document-store database niet per se overbodig maakt, maar toch minder noodzakelijk maakt. Dit maakt MongoDB zeker geen slecht alternatief, maar alles bij elkaar genomen, past TimescaleDB beter bij onze use-case.

2.1 Dataregistratie

Wat betreft de dataregistratie maken we eerst een schematische voorstelling van alle tabellen en hun properties. Omdat we met verschillende tabellen werken, hebben we sowieso nood aan een relationele database.

2.1.1 Klassediagram



2.1.2 Opties

We kunnen kiezen voor databases gebaseerd op SQL of op NoSQL. Voor ons project lijkt SQL hier de betere keuze, aangezien dit ons toegankelijker lijkt voor een toekomstige developer of iemand die de app onderhoudt en omdat we belang hebben bij duidelijke gestructureerde relaties tussen de tabellen. Bovendien wordt in het onderzoek naar een database voor de bekerrecyclage duidelijk dat we ook hier zullen werken met een technologie die gebaseerd is op SQL. Met andere woorden proberen we ook hier beide projecten zo dicht mogelijk bij elkaar te laten aanleunen, zodat dit een minder grote leercurve vormt voor de persoon die dit later van ons zal overnemen.

De opties waartoe we ons dan limiteren zijn SQLite, MySQL of PostgreSQL. Er is weinig verschil tussen deze soort databases wat betreft functionaliteit, maar er zijn wel argumenten om het ene boven het andere te verkiezen.

Alle drie de soorten worden bijvoorbeeld ondersteund door Java

Zo blijkt SQLite niet geschikt voor apps waarbij er vele ‘write’ operaties tegelijkertijd moeten kunnen gebeuren en laat dat nu net nodig zijn voor ons project. Want als er vele medewerkers tegelijkertijd het terrein op willen, zullen ze niet op elkaar kunnen wachten. Daarenboven ondersteunt SQLite ook veel minder datatypes.

MySQL lijkt al beter geschikt te zijn dan SQLite, omdat het zo wijdverspreid in gebruik is, snel is en ingebouwde features heeft qua security. Toch is het ook hier belangrijk om op te merken dat het problematisch kan zijn wanneer er vele ‘read-writes’ tegelijkertijd voorkomen. In principe zal onze dataset eerder beperkt blijven, maar als we het risico op vertragingen zo klein mogelijk willen houden, kan dit toch belangrijk zijn.

PostgreSQL heeft dan weer als nadeel dat het minder snel zou zijn dan MySQL wat betreft ‘read’-operaties.

2.1.3 Conclusie

Op het eerste zicht lijkt de keuze tussen MySQL en PostgreSQL dus weinig verschil met zich mee te brengen. Beide databases hebben hun eigen sterktes, maar geen enkele van de twee lijkt daar duidelijk veel beter geschikt te zijn voor de kleine database die wij zullen hebben.

Wat betreft de bekerrecyclage is er echter wel een duidelijke keuze en die maakt gebruik van PostgreSQL. Om zoveel mogelijk de uniformiteit te bewaren, kiezen we dus ook hier voor PostgreSQL.

3 Referenties

drake, o. a. (2022, 03 10). *SQLite vs MySQL vs PostgreSQL*. Opgehaald van DigitalOcean: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

Inc., M. (2024, 11 11). *MongoDB: The Developer Data Platform | MongoDB*. Opgehaald van MongoDB: https://www.mongodb.com/

Inc., T. (2024, 11 11). *How to store time-series Data in MongoDB and why that's a bad idea*. Opgehaald van Timescale: https://www.timescale.com/blog/how-to-store-time-series-data-mongodb-vs-timescaledb-postgresql-a73939734016/

Inc., T. (2024, 11 11). *TimescaleDB*. Opgehaald van Timescale White-paper benchmarking: https://assets.timescale.com/whitepapers/Timescale_WhitePaper_Benchmarking_Mongo.pdf

Inc., T. (2024, 11 11). *TimescaleDB vs. InfluxDB*. Opgehaald van Timescale: https://www.timescale.com/blog/timescaledb-vs-influxdb-for-time-series-data-timescale-influx-sql-nosql-36489299877/

Pfeiffer, S. (2021, 09 21). *Should I use SQLite, PostgreSQL or MySQL?* Opgehaald van DEV: https://dev.to/codesphere/should-i-use-sqlite-postgresql-or-mysql-1o4b

Reserved, I. I. (2024, 11 11). *Time series database (TSDB) explained | InfluxData*. Opgehaald van InfluxData: https://www.influxdata.com/time-series-database/