DAB - Noter

Alexander Rasmussen

27. januar 2018

Indhold

1 Lektion 1 - Objektorientering (OO) og databaser

1.1 Forberedelse

Forbedredelse til første lektion i database, som omhandler databaser generelt.

- 1.1.1 og 1.1.2 Handler om "Hvad er en database/ et DataBase Management System"
- 1.1.3 og 1.1.4 Handler om "Beskrivelse af nogle OO koncepter"
- 1.1.5 og 1.1.6 Handler om "Polyglot persistering"
- 1.1.7 og 1.1.8 Handler om "NoSQL databaser"
- 1.1.9 og 1.1.10 Handler om "SQL databaser"
- 1.1.11 Handler om "Lagdelt arkitektur"

1.1.1 Wiki - Database - Link til kilde

Database: Er typisk en samling data. Den består typisk af en række skemaer, tabler, queries rapporter og andre elementer. Den bruges til at holde styr på fx hvor mange rum der er ledige på et hotel.

Database-management system (DBMS): Er et stykke software som end-user inteagere med databasen med, samt andre programmer.

Disse database kan nogle gange dække over begge begreber.

DBMS dækker over en række funktioner som kan blive delt ind i 4 hovedgrupper

- **Data definition** - Creation, modification and removal of definitions that define the organization of the data.
- **Update** - Insertion, modification, and deletion of the actual data.
- **Retrieval** - Providing information in a form directly usable or for further processing by other applications. The retrieved data may be made available in a form basically the same as it is stored in the database or in a new form obtained by altering or combining existing data from the database.
- **Administation** - Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information that has been corrupted by some event such as an unexpected system failure.

DBMS kan være general-purpose eller special-purpose. Generel-purpose er fx adabas, Oracle, MySQL, Microsoft SQL server, da dette kan skabe overhead da de kan ret meget, kan det tit være sådan at man bruger et speciel-purpose. Det kan fx være når man arbejder med emails, og så kommunikere man med databasen, igennem et API eller andet så man ikke, viser hvordan man snakker med DBMS direkte.

Next gen post-relational databaser er NoSQL og NewSQL

Vigtige ting for en Database, er Performance, sikkerhed, og tilgenelighed.

1.1.2 DB-Engines - Database - Link til kilde

En database er en samling af data, som er styret af noget specifikt softwar (DBMS). DBMS og en Database er samlet et database system. En database er meget mere end bare user data, det indholder også andet data til bl.a. drift - det kan være log filer mm.

1.1.3 LosTechies - Entities, Value Objects, Aggregates and Roots - Link til kilde

- **Entities:** Handler om at mange objekter måske fundamentalt er det samme men faktisk ikke er det, det gælder fx. når man kigger på personer eller lignede. Så kan man hedde det sammen uden at være den samme. Når du så har med samme person at gøre i følge navn kan de give dem et stedfortræder ID. CPR-Nummer vil jeg mene er et sådan ID, da der er du den samme person i alle systemer, det er et unik ID på dig som person. Kig 1.2.1
- **Value Objects:** Handler om at man har mange af det samme objekt der conceptuelt er ens. Fx en spand maling med den samme farve er jo ens. Vi er her ligeglad med om vi tager den ene eller den anden spand med blå maling.
- **Agggregates and Roots:** Dette bruges til at trække en kant rundt om en eller flere entities. Hver aggregates har en Root entity, som er det eneste en medlem udefra har lov til at holde en reference til inde i Aggregaten.

  Der er en række regler for dette som kan ses nedenfor:

  - The root Entity has global identity and is ultimately responsible for checking invariants
  - Root Entities have global identity. Entities inside the boundary have local identity, unique only within the Aggregate.
  - Nothing outside the Aggregate boundary can hold a reference to anything inside, except to the root Entity. The root Entity can hand references to the internal Entities to other objects, but they can only use them transiently (within a single method or block).
  - Only Aggregate Roots can be obtained directly with database queries. Everything else must be done through traversal.
  - Objects within the Aggregate can hold references to other Aggregate roots.
  - A delete operation must remove everything within the Aggregate boundary all at once
  - When a change to any object within the Aggregate boundary is committed, all invariants of the whole Aggregate must be satisfied.

