Marius Begby, Daniel Nilsen Johansen, Bharati Mainali

Oppgave 1 - NoSQL i teori

Dere skal nå finne ut hva NoSQL er, hva det betyr og når dette brukes. I tillegg skal dere sammenlikne NoSQL med de tradisjonelle Relasjonsdatabasene. Hva er fordeler og ulemper med de forskjellige database-typene, og når foretrekker man den ene over den andre?

Svaret på oppgaven skal leveres slik at den best mulig og mest oversiktlig videreformidler den kunnskapen dere har tilegnet dere om emnet.

Hva er NoSQL, og hvilken betydning har denne typen databaser?

NoSQL, også kalt “Not only SQL” er en samling av databasesystemer som skiller seg fra tradisjonelle “RDBMS” (Relational Database Management System) databaser, slik som MySQL.

NoSQL databaser blir ofte hyllet for å ha høy funksjonalitet, enkelt og rask utvikling samtidig som at de skalerer bra i større systemer. De er veldig fleksible, og egnes spesielt bra til å lagre ustrukturerte data, noe som tradisjonelle databaser ikke er like godt egnet til. NoSQl databaser kan lagre strukturerte, semistrukturerte og ikke-strukturerte data. Dette gjøres vanligvis på ett av følgende vis: Dokument-basert lagring, key-value lagring, kolonne-basert lagring eller graf-basert lagring. Dette er de fire store typene av NoSQL databaser.

Dokument databaser - lagrer data i dokumenter lignende slik som JSON objekter, og hvert dokument inneholder felter som holder på verdiene. Verdiene kan igjen være felter, eller datatyper slik som strenger, tall, boolean, lister eller objekter. I dokument databaser kan man søke (query) på både feltet og verdiene.
Key-value databaser - her inneholder hvert element en nøkkel og en verdi, og er en “simpel” type NoSQL database. Fordelen er at spørringer ofte går mye raskere, men funksjonaliteten er begrenset. Her kan man bare søke på nøkler, ikke verdier.
Kolonne databaser - disse NoSQL databasene lagrer data i tabeller, rader og dynamiske kolonner, og er egnet for høyt analytisk og komplekse spørringer. Disse bruker mindre minne for å gi ut data fra spørringer.
Graf databaser - lagrer data i noder og forholdene mellom nodene. Dette kan typisk være personer, steder og andre ting hvor forholdet mellom dem kan være viktig.

Den største ulempen med NoSQL databaser er som regel ikke har noe særlig god transaksjonshåndtering, men dette er ikke nødvendigvis alltid sant. F.eks. har MongoDB, en veldig populær NoSQL dokumentdatabase, god ACID(Atomicity, Consistency, Isolation, Durability) transaksjonshåndtering. Dermed er ulempene veldig få, og NoSQL databaser slik som MongoDB blir i større og større grad tatt i bruk. Behovet for hvilke type data og hvordan vi vil lagre det endrer seg med tiden, og NoSQL databaser er ofte et godt valg i dag.

Marius Begby, Daniel Nilsen Johansen, Bharati Mainali

Sammenlikning av NoSQL mot tradisjonelle relasjonsdatabaser

	NoSQL	**Tradisjonelle RDBMS**
Eksempler på databaser	MongoDB(dokument), Redis(key-value), Cassandra(kolonne), Neo4j(graf)	MySQL, Oracle, PostgreSQL
Lagringsmodell	Dokument: JSON dokumenter Key-value: Nøkkel-verdi par Kolonne: Tabeller med rader og dynamiske kolonner Graf: Noder og relasjoner	Tabeller med kolonner og rader.
Brukes til	Dokument: Generelle applikasjoner og lagring Key-value: Store mengder data med enkle spørringer Kolonne: Store mengder data med forutsigbare spørremønstre Graf: Analysering og følge relasjoner mellom data	Generelle applikasjoner og lagring
Database skjemaer	Fleksible	Fastsatte
ACID transaksjoner	Ikke vanlig, men støttes f.eks. i MongoDB	Støttet
Sammensetning av data(joins)	Vanligvis ikke behov	Vanligvis nødvendig/behov
Skalerbarhet	Horisontal skalering (kan skalere på flere, men små servere)	Vertikal skalering (krever større og mer kostbare servere)
Sikkerhetskopiering	Mindre støtte/funksjonalitet for det. F.eks. har MongoDB noen løsninger, men som regel er ikke løsningene til NoSQL veldig utprøvd.	Stor støtte for dette, og godt prøvd.

