Big Data 2019  
Portfolio assignment - part two

Mette Strand Hornnes

Innholdsfortegnelse

[1 Choosing the right storage technology 3](#_Toc23863996)

[1.1 Column Store Database - Apache Cassandra 3](#_Toc23863997)

[1.1.1 Om Column Store Database & Cassandra: 3](#_Toc23863998)

[1.1.2 Datasett 5](#_Toc23863999)

[1.1.3 Spørringer på dette datasettet 5](#_Toc23864000)

[1.1.4 Beskrivelse av datasettet og hvorfor dette passer 6](#_Toc23864001)

[1.1.5 Ved annen bruk av dataene 6](#_Toc23864002)

[1.2 Graph Database - Neo4j 6](#_Toc23864003)

[1.2.1 Om Graph Database & Neo4j 6](#_Toc23864004)

[1.2.2 Datasett 8](#_Toc23864005)

[1.2.3 Spørringer på dette datasettet 8](#_Toc23864006)

[1.2.4 Beskrivelse av datasettet og hvorfor dette passer 8](#_Toc23864007)

[1.2.5 Ved annen bruk av dataene 8](#_Toc23864008)

[1.3 Key-value database - Riak 9](#_Toc23864009)

[1.3.1 Om key-value database & Riak 9](#_Toc23864010)

[1.3.2 Datasett 9](#_Toc23864011)

[1.3.3 Spørringer på dette datasettet 9](#_Toc23864012)

[1.3.4 Beskrivelse av datasettet og hvorfor dette passer 9](#_Toc23864013)

[1.3.5 Ved annen bruk av dataene 9](#_Toc23864014)

[1.4 Document Store Database - Mongo DB 9](#_Toc23864015)

[1.4.1 Om Document Store Database & Mongo DB 10](#_Toc23864016)

[1.4.2 Datasett 10](#_Toc23864017)

[1.4.3 Spørringer på dette datasettet 10](#_Toc23864018)

[1.4.4 Beskrivelse av datasettet og hvorfor dette passer 10](#_Toc23864019)

[1.4.5 Ved annen bruk av dataene 10](#_Toc23864020)

[2 Practical use of one of the storage technologies 10](#_Toc23864021)

[3 Kildeliste: 13](#_Toc23864022)

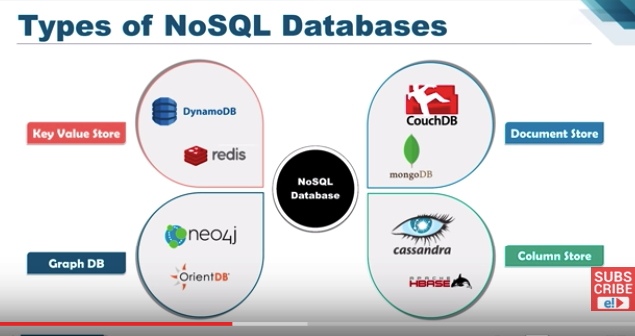
Introduction

In this second, and last, part of your portfolio, you will demonstrate that you

have understood the concept of polygot persistence and are capable of choosing

the right storage technology for datasets you select yourself.

# 1 Choosing the right storage technology



A central tenet in the NoSQL movement is that there is no one-size-fits-all data model and storage technology. In the second part of this course, you have learned about four major data models and database engines for those data models.

In this assignment you will, for each of those four data models, find a dataset in the Kaggle Dataset directory for which that data model is the most suitable.

Keep in mind that choosing the right storage technology is not only about understanding the data itself, but also about how that data is created and used.

## 1.1 Column Store Database - Apache Cassandra

### 1.1.1 Om Column Store Database & Cassandra:

I motsetning til vanlig database hvor data ligger radvis, vil data i column store-database ligge i kolonner som er lagret hver for seg. Dette gjør blant annet operasjoner som krever aggregering veldig raskt, da man kun trenger å lese utvalgte kolonnene istedenfor for hele raden.

**Komponenter:**Denne type database benytter et konsept kalt keyspace som er et skjema for hvor dataene befinner seg. Dette keyspacet inneholder alle Column families som inneholder rader med de ulike kolonnene.   
Hver rad har en unik nøkkel som identifiserer denne raden. Hver kolonne for hver rad inneholder navn, verdi og timestamp for dato og tid for når data ble lagret.