Hvis man kort skal opsummere så kan man sige at vi repræsentere vores rigtige verden i en conceptuel model, og vores conceptuelle model er lavet i kode i form af entities og Value objects. For så at gøre denne model simplere så bruger vi Aggregates and Roots, som styrker de konstante ting ved operationerne.

1.1.4 LosTechies - Driving and refining Ubiquitous Language - Link til kilde

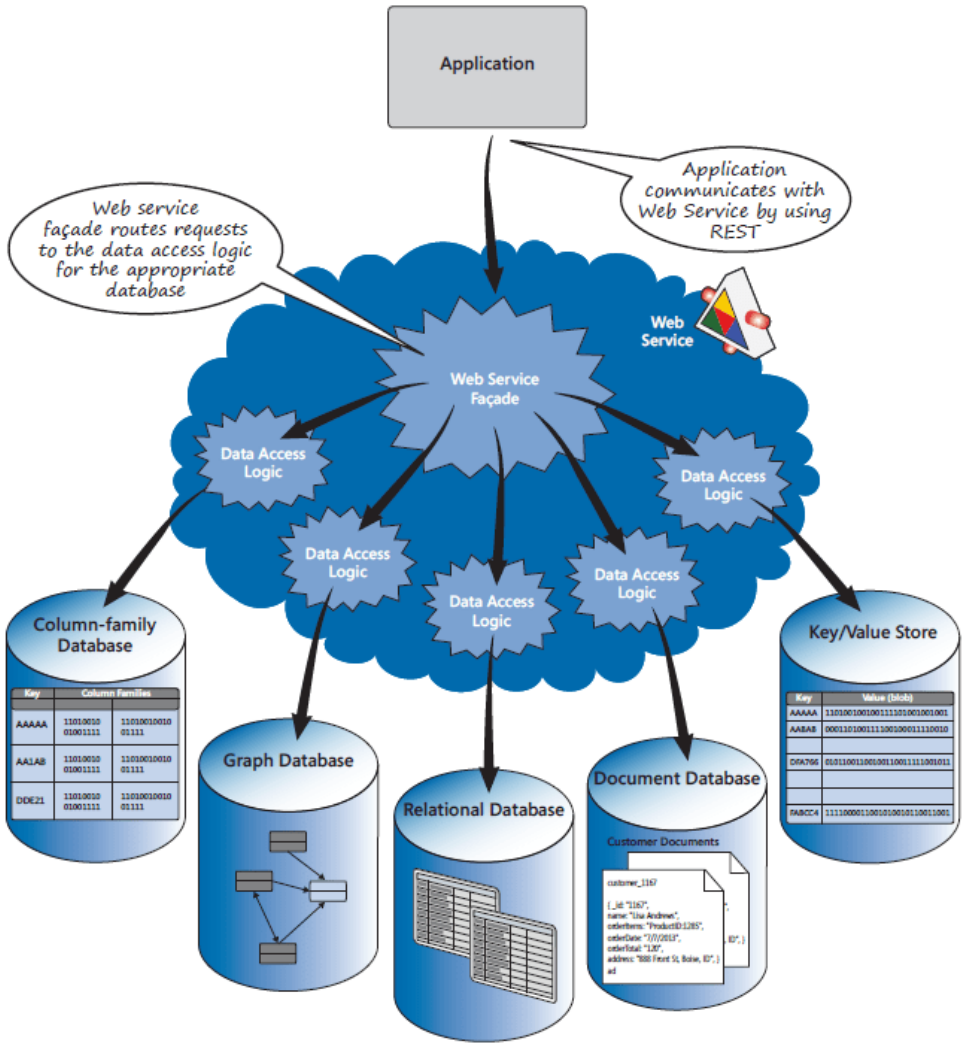
Vigtigt citat: "Language drives the model, the model can represent the language, but the model never drives the language unless through user stories."For at sikre at sproget ikke bliver korrupt af modellen eller systemet skal der være nogle ting i teamet:

- Ærlighed
- Tillid
- Kommunikation

1.1.5 James Serra - What is Polyglot Persistence? - Link til kilde

Polyglot Persistence: Det betyder at man gemmer data på forskellige teknologier for at tilpasse sig til den type data man skal gemme. Der er forskellige typer data der gemmes bedst på forskellige typer data stores.