Marius Begby, Daniel Nilsen Johansen, Bharati Mainali

Oppgave 2 - NoSQL i praksis

*Dere skal nå ta for dere **minst to** forskjellige typer NoSQL databaser. De dere har valgt skal sammenliknes og dere skal finne et gratis verktøy, gjerne online hvor dere kan sette opp en database av en av disse typene. Mot databasen skal dere gjøre noen enkle spørringer.*

Svaret skal leveres på en slik måte at det kommer tydelig frem forskjeller og likheter mellom de typene dere har valgt å se nærmere på. Det skal også komme tydelig frem hvilket verktøy dere har benyttet for å opprette en database, gjerne med lenker eller for eksempel en liten video som viser hvilket program dere har benyttet og hvordan dere bruker programmet. Selve databasen skal dere vise/beskrive og spørringer og resultater skal inkluderes i sluttrapporten.

MongoDB

Først installerte vi mongodb og startet serveren. Vi installerte også verktøyet mongosh for å gjøre spørringer opp mot MongoDB databasen. Når vi koblet til serveren forsøkte vi å kjøre forskjellige spørringer, slik som: bytte/velge database, legge inn data, endre data, hente ut data. Dette gikk overraskende fint, og vi oppdaget med en gang at MongoDB er ekstremt fleksibelt. Man trenger ikke på forhånd å opprette tabeller eller databaser, du bare skriver at du skal bruke databasen “oblig5”, også er det fikset. Samme med “tabeller”, vi skrev at vi skulle legge inn data i “oppgave”, uten å ha opprettet dette på forhånd, og fikk beskjed at spørringen gikk gjennom.

Spørringer

```
test> use ("test2");
switched to db test2
test2> use ("oblig5");
switched to db oblig5
oblig5> db.oppgave2.insertOne({"_id": 1, "navn": "test", "nummer": 1})
{ acknowledged: true, insertedId: 1 }
oblig5> db.oppgave2.find()
[ { _id: 1, navn: 'test', nummer: 1 } ]
oblig5> db.oppgave1.find()

oblig5> db.oppgave2.insertOne({"_id": 2, "navn": "Dette er en lengre test", "nummer": 7})
{ acknowledged: true, insertedId: 2 }
oblig5> db.oppgave1.find()

oblig5> db.oppgave2.find()
[
  { _id: 1, navn: 'test', nummer: 1 },
  { _id: 2, navn: 'Dette er en lengre test', nummer: 7 }
]
oblig5> db.oppgave2.findOne()
{ _id: 1, navn: 'test', nummer: 1 }
oblig5> db.oppgave2.updateOne({ navn: "test"}, {$set: {nummer: 2}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
oblig5> db.oppgave2.find()
```

```
[
  { _id: 1, navn: 'test', nummer: 2 },
  { _id: 2, navn: 'Dette er en lengre test', nummer: 7 }
]
oblig5>

oblig5> db.oppgave2.updateOne({ navn: "test"}, {$set: {nummer: 7}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
oblig5> db.oppgave2.find()
[
  { _id: 1, navn: 'test', nummer: 7 },
  { _id: 2, navn: 'Dette er en lengre test', nummer: 7 }
]
oblig5> db.oppgave2.updateOne({ nummer: 7}, {$set: {nummer: 8}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
oblig5> db.oppgave2.find()
[
  { _id: 1, navn: 'test', nummer: 8 },
  { _id: 2, navn: 'Dette er en lengre test', nummer: 7 }
]
oblig5> db.oppgave2.updateOne({ nummer: 7}, {$set: {nummer: 8}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
oblig5> db.oppgave2.find()
[
  { _id: 1, navn: 'test', nummer: 8 },
  { _id: 2, navn: 'Dette er en lengre test', nummer: 8 }
]
oblig5> db.oppgave2.updateMany({ nummer: 8}, {$set: {nummer: 7}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
oblig5> db.oppgave2.find()
[
  { _id: 1, navn: 'test', nummer: 7 },
  { _id: 2, navn: 'Dette er en lengre test', nummer: 7 }
]
```