**Fordeler med Column Store DB:**

* Kan komprimeres - Da hver kolonne kan ha hver sin datatype
* Raskt med aggregeringsoperasjoner – Da det er raskt å hente data fra ulike kolonner
* Enkelt å skalere – Velig lett å kun legge til en kolonne om dette trengs

**Egner seg til:**

* Applikasjoner som er geografisk lagret utover flere datasentre
* Applikasjoner som kan tåle noe inkonsistens i replikaene for en kort tid
* Applikasjoner som krever lagring av store volum med data (hundrevis av TB)

**Vanlige bruk:**

* Sikkerhetsanalyser som benytter nettverkstrafikk
* Big Science
* Markedsanalyse
* Nettskalering – Som søk

**Apache Cassandra - Eksempel på en Colum Store DB**

Cassandra er et distribuert databasesystem som er designet for å håndtere store volum med strukturert data. Dataene som brukes kan være fordelt utover ulike maskiner. Dette gir høy skalerbarhet, gjør det mulig å håndtere store mengder data raskt (mye raskere enn for eksempel MySQL), samt gir høy tilgjengelighet. Cassandra er dermed laget med tanke på at feil på en maskin (software eller hardware) kan oppstå og hindrer derfor «Single point of failures».

<https://www.guru99.com/cassandra-tutorial.html>

Arkiktetur

Arkitekturen til Cassandra består kun av noder (maskiner/servere) eller clustere (samling av noder) og ingen master. Alle nodene spiller samme rolle og er selvstendige samtidig som de er koblet til de andre nodene med et peer-to-peer-system. Hver node kan skrives til eller leses fra uavhengig av hvor dataene ligger, noe som også hindrer Single point of failues.

Med en såkalt gossip-protokoll deles informasjonen om data gjennom clusteret hvert sekund.   
Det lages også kopier (replicas) av dataene som legges på andre noder, noe som igjen er med på å hindre feil og tap av data.

<https://intellipaat.com/blog/tutorial/cassandra-tutorial/brief-architecture-of-cassandra/>

Skriving av data skjer på følgende måte:

1. Klienten skriver til en av nodene
2. Fra denne skrives replicas til andre noder (antall bestemmes av replication factor som settes ved oppsett)
3. Nodene bekrefter lagring tilbake til den første noden, som igjen bekrefter tilbake til klienten om at lagring er fullført (Acknowledge-melding).
4. Når klienten mottar denne ACK-meldingen skrives info om datalagringen til en commit-logg.

### 1.1.2 Datasett

**For Column store database har jeg valgt følgende datasett:**Supermarket sales - Historical record of sales data in 3 different supermarkets  
 **Link til datasett:**<https://www.kaggle.com/divyeshardeshana/warehouse-and-retail-sales>

**Hvorfor denne datamodellen?:**

Data med oversikt over salg gjort. Her kan man lett ved å benytte aggregeringen kunne finne nyttig informasjon som gjennomsnitt solgt osv. . . .

Dette datasettet inneholder informasjon over salg gjort i ulike supermarkeder. Da mye av denne informasjonen vil være nyttig for å for eksempel finne ut hvordan pris påvirker salget, hvor og hva som selges vil column store db være den riktige måten å lagre denne dataen på. I motsetning til ved en tradisjonell Row-stored database hvor man måtte ha lest mye data som ikke er av interesse, vil man enkelt kunne hente ut kun de kolonnene som er av interesse for å få den markedsanalysen man ønsker. På grund av Column stored DB sine sin gode evne til å utføre aggregeringsoperasjoner ville lagring og håndtering av slik type data passet denne.   
Med slike dataer vil det også hende at ny informasjon om dataen må legges til. Dette gjøres enkelt i denne type database-modellen ved å kun legge til en ny kolonne.

<https://www.columnardatabase.com/>

### 1.1.3 Spørringer på dette datasettet

Describe queries which might be useful to run on that dataset. You do not need to implement those queries to an appropriate query language, but you should be able to describe what the parameters of the queries will be, how the data model can be leveraged to answer those queries, if it had been implemented.

**Aggregeringer‼!**

Cassandra benytter Cassandra Query Language (CQL).

Hente data

SELECT \* from tabell WHERE . . .

Where kan kun brukes på kolonner som er primary key ELLER er indexert

### 1.1.4 Beskrivelse av datasettet og hvorfor dette passer

If the process that created the data is described in the dataset description, argue for why this process and throughput makes this data model suitable. If the origin process is not already described, imagine a process that might be a reasonable source of the data, and argue for why this imagined process and expected throughput makes this data model suitable.

### 1.1.5 Ved annen bruk av dataene

There are some situations were long-term storage, day-to-day use and historical analysis of the data presents different requirements for the storage technology, thereby necessitating different choices for each stage in thelife-cycle of the dataset. What are your thoughts concerning this for your datasets? Will a different storage technology be more appropriate for a different use of the same data?