Hvis man tager et eksempel kan man fx kiggede på billedet nedenfor.



Figur 1: Eksempel på hvordan polyglot persistence kan blive brugt

På 1 kan det ses hvordan forskellige ting kan gemmes forskelligt. På 1 kan der ses en guideline for hvilke database typer der skal bruges baseret på den data der skal gemmes

Tabel 1: Polyglot Persistence

Functionality	**Considerations**	**Database Type**
User Sessions	Rapid Access for reads and writes. No need to be durable.	Key-Value
Financial Data	Needs transactional updates. Tabular structure fits data.	RDBMS
POS Data	Depending on size and rate of ingest.,Lots of writes, infrequent reads mostly for analytics.	RDBMS (if modest), Key Value or Document (if ingest very high) or Column if analytics is key.
Shopping Cart	High availability across multiple locations.,Can merge inconsistent writes.	Document, (Key Value maybe)
Recommendations	Rapidly traverse links between friends, product purchases, and ratings.	Graph, (Column if simple)
Product Catalog	Lots of reads, infrequent writes. Products make natural aggregates.	Document
Reporting	SQL interfaces well with reporting tools	RDBMS, Column
Analytics	Largescale analytics on large cluster	Column
User activity logs,CSR logs, Social Media analysis	High volume of writes on multiple nodes	Key Value or Document

Det kan helt sikkert godt betale sig at lærer de forskellige teknologier der skal bruges til de forskellige elementer af data, da det i sidste ende, giver et meget bedre output set i forhold til performance, dog bliver systemet mere komplekst. Med NoSQL kan man nemmere scale ud, i forhold til ewlational databaser, som er en smule begrænset der.

1.1.6 Martin Fowler - PolyglotPersistence - Link til kilde

Meget det samme som 1.1.5 der er dog nogle gode punkter der er værd at nævne.

Det er vigtigt når man går i gang man gør sit mål klart så det er nemt at finde ud af hvilken type data storage man skal bruge. I stedet for relational, bør man også kigge på andet. Selvdølgelig kan det skabe en større kompleksitet, men det er fremtiden. Samtidig er det nemt at lave et skift over til NoSQL, det har bl.a. The Gaurdian gjort.

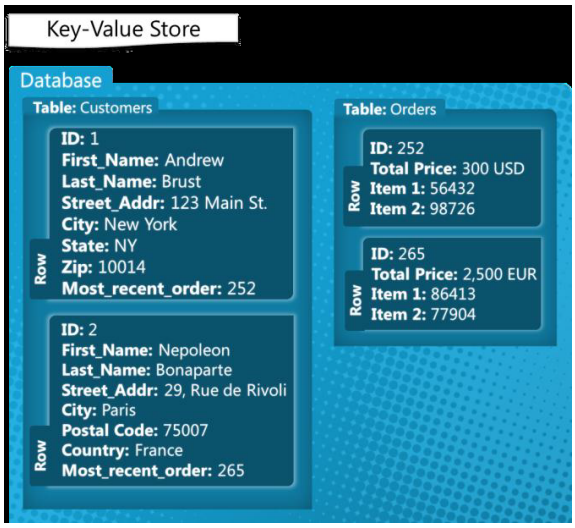
1.1.7 Microsoft - Getting Acquainted with NoSQL on Windows Azure - Link til kilde

NoSQL bliver for det meste brugte, i det offentlige, store web sites hvor det er vigtigt hurtigt at kunne få data.

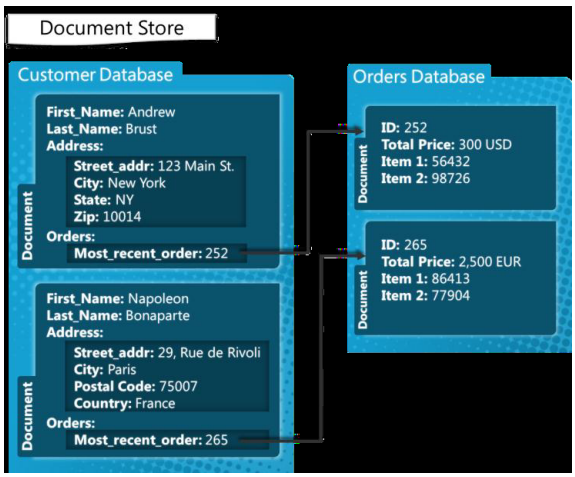
NoSQL kan tilbyde følgende feature:

- **Key-value stores:** - Et key value par kan være mobilnummer som hænger sammen med et nummer. Key value stores kan bruges til at holde ordbøger, liste af produktet osv. generelt hvor der er brug for lister til at holde data. Values kan sagtens være lange tekster.

