

Afsluttende projekt_databaser

Opgavebeskrivelse

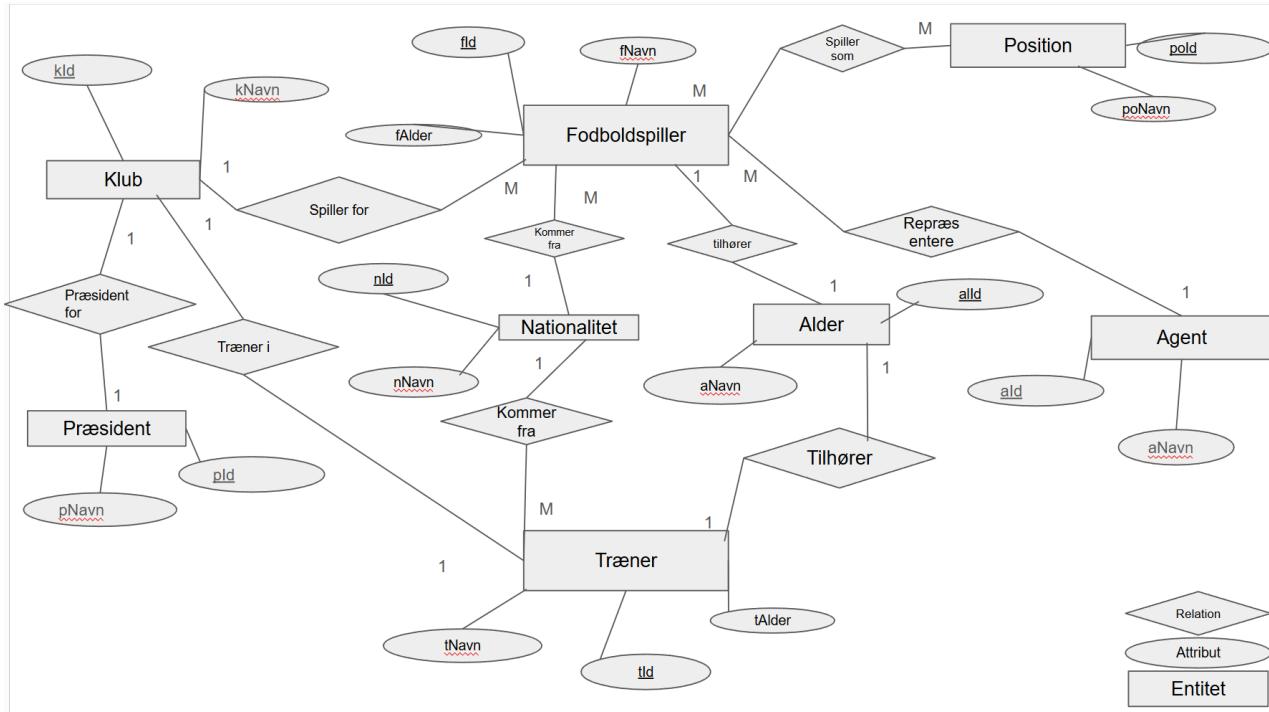
Du/I skal lave en database over en del af virkeligheden (problemområdet), du/I vælger selv hvilken del af virkeligheden.

1. Start med at beskrive **problemområdet**, eks:

Det nye fodboldspil firma UFL vil inkludere den danske Superliga i spillet. De ønsker oplysninger om spillernes navne, alder, position, klub og nationalitet. Derudover skal der registreres information om trænerernes navne, tilknytning til klub og nationalitet. Da de ønsker en karrieremode, skal databasen også indeholde oplysninger om spillernes agentfirmaer samt hvem der er klubpræsidenten.

2. Næste skridt er at identificere mulige entiteter, relationer og attributter samt udarbejde et **E/R-diagram** (konceptuel model) over disse.

Analyse af problemområdet		Attribut (egenskab)
Entitet (Navneord)	Relation (udsagnsord)	
Fodboldspiller	Præsident for	FodboldspillerNavn FodboldspillerId
Klub	Spiller for	FodboldspillerAlder AgentAlder
Nationalitet	Træner i	AgentId AgentNavn TrænerAlder Træner Id
Alder	Er agent for	TrænerNavn TrænerNationalitet
Træner	Kommer fra	FodboldspillerNationalitet PræsidentNavn
Agent	Tilhører	Position Navn Position Id Alder Id Alder Navn
Præsident	Repræsentere	Klub Id National Id
Position	Spille som	National navn



3. Når du/I har afdækket relationsforholdene kan I tage fat på den **relationelle model**, som I efterfølgende normaliserer (hvis du/I føler I kan det).