## 1.2 Graph Database - Neo4j

### 1.2.1 Om Graph Database & Neo4j

Med Graph Databaser er data representert som noder og kanter. Hver node i databasen representerer en enhet (For eksmepel en person på Twitter) med informasjon i key-value par og kantene representerer relasjonen mellom disse (For eksempel er «venn til»). Disse relasjonene er viktig i Graph databaser. Disse relasjonene gjør det lett og med korte spørringer mulig å få ut data som med RDBMS ville krevd kompliserte spørringer med kanskje flere joiner.

**Ulike typer grafer innenfor denne:**

* Undirected graphs: Noder og relasjoner er utskiftbare og går begge veier. Venner på Facebook er et eksempel på denne typen
* Directed graphs: Noder og relasjoner kan ikke gå begge veier. Et eksempel på dette er følgere på Twitter hvor en kan følge en annen uten at denne personen følger tilbake.
* Graphs with weight: Relasjoner mellom noder er vektet noe som gjør det mulig å utføre noen operasjoner før eller etter andre.
* Graph with labels: Denne typen har en slags merkelapp som forteller hva slags node der er, samt relasjon det er mellom noder. Eksempel:
  + En node kan være en Person, en forballspiller og Student  
    Relasjoner kan være venner, kollega, søster osv.
* Property graph: Den mest komplekse typen hvor vi kan tildele noder og relasjoner ulike egenskaper. Eksempel:
  + En node Person har egenskaper som id, navn, alder osv. som legges om key-value par

Relasjonen kan ha id.

<https://bbvaopen4u.com/en/actualidad/neo4j-what-graph-database-and-what-it-used>

**Fordeler med Graph Database:**

* Høy og stabil ytelse (Spesielt når det kommer til data som skal være koblet sammen):
  + Graph DB er rask da kun den delen av databasen som trengs for å respondere på spørringen trenger å traverseres.
  + Ytelsen holder seg som oftest til tross for økning av data.
* Fleksibelt:
  + Lett å endre innhold og relasjonen mellom dataene
  + Passer til den smidige måten systemer bli laget på i dag, ved at ting kan endres underveis.
* Lett å gjøre spørringer: Ingen joiner.

**Egner seg til:**

Graph Databaser egner seg meget godt til graflignende spørringer, som for eksempel å finne korteste vei mellom to eller flere elementer.

**Neo4j - Eksempel på en Graph Database**

Neo4j er den ledende Graph Database – motoren. Denne benytter altså grafer for å representere data og relasjonen mellom disse og gjør det lett, med språket Cypher (CQL), å gjøre spørringer.

<https://stackshare.io/stackups/cassandra-vs-neo4j>

### 1.2.2 Datasett – Facebook Data

**For Graph Database har jeg valgt følgende datasett:**Facebook Data - Exploratory Data Analysis giving insights from Facebook dataset  
 **Link til datasett:**<https://www.kaggle.com/sheenabatra/facebook-data>

**Hvorfor denne datamodellen?:**

Dette datasettet inneholder informasjon om ulike personer, samt en relasjon til andre. Ved å lagre dette som en Graph DB ville det vært enkelt å lage hver person som en node, sette informasjonen som navn, alder og fødselsdato som egenskaper, samt kunne sette ulike relasjoner til andre noder. Det vil være naturlig å hente informasjon om venner og venners venner ut ifra et slikt datasett, noe som Graph DB løser enkelt. Både ved å gjøre spørringen mindre avansert og raskere da det kun trenger å gå igjennom delen av databasen som kreves for å kunne svare på spørringen.   
Da Graph DB gjør det enkelt å endre på innhold og relasjoner mellom noder passer dette slik type data bra, da denne dataen ofte kan endres ved at man for eksempel får nye venner.   
Facebookdata er også noe som typisk vil kunne vokse ganske raskt i størrelse. Dette er også noe Graph DB håndterer fint uten at det påvirker ytelsen.

### 1.2.3 Spørringer på dette datasettet

Typiske spørringer på dette datasettet kan være (x er en node:person i datasettet)

1. Hvem er x sine venner
2. Hvem er venner til x sin venner
3. Hvem av x sine venner som «liker VG»

Her blir id-en til x, samt i den siste spørringen september, sendt med som parametre.   
Graph DB løser en slik spørring enkelt ved å gå til noder som med relasjonstype «venner» er koblet til node x, og eventuelt videre fra disse nodene videre.

MATCH (x:person)-[:FRIEND-WITH]→ (friend: person) -[:LIKES]→ (:Website {name: “VG”});

### 1.2.4 Beskrivelse av datasettet og hvorfor dette passer

**Datasett-beskrivelse**

This exploratory data analysis gives insights from Facebook dataset which consists of identifying users that can be focused more to increase the business. These valuable insights should help Facebook to take intelligent decision to identify its useful users and provide correct recommendations to them.