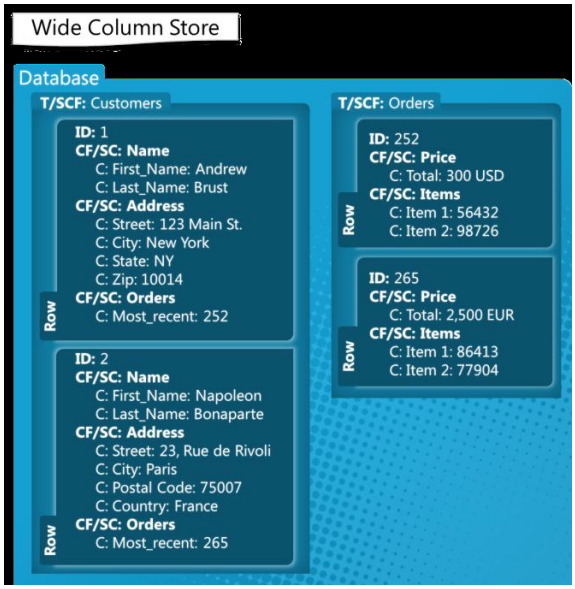
- **Document stores:** - En NoSQL database som behandler records som documenter. Disse dokumenter kan være en unik URL. Man kan få documents stores for kay-value stores. Se evt billede nedenfor
- **Wide column stores:** - Se billede?
- **Graph databases:** - Kigger på entities i et domæne og kigger på sammenhængen mellem dem. En entity er en node og relationen er kanter. Vil tro det kan bruges til at finde ud af venner på facebook, eller måske forbindelser på linkedin.



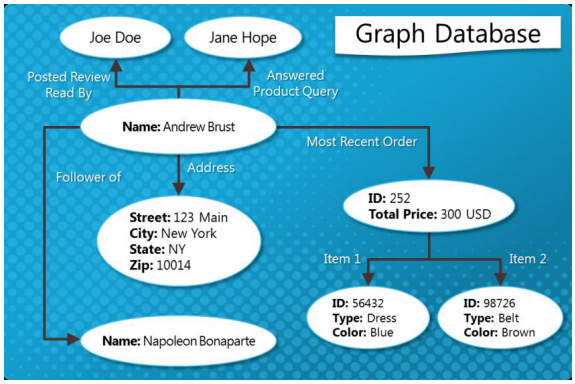
Figur 2: Eksempel på key-value store fra microsoft



Figur 3: Eksempel på document store fra microsoft



Figur 4: Eksempel på wide column store fra microsoft



Figur 5: Eksempel på Graph database fra microsoft

Generelle NoSQL træk NoSQL skal tit bruges queries, og disse queries bliver brugt op og bliver hentet forskellige steder, det kan være forskellige servere og lignede. Denne metode kaldes map-reduce acknowledges. Map steppet er til at sprede queries ud, mens reduce er det der sammen smelter det hele.

Mange NoSQL databaser burger "eventual consistency"som betyder at når der kommer en database ændring så vil den blive transmittet asynkront ud til de andre dele. Dog bruger ikke alle denne model, der er nogle der er fully transactional.

MongoDB gemmer ikke data som en normal Database, men mere som noget der minder om JSON, det kaldes BSON i MongoDB.