Relationelle model

Agent (ald, aNavn)

Alder (alld, alNavn)

Position(pold, poNavn)

Nationalitet(nld, nNavn)

Praesident(pld, pNavn)

Klub (kld, kNavn, pld, fld)

Traener(tld, tNavn, kld, nld)

Fodboldspiller(fld, fNavn, alld, ald, kld, nld)

SpillerSom(fld, pold)

Normalisere

Fodboldspiller

Fodboldspiller(fId, fNavn, alld, aId, kId, nId)

Alder

Alder(alld, alNavn)

Agent

Agent(aId, aNavn)

Klub

Klub(kId, kNavn, pId)

Præsident

Præsident(pId, pNavn)

Nationalitet

Nationalitet(nId, nNavn)

Træner

Træner(tId, tNavn, kId, nId)

Position

Position(poId, poNavn)

SpillerSom (*M:N forhold mellem Fodboldspiller og Position*)

SpillerSom(fId, poI)

Normalisere - Forklaring

Fodboldspiller(fld, fNavn, alld, ald, kld, nld)

Alle oplysninger om en spiller er samlet i én tabel, men vi bruger ID'er til at henvise til alder (alld), agent (ald), klub (kld) og nationalitet (nld). Det betyder, at vi ikke gentager navne (som fx aldersgruppe eller klubnavn) direkte i spiller-tabellen.

Vi henviser i stedet via ID'er, hvilket gør det nemmere at undgå gentagelser og opdatere data ét sted.

Alder(alld, alNavn)

Alder (alld) (eller aldersgruppe) har fået sin egen tabel.

Det gør det muligt at ændre navnet på en aldersgruppe ét sted, uden at skulle opdatere hver enkelt spiller.

Det sikrer konsistens i navngivning og data.

Agent(ald, aNavn)

Agenterne (ald) er også gemt separat.

Flere spillere kan dele samme agent (ald), og vi undgår at gentage agentens navn ved at bruge ald som reference.

Klub(kld, kNavn, pld)

Hver klub er gemt med sit eget ID, og præsidenten (pld) for klubben (pld) gemmes også her.

Det betyder, at vi ved, hvilken præsident (pld) der hører til hver klub – uden at gentage klub (kld) - eller præsidentnavne flere steder.

Præsident(pld, pNavn)

Præsidenten (pld) har sin egen tabel, så vi kan tilknytte én præsident (pld) til en klub (kld) via pld.

Det giver fleksibilitet, hvis præsidenten (pld) ændrer navn, eller skifter klub(kld).

Nationalitet(nld, nNavn)

Nationaliteter (nld) er også adskilt.

Både spillere (fld) og trænere (tld) kan have en nationalitet (nld), og ved at henvise via nld undgår vi at skrive landets navn igen og igen.

Træner(tld, tNavn, kld, nld)

Trænere (tld) er gemt i deres egen tabel.

De har tilknytning til en klub og en nationalitet via ID'er, så vi undgår at gentage klub (kld)- og landenavne (nNavn)

Det gør det også nemmere at finde fx alle trænere fra samme land (nld).

Position(pold, poNavn)

Positioner som "målmænd", "angriber" osv. er adskilt i en tabel.

Det sikrer ensartet navngivning og gør det nemt at administrere positionsdata ét sted.

Spiller Som(fld, pold)

En spiller kan have flere positioner, og der kan være flere der spiller i den samme position, derfor er dette en separat relation (mange-til-mange).

Hver række knytter én spiller til én position.

Det gør det muligt at finde alle spillere på en given position, eller alle positioner for en given spiller.

Første Normalform (1NF)

Alle tabeller har kun én værdi per celle.

Fx: SpillerSom har én spiller og én position pr. række – ikke en liste af positioner.

Der er ingen gentagne grupper.

Alle tabeller opfylder 1NF.

Anden Normalform (2NF)

Alle tabeller er i 1NF.

I tabeller med sammensatte nøgler (som SpillerSom) afhænger alle attributter af hele nøglen – her er der kun selve nøglen.

I tabeller med én primærnøgle (fx Fodboldspiller), afhænger alle attributter (fNavn, alld, ald, kld, nld) af hele nøglen (fld).

Alle tabeller opfylder 2NF.

Tredje Normalform (3NF)

Alle tabeller er i 2NF.

Der er ingen afhængigheder mellem ikke-nøgle-attributter.

