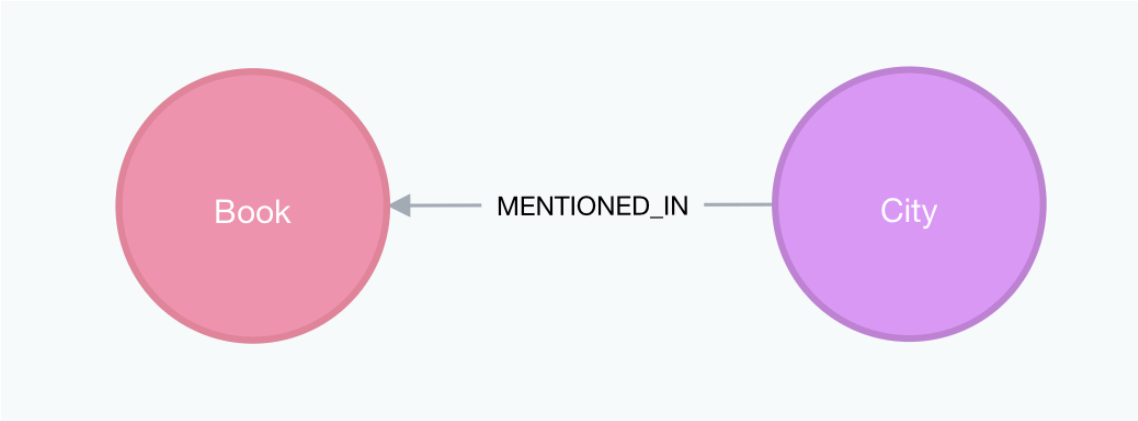
Gutenberg rapport

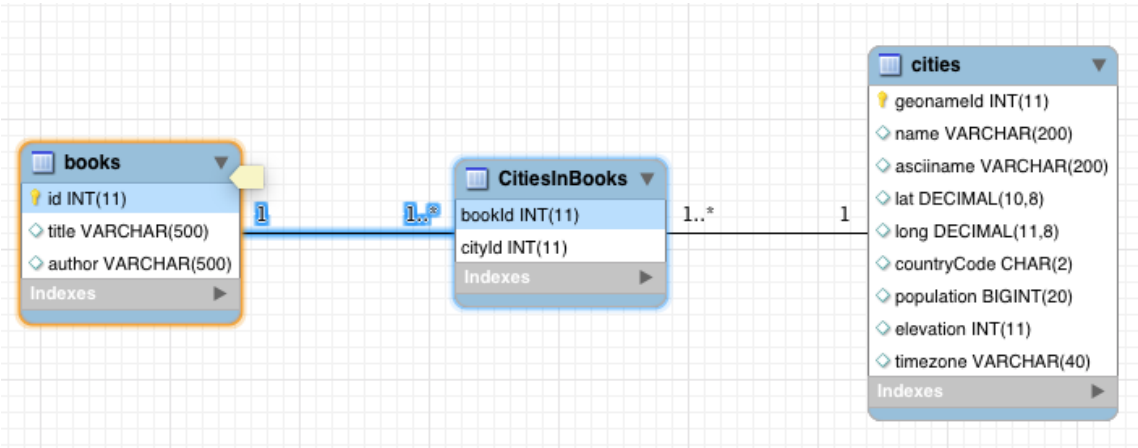
Cheesy Choppin - Andreas Larsen & Emil Pilgaard

Vi har valgt at benytte MySQL og Neo4j som vores to databaser til dette projekt.

I vores Neo4j database har vi to node labels: Books og City. Imellem sig har de relationen MENTIONED_IN, som fortæller hvilke byer der bliver nævnt i en given bog.



I vores MySQL database har vi 2 tabeller: books, cities, og en junktiontabel kaldet CitiesInBooks, tabellen indeholder alle relationerne mellem bøgerne og byerneinformationerne om hvilke bøger der indeholder hvilke byer.



Vi startede med at importerer alle byerne til MySQL fra en klient af ved at køre følgende query:

```
LOAD DATA LOCAL INFILE '/Users/"your_username"/examproject/gutenberg/cities5000.csv'
INTO TABLE cities
fields terminated by ',' enclosed by '""'
lines terminated by '\n'
ignore 1 lines;
```

Efterfølgende har vi lavet en metode i Java som først henter alle byerne ned i en liste som vi kunne bruge senere. Derefter læste vi en fil ad gangen, for at finde titellen, forfatteren og alle ord der har et stort begyndelsesbogstav, og gemme dem i en liste. Derefter kaldte vi en metode som sammenligner de fundende ord med vores liste af byer, for at sortere alle de ord fra som ikke er byer. Så indsatte vi book-objekterne som består af forfatteren og titlen til vores MySQL database, og fik så et id retur som databasen har genereret. Dette id kunne vi nu bruge til at indsætte alle relationerne mellem bogen og de fundende byer i denne bog.

For at vi ikke skulle skanne alle bøgerne 2 gange har vi eksporteret de tre tabeller fra MySQL til csv filer, og derefter kunne vi køre en metode i Java som læser en cql fil og køre vores cypher queries som vil loade de tre csv filer ind i Neo4j.

Vi har lavet nogle hastighedstest af de forskellige databasekald vi bruger. Opsætningen i forhold til de to databaser er den samme, vi har lavet 5 forskellige book og city objekter, som bliver brugt til at kalde metoderne i begge databaser.
Tiderne for de forskellige databasekald kan ses nedenfor. Tiderne er den totale tid for de fem databasekald, og ikke per kald, hvilket er årsagen til de høje tider.

Metode	MySQL	Neo4j
GetBooksByAuthorName	877ms	1345ms
GetBooksByGeoLocation	1032ms	2479ms
GetCities	1713ms	2773ms
GetCitiesByBookTitle	126ms	485ms
GetBooksByCity	284ms	936ms

Det vi har kunnet se ud fra de hastighedstest vi har lavet er at MySQL er hurtigere over hele linjen, hvilket også hænger meget godt sammen med teorien om at MySQL er meget hurtig så længe den ikke skal joine mange tabeller, i modsætning Neo4j som virkelig imponerer, når der skal forbindes mange relationer, hvilket ikke er tilfældet i dette system.

Vores anbefaling til andre der ønsker sig at give sig i kast med at bygge et lignende system vil derfor være at anvende MySQL over Neo4j.