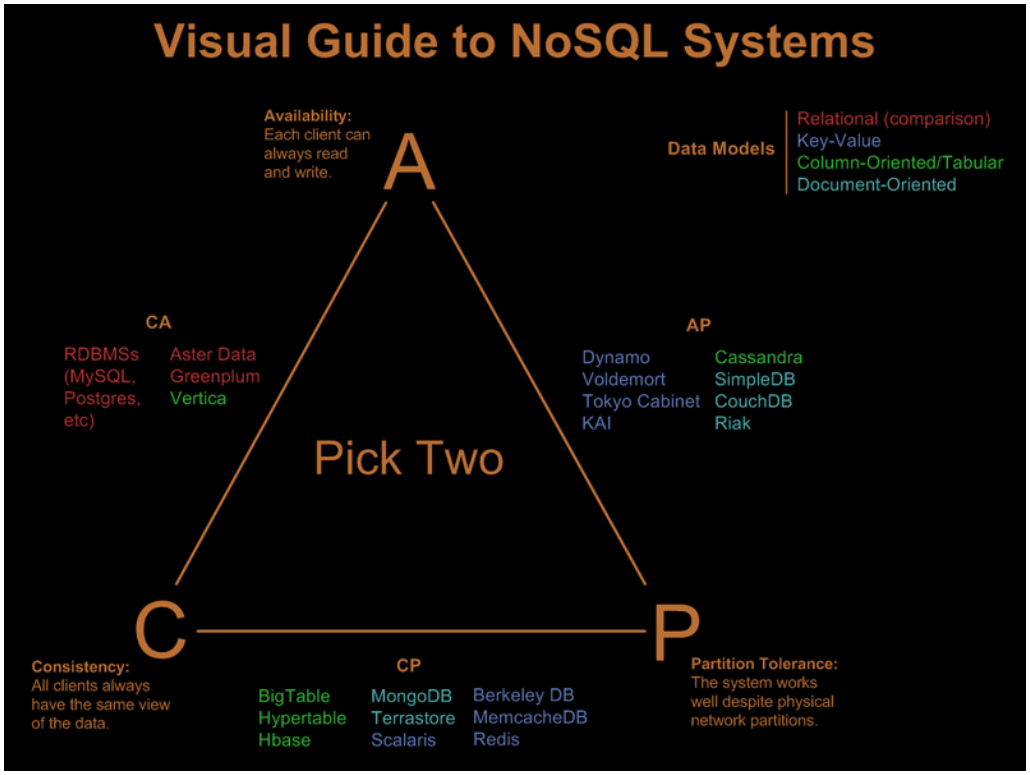
1.1.8 James Serra - Types of NoSQL databases - Link til kilde

Der er en regel som siger at det er umuligt for en computer system simultant at garantere

- Consistency

- Availability
- Partition tolerance

Dette kaldes CAP Theorem se 6



Figur 6: CAP Theorem billede der viser forskellige Databaser inden for de forskellige emner.

Der står mere på denne side omkring hvor man vil bruge hvilke typer DB's.

1.1.9 Wiki - Relation Database Link - Link til kilde

Relational model er en relationsmodel hvor man har flere tabler med rækker og kolloner og så relatere man til dem. Man bruger for det meste en primary key til hver række i en tabel, og den er unik, på den måde kan man relatere dem til hinanden. De fleste Relational DB's bruger SQL

Terminologi

Primary key -> Foreign key - En nøgle som matcher en primary key i en anden tabel.

1.1.10 DB-Engines - Relational DBMS - Link til kilde

Informationen tænker jeg er fint beskrevet i 1.1.9 ellers læs artiklen.

1.1.11 Wiki - Multitier architecture - Link til kilde

Dette er en arkitektur som er en client-server arkitekturen er fysisk sepereret. Det kan kaldes for n-tier arkitektur, da der er n lag. De mest normale lag er:

