Note til DAB eksamensspørgsmål 1

Forklar pincipperne bag Entity/Relationship modellering, herunder princippet for et Entity Relationsship Diagram (ERD). Hvorledes opstilles et ER Diagram?

ER modelling

ER modeller bruges til at gengive "real world" datastrukturer.

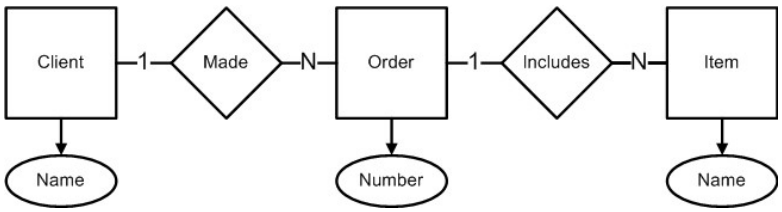
En ER model vil typisk bestå af følgende:

- Entiteter
  - I programmatisk forstand er dette et objekt. En entitet har attributter og relaterer sig til andre entiteter. I en ER model vises en entitet som en firkant (chens notation).
- Relationer
  - Beskriver associationer mellem entiteter Eks. "Har en", "Er en" osv. I en ER model visualiseres dette enten som en rude, eller en streg. Disse har påskrevne multipliciteter
  - Kardinaliteter
    - 1:m
    - m:1
    - 1:1
    - m:m
  - Flere typer af relationer.
    - Binær, Tærtier og Rekursiv, frivillig og generalisering.
    - Pas på med redundante relationer.
- Atributter
  - Ligesom properties. Dette er data gemt i et objekt/entitet. Eks "Fornavn" i entiteten "Person".
  - Nøgler er også attributter og bruges til at identificere andre objekter eller records i samme database.
    - Primær nøgle – En entitets egen nøgle til identifikation.
    - Fremmednøgle – Bruges til at identificere en anden entitets/objekts primærnøgle.
    - Kandidatnøgle – kan bruges som en primær nøgle.

Hvis det giver mening bør hver entitet kun indeholde det som kan repræsentere et enkelt objekt i virkeligheden.

Hvordan opstilles et ER diagram?

1. Find entiteter i systemet
2. Fjern dubletter af entiteter med samme navne.
3. Find attributter for hver af entiteterne, inkl. Nøgler.
4. Find relationerne mellem de opstillede entiteter og beskriv mulitipliciteter.
   a. Pas på med redundante relationer. Fjern hvis nogle.



Normalisering / Normalform

opdeling af normalform - 1nf -> 2nf -> 3nf -> bcnf -> 4nf

Opdeles i lag, og kræver at samtlige foregående lag er opfyldt.

Normalform benyttes til at sikre data integritet (der må ikke være modstridende data) og værne mod redundant data (der må ikke være gentagelser) i en database. Dette gør vi for at undgå uoverenstemmelser og uregelmæssigheder, hvor vi f.eks. risikerer at slette mere data end tiltænkt, eller at der opstår tvivl om hvilke data er gyldige.

Når vi udarbejder en database efter normal form, anvender vi vores data til at afgøre hvordan vores database skal opbygges. Hvis vi følger principperne for konceptuel ER-modellering(EntityRelation), kan vi næsten altid sige at databasen er i 4NF. Ren data proppet ind i en enkelt tabel siges at være i 0NF

Funktionel afhængighed – når 2 eller flere attributer er direkte afhængige af en anden. F.eks. er et bynavn og et postnummer direkte afhængige af hinanden.

1NF - omhandler anvendelse af primære og fremmed nøgler, samt at få adskildt gentagelser, så der ikke opstår redundant data.

Et eksempel kan være en række sensor målinger

Sensor nr	Sensor type	Måletidspunkt	Måling
1	Temperatur	Mandag	20
1	Temperatur	Onsdag	23
2	Luftfugtighed	Mandag	50
2	Luftfugtighed	Onsdag	52
2	Luftfugtighed	Fredag	55

Udfra tabellen kan vi se at der er gentagen data. Hver gang der skal indsættes en ny måling, er det nødvendigt også at indsætte sensor nummer og sensor type. Dette kan give komplikationer ved database operationer som insert, update og delete. Ved at opdele tabellen og angive primære og fremmed nøgler, fjerner vi disse komplikationer

Sensor nr (P-key)	Sensor type
1	Temperatur
2	Luftfugtighed

Sensor nr (F-key)	Måletidspunkt (P-key)	Måling
1	Mandag	20
1	Onsdag	23
2	Mandag	50
2	Onsdag	52
2	Fredag	55

2NF - omhandler situationer hvor en attribut er direkte afhængig af en anden attribut i samme tabel. F.eks. hvis en bestemt afdeling ligger i en bestemt by, så er disse direkte linket til hinanden, og skal placeres i sin egen tabel. Dette vil ofte være opfyldt allerede ved 1NF, da det skaber gentagen data.

3NF - omhandler transitive funktionelle afhængigheder. Disse opstår kun hvis en tabel har mere end 1 attribut som ikke er en nøgle. Man kan derfor sige at tabeller uden dette, som opfylder 2NF, også opfylder 3NF.

Spørgsmål 6 – Relationel Algebra

Hvad er Relationel Algebra (RA)? Kom herunder ind på: relation, tuple, set og operatorer (select, projection, join...) samt RA' betydningen for opbygningen af SQL (Structured Querry Language) erklæringer

Hvad er Relationel Algebra?

En entydig måde at beskrive en database's opførsel på.

Terminologi

Relation	Et set af tupler
Tuple	En samling af attributter
Attribut	En kolonne
Domæne	Typen af data en i kolonne/attribut
Set	En matematisk definition af en samling af objekter i en relation, som ikke indeholder dubletter

Operatorer

For at kunne bruge en operator på et Set skal nogle bestemte betingelser være opfyldt:

- Relationerne (tabellerne) skal have samme antal kolonner/attributter.
- Domænerne skal være ens (kolonnerne skal indeholde de samme type data).

Select

Laver en horisontal partitionering af en tabel og finder hele rækker som opfylder en eller flere betingelser.

Project

Laver en vertikal partitionering og returnere kun de kolonner/attributter, som vi er interesserede i.

Join

Kombinere alle tupler/rækker i tabeller/relationer, som deler en eller flere attributter/kolonner.

RA' betydningen for opbygningen af SQL (Structured Query Language) erklæringer

For at SQL kan bruges til at tilgå en database, skal det laves om til RA, som gør det matematisk og entydigt definerbart hvad der skal ske i forhold til databasen.