**Hvorfor Graph DB egner seg til dette**Som beskrevet over vil dette datasettet hjelpe til med å kunne gi riktig anbefalinger til riktige brukere. For å gjøre dette kreves det at databasen raskt kan søke igjennom noder (brukere) som har en relasjon til ulike sider eller andre personer. Noe som, også nevnt over, Graph DB gjør.

### 1.2.5 Ved annen bruk av dataene

There are some situations were long-term storage, day-to-day use and historical analysis of the data presents different requirements for the storage technology, thereby necessitating different choices for each stage in thelife-cycle of the dataset. What are your thoughts concerning this for your datasets? Will a different storage technology be more appropriate for a different use of the same data?

## 1.3 Key-value database - Riak

### 1.3.1 Om key-value database & Riak

. . . .

### 1.3.2 Datasett

Find a dataset in Kaggle’s dataset directory (<https://www.kaggle.com/datasets>) and argue for why this data set is best stored and used using this data model.

### 1.3.3 Spørringer på dette datasettet

Describe queries which might be useful to run on that dataset. You do not need to implement those queries to an appropriate query language, but you should be able to describe what the parameters of the queries will be, how the data model can be leveraged to answer those queries, if it had been implemented.

### 1.3.4 Beskrivelse av datasettet og hvorfor dette passer

If the process that created the data is described in the dataset description, argue for why this process and throughput makes this data model suitable. If the origin process is not already described, imagine a process that might be a reasonable source of the data, and argue for why this imagined process and expected throughput makes this data model suitable.

### 1.3.5 Ved annen bruk av dataene

There are some situations were long-term storage, day-to-day use and historical analysis of the data presents different requirements for the storage technology, thereby necessitating different choices for each stage in thelife-cycle of the dataset. What are your thoughts concerning this for your datasets? Will a different storage technology be more appropriate for a different use of the same data?

## 1.4 Document Store Database - Mongo DB

### 1.4.1 Om Document Store Database & Mongo DB

. . . . . .

### 1.4.2 Datasett

Find a dataset in Kaggle’s dataset directory (<https://www.kaggle.com/datasets>) and argue for why this data set is best stored and used using this data model.

### 1.4.3 Spørringer på dette datasettet

Describe queries which might be useful to run on that dataset. You do not need to implement those queries to an appropriate query language, but you should be able to describe what the parameters of the queries will be, how the data model can be leveraged to answer those queries, if it had been implemented.

### 1.4.4 Beskrivelse av datasettet og hvorfor dette passer

If the process that created the data is described in the dataset description, argue for why this process and throughput makes this data model suitable. If the origin process is not already described, imagine a process that might be a reasonable source of the data, and argue for why this imagined process and expected throughput makes this data model suitable.

### 1.4.5 Ved annen bruk av dataene

There are some situations were long-term storage, day-to-day use and historical analysis of the data presents different requirements for the storage technology, thereby necessitating different choices for each stage in thelife-cycle of the dataset. What are your thoughts concerning this for your datasets? Will a different storage technology be more appropriate for a different use of the same data?

# 2 Practical use of one of the storage technologies

Here, you will choose one of the datasets from section 2.1 and use the associated database engine you have learned about in the course, namely, Cassandra, neo4j, Riak or Mongo DB to implement your solution. If you described an extensive solution for that dataset, you don’t need to implement everything but your implementation should support the following:

* Importing the dataset into the database
* Running one of the queries you described
* Adding, removing or modifying the data in the database using procedures with names and behavior which are a natural part of a use-case for the dataset you selected.

Keep in mind that demonstrating an understanding beyond the barest rudiments will be beneficial for your grade. Also, while the examples in the lectures will be using Java, you are free to use either Java, Scala, or Python for your work.

Delivery

The delivery should be a zip file containing the following parts.

3.1 Source code

You should provide the source code you have used in this project as part of the

zip file using a folder and naming scheme which clearly identifies the purpose of

each file.

3.2 Report

You should provide the report for this project as a single PDF file. The contents

of the report are expected to be fairly extensive.

# 3 Kildeliste:

<https://www.guru99.com/cassandra-tutorial.html>

<https://intellipaat.com/blog/tutorial/cassandra-tutorial/brief-architecture-of-cassandra/>

<https://stackshare.io/stackups/cassandra-vs-neo4j>

<https://bbvaopen4u.com/en/actualidad/neo4j-what-graph-database-and-what-it-used>