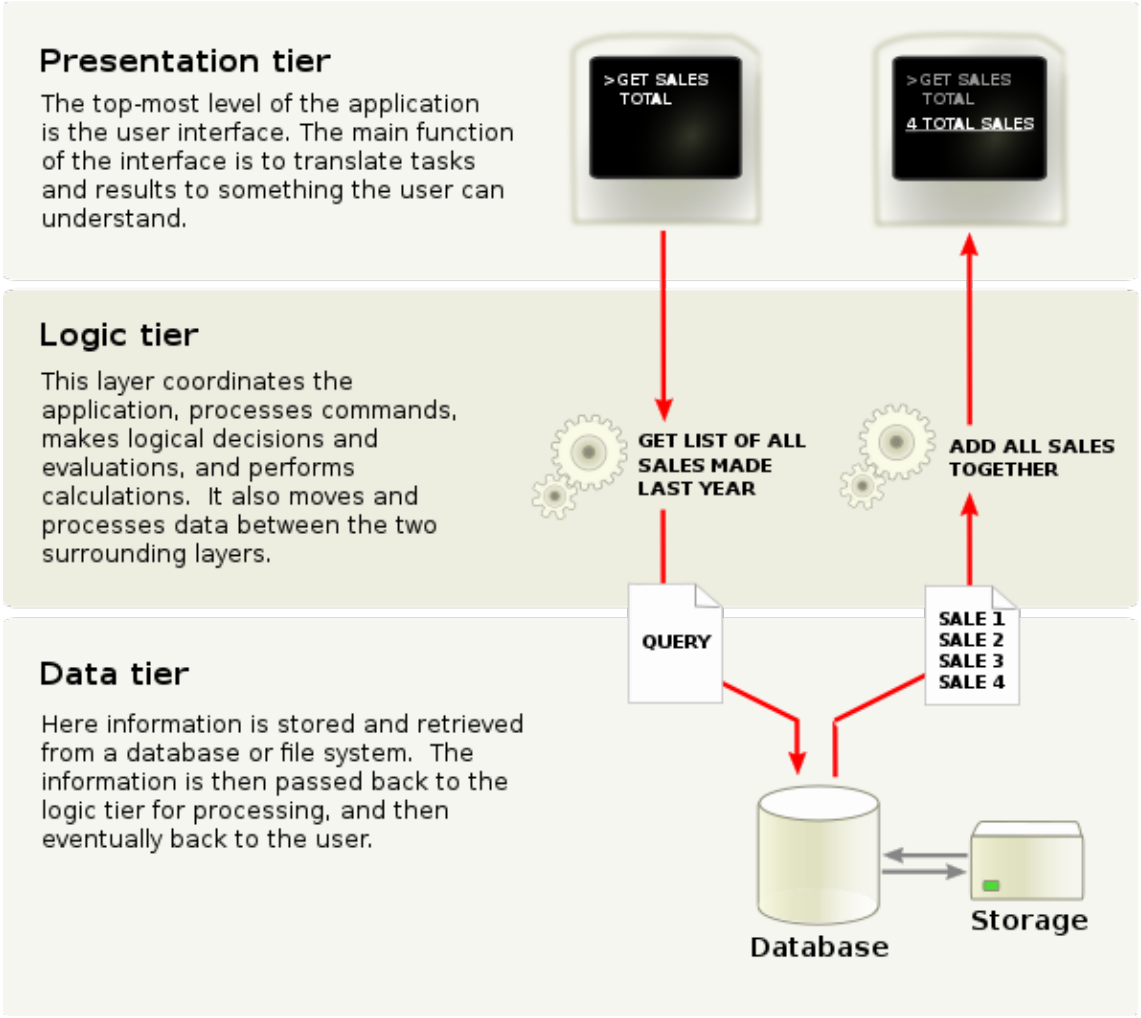
- Prensentation layer - UI, veiw, mm.

- Application layer - Service
- Business layer - domain layer, business logic layer.
- Data access layer - logging, networking persistence layer.

Typisk er applications laget en under lag til Business layer og er encapsuleret af en API.

7 viser de 3 lag:

- **Presentation tier:** Det er det øverste lag. Det er det lag brugeren bruger, ved fx at gøre brug af en GUI eller lignede.
- **Application tier:** Det er det logisk lag fra presentation tier. Det styer applicationens funktionallitet.
- **Data tier:** Data tier har data persistence mechanisms. Diverse servere. Data laget bør tilbyde en API der kan bruges.



Figur 7: CAP Theorem billede der viser forskellige Databaser inden for de forskellige emner.

End-to-emd traceability kan være en svær ting med et n-tier system.

1.2 Lektion

1.2.1 Spørgsmål

Hvad er en entity helt præcist?

Hvad er en value objekt helt præcist?

Hvad er en aggregates and roots helt præcist?

Hvad er en Wide column stores