I4DAB

Forår 2019

Journal #3: Social Network

Deltagere:

Studienummer	**Navn**
201704441	Frank Andersen
201704714	Michael Møller-Hansen
201505470	Mads Skytte Nielsen

Indholdsfortegnelse

1 Introduktion

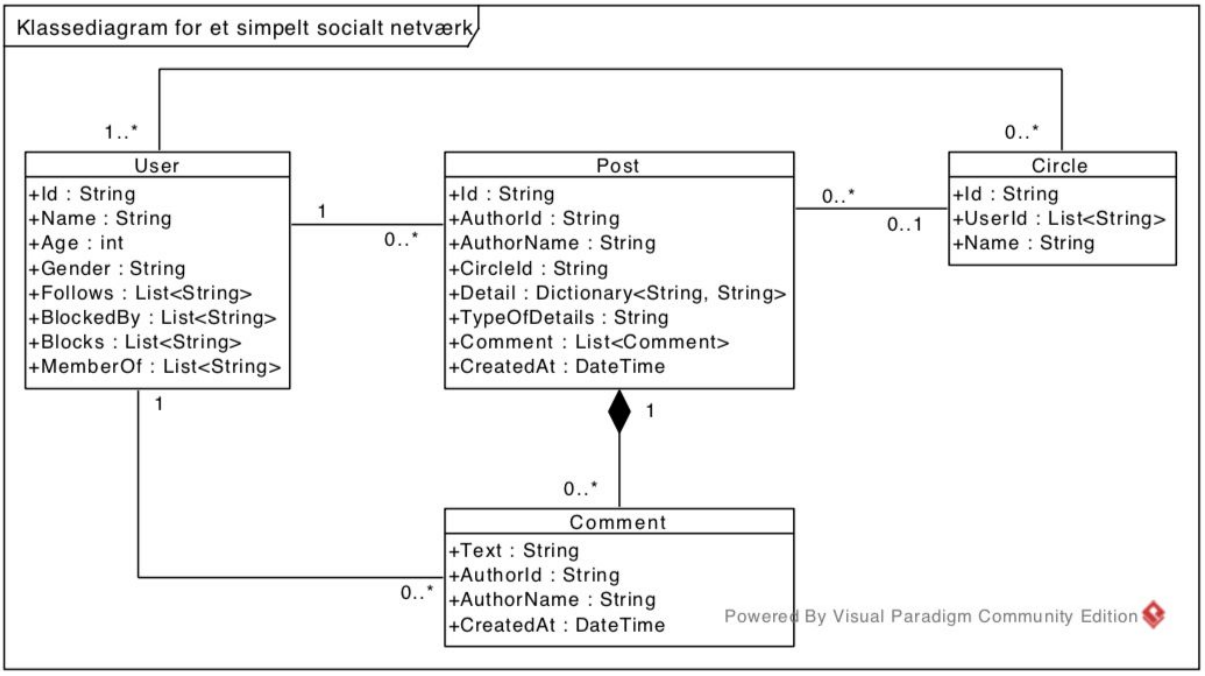
I denne opgave beskrives designet og implementeringen af et simpelt social netværk baseret på en NoSQL-database, som i denne kontekst er MongoDB.

Implementering af det sociale netværk er lavet som et REST API, hvorigennem der kan oprettes, opdateres og hentes data.

Vedlagt løsningen er en README.md fil, som beskriver hvordan API'et benyttes. Derudover er der også vedlagt en importfil til programmet Postman, som gør det nemmere at benytte API'et. Dette står beskrevet i detaljer i README.md-filen.

2 Schema

På *figur 1* ses et klassediagram for det simple sociale netværk.



Figur 1 - Klassediagram af NoSQL databasen for det simple sociale netværk.

Klassediagrammet på *figur 1* er designet mod at optimere læsehastigheden at data, således at en væg eller et user feed kan hentes ved brug af færrest mulige collections. Dette ses bl.a. ved, at "har-mange"-relationen mellem Post og Comment, hvor Comment er indlejret i Post, og at "mange-til-mange" relationer mellem f.eks. Circle og User er gemt hos begge entities. En Circle har altså en liste af UserIds, mens User også har en liste af dennes Circles (kaldet MemberOf).

Entitien Post har en Dictionary<string, string> under navnet Detail, som giver mulighed for at tilføje vilkårligt indhold til en Post i formatet "key-value"-par. Derudover har Post også en

property kaldet TypeOfDetails, som er en descriptor, der beskriver typen af det indhold, som findes under Detail.

2.1 Problemer

2.1.1 Comments indlejret i Post

Når en Post har fået mange Comments, så vokser størrelsen på Post, da Comments er indlejret i denne. Dette gør, at det bliver tungt at læse Post, da alle indlejrede Comments også kommer med.

En løsning ville i dette henseende ville være at udskille Comments til sin egen collection, hvorunder ét dokument indeholdt alle kommentarer til én Post. Dermed kunne Post blot indeholde en reference til det dokument, som indeholdt alle Comments.

2.1.2 Slette Circles (N+1)

Som tidligere beskrevet, så ligger "mange-til-mange" relationer på begge entities. Dette medfører bl.a., at når en Circle skal slettes, så skal N brugere opdateres, da disse ellers ville indeholde en reference til en nu slettet Circle.

3 Queries

Herunder er der beskrevet de to almindelige queries som optræder i systemet. Dette er når en bruger skal se sit eget feed eller en anden brugers væg.

3.1 Se eget feed

Først hentes "egen bruger" ud at databasen. Denne indeholder hvem han følger, hvem han har blokeret, hvem han er blokeret af og hvilke circles han er medlem af. Disse kriterier benyttes til at hente de 10 nyeste Posts, som denne bruger er berettiget til at se.

3.2 Se anden brugers væg

I forbindelse med at en bruger læser en anden brugers væg, så hentes først begge brugere ud af databasen. I denne kontekst kaldes den bruger, hvis væg der kigges på, for *owner*, mens den bruger, som kigger på væggen, kaldes *viewer*.

Såfremt *viewer* ikke er blokeret af *owner*, så hentes de 10 nyeste Posts, som *owner* har skrevet, hvor Post er enten public eller i en Circle, som *viewer* også er en del af.

4 Sharding

I dette afsnit beskrives overvejelser ift. sharding af collections i databasen, såfremt det simple sociale netværk blev populært.

4.1 User

Det antages, at brugere fra et land oftere er følgere af andre bruger fra samme land, hvorfor det vil give mening at angive brugerens land som del af en compound shardkey. Herudover kunne der f.eks. anvendes brugerens navn til at distribuere brugere jævnt over flere shards.

4.2 Post

En post ville typisk blive tilgået alt afhængigt af hvilken circle den er del af, og dermed ville det være oplagt at anvende CircleId'et som en del af compound shard key. Den anden del af shardkeyen, kunne f.eks. være udfra lokation eller authorId.
Posts vises hovedsagligt i et feed eller på en væg, og det må forventes at en dansk bruger hovedsageligt ser danske opslag, hvor det giver mening af gruppere disse i én shard.

4.3 Circle

Det vil være naturligt at distribuere circles på tilsvarende måde som de users, den indeholder. Dette vil sige henholdsvis efter lokation og navn.