Fx: I Fodboldspiller afhænger fNavn, alld, ald, kld og nld kun af fld – ikke af hinanden.

Alle tabeller opfylder 3NF.

4. Endelig skal databasen skabes vha. DDL (**logisk model**) og befolkes vha. DML.

I .txt filen 😊

5. DQL (også placeret nederst i .txt filen, nem at teste 😊)

-- 1) Ugt dansk talent: spillere, som er 21 eller under 21 år med dansk pas. Man kan bruge resultatet ved en transfer til klubben, hvor man filtrerer efter ungt dansk talent på 21 år eller under med dansk pas.

```
SELECT
    n.nNavn AS
    Nationalitet,
    f.fNavn AS Spiller,
    a.alNavn AS Alder,
    k.kNavn AS Klub,
    STRING_AGG(p.poNavn, ',' ORDER BY p.poNavn) AS positioner
FROM Fodboldspiller f
JOIN Alder a ON f.alld = a.alld
JOIN Nationalitet n ON f.nld = n.nld
JOIN Klub k ON f.kld = k.kld
JOIN SpillerSom ss ON f.fld = ss.fld
JOIN Position p ON ss.pold = p.pold
WHERE CAST(a.alNavn AS INTEGER) <= 21
    AND n.nNavn = 'Danmark'
GROUP BY f.fNavn, a.alNavn, k.kNavn
```

ORDER BY a.alNavn, f.fNavn;

-- 2) Klub-statistik: antal spillere og gennemsnitsalder pr. klub. Når man starter karrieremode i NFL, kan man bruge hver klubs gennemsnitsalder og antal spillere ved trænerkarrierestart til at træffe sit klubvalg.

```
SELECT
    k.knavn           AS Klub,
    COUNT(*)          AS Antal_spillere,
    ROUND(AVG(CAST(al.alnavn AS NUMERIC)), 2) AS Gennemsnitsalder
FROM fodboldspiller f
JOIN alder         al ON f.alid = al.alid
JOIN klub          k ON f.kid  = k.kid
GROUP BY k.knavn
ORDER BY antal_spillere DESC;
```

-- 3) Liste over fodboldspillernes agenter, hvis de har en, ellers markeret som "Ingen Agenter". Dermed kan man se, hvilke spillere der er repræsenteret af hvilke agentfirmaer, så hvis man vil transfere en spiller, kan man starte med at kontakte det pågældende agentfirma.

```
SELECT
    ag.aNavn          AS Agent,
    COUNT(*)          AS Antal_spillere,
    STRING_AGG(f.fNavn, ', ' ORDER BY f.fNavn) AS Spillere
FROM Fodboldspiller f
JOIN Agent ag ON f.alid = ag.alid
GROUP BY ag.aNavn
ORDER BY antal_spillere DESC;
```

Produktkrav

Du/I skal samle følgende i én mappe på GoogleDrev og dele linket under afleveringen i Lectio:

- Beskrivelse af problemområdet
- E/R-diagram (en tegning / powerpoint)
- Relationel model evt. inkl. overvejelser vedr. normalisering (hvis du/I har gjort dette)
- Databasen (.db-fil)
- Et tekstdokument (eks.TextEdit / Notepad) med din/jeres DDL og DML kode.
- 3 eksempler på DQL-kode, hvor du/I viser, hvad du kan og hvad din database kan bruges til (inkluder en beskrivelse af hvad dine *queries* gør). Se eksempel nedenfor:

Udregn gennemsnitsalderen på alle elever, der har bestået Thomas' informatikhold.

```
SELECT AVG(e.alder)
FROM elever e
JOIN underviser u ON e.id = u.elevId
JOIN lærer l ON u.lærerId = l.id
JOIN fag f ON u.fagId = f.id
WHERE l.navn = "Thomas"
AND f.navn = "Informatik"
AND u.karakter >= 2;
```

Vurderingsgrundlag

- Afspejler *E/R-diagrammet* problemområdet korrekt (eks. er alle elementerne medtaget)?
- Sikrer den *relationelle model* god *performance* og reducerer *redundans*?
- Virker databasen – kan alle data tilgås?
- Er *DDL* og *DML* korrekt implementeret?
- Hvor avanceret er *DQL*-eksemplerne (og fungerer de ifølge hensigten)?

Keep it simple. Forsøg at holde jer til en model med få entiteter og relationer, men hvis I laver en med "mange" og I formår at gøre det rigtigt, så er der selvfølgelig ekstra point for det.