Redis

Redis var faktisk litt mer komplisert å få i gang. Det gikk fint å installere selve serveren, men den hadde ikke noe CLI verktøy for å skrive spørringer. Vi fant et tredjepartsverktøy, iredis, som vi utførte spørringene med. Etter dette gikk det fint. Vi fikk inntrykket at key-value databaser slik som redis er ekstremt simple, men også raske og effektive. I spørringene satt vi verdier, inkrementerte dem og hentet dem ut.

```
127.0.0.1:6379> SET verdi1 "Dette er en test"
OK
127.0.0.1:6379> GET verdi1
"Dette er en test"
127.0.0.1:6379> SET tall 7
OK
127.0.0.1:6379> GET tall
"7"
127.0.0.1:6379> INCR tall
(integer) 8
127.0.0.1:6379> INCR tall
(integer) 9
127.0.0.1:6379> INCR tall
(integer) 10
127.0.0.1:6379> GET tall
"10"
```

Sammenlikning

	MongoDB	**Redis**
Databasetype	Dokumentbasert	Key-value basert
Hastighet	Rask, bruker disk	Raskere, bruker RAM
Skalerbarhet	Skalerer bedre, krever ikke like mye RAM	Skalerer dårligere, krever mer RAM
Brukervennlighet	Bedre enn SQL	Bedre enn SQL
Tid nødvendig for å lære	Mindre enn SQL	Mindre enn SQL
Kryptering	Bygget inn	Ikke bygget inn

Marius Begby, Daniel Nilsen Johansen, Bharati Mainali

Oppgave 3 - Refleksjon

Skriv et lite refleksjonsnotat som gruppe om hvordan det har gått å arbeide med denne innleveringen. Si noe om hvem som har bidratt, og om noen har bidratt veldig mye eller veldig lite. Dere skal også si noe om hvordan dere har arbeidet som gruppe, om dere arbeidet digitalt eller møttes fysisk for eksempel. Dere skal også levere et lite avsnitt hvor dere skal drøfte hvorvidt det var positivt eller negativt å gjennomføre denne obligen som workshop istedenfor som vanlig forelesning.

Vi møtte fysisk som gruppe, mest sannsynlig fordi vi alle bor på Remmen, og kan enkelt møte opp og samarbeide på skolen, og også mtp. at vi kjenner hverandre fra før. Det var positivt å jobbe i gruppe med tanke på at man får forskjellige synspunkter på ting.

Når vi skulle finne informasjon om NoSQL og sammenlikne med RDBMS satt vi hver for oss og Googlet, og skrev inn i et felles dokument.

Vi avtalte først en tid til å møte opp og starte på oppgaven, etter dette møte hadde vi fortsatt noe igjen på obligen, så vi avtalte et nytt møte. Totalt brukte vi ca. 6 timer sammen til å arbeide på oppgavene.

Vi har ikke benyttet oss av øvingstimer for å gjøre oppgavene, men skal møte opp for å vise frem/levere.

Kilder

https://www.mongodb.com/nosql-explained
https://www.mongodb.com/nosql-explained/nosql-vs-sql
https://www.mongodb.com/basics/acid-transactions
https://www.techopedia.com/7/31991/storage/what-is-the-difference-between-a-nosql-database-and-a-traditional-database-management-system
https://www.geeksforgeeks.org/advantages-and-disadvantages-of-sql/
https://technologypoint.in/advantages-and-disadvantages-of-nosql-databases/
https://hevodata.com/learn/redis-vs-mongodb/