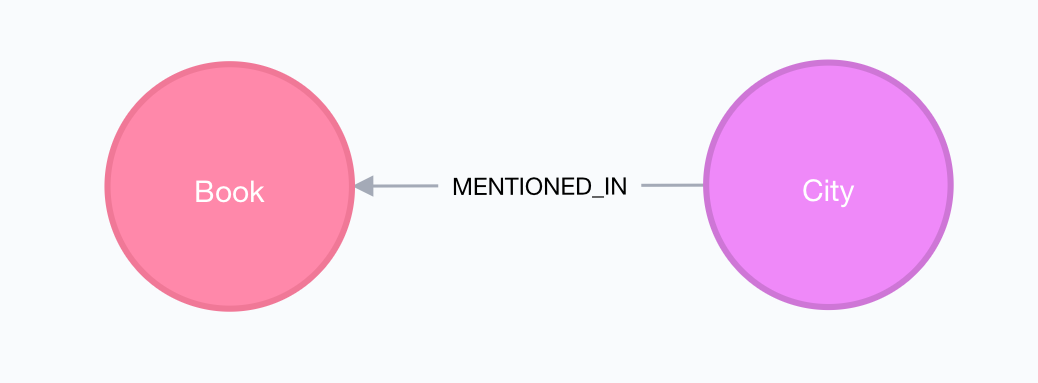
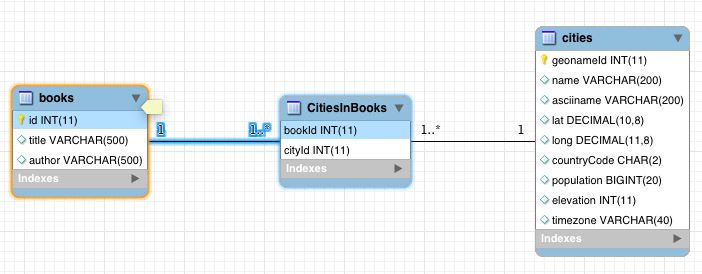
# Gutenberg raport

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

I vores Neo4j database har har vi to node labels: Books og City og imellem sig har de relationen MENTIONED\_IN

I vores MySQL database har vi 3 tabeller, en books, cities og CitiesInBooks som er en junktion tabel der indeholder informationerne om hvilke bøger der indeholder hvilke byer. 

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

*LOAD DATA LOCAL INFILE '/Users/Emil/examproject/gutenberg/cities5000.csv' INTO TABLE cities*

*fields terminated by ',' enclosed by '"'*

*lines terminated by '\n'*

*ignore 1 lines;*

herefter har vi lavet en metode i Java som først henter alle byerne ned i en liste som vi vil bruge senere, derefter læser vi en fil ad gangen, og finder titel, forfatter og alle ord der har et stort startbogstav, og gemmer dem i en liste. Derefter kalder vi en metode som sammenligner vi de fundende ord med vores liste af byer. Så indsætter vi bog objekterne som består af forfatteren og titlen til vores MySQL database og får et id retur som databasen har genereret, dette id kan vi nu bruge til at indsætte alle relationerne mellem bogen og de fundende byer.

For at vi ikke skal skanne alle bøgerne igen har vi eksporteret de tre tabeller fra MySQL til csv filer, og derefter kan vi fra java køre et script som vil læse csv filerne ind i Neo4j.

Vi har lavet nogle hastighedstest af de forskellige databasekald vi laver. Opsætningen i forhold til de to databaser er den samme, vi har lavet 5 forskellige bog og by 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 imponere, når der skal forbindes mange relationer.

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.