NTNU
Institutt for informatikk
og e-læring

Introduksjon til datamodellering

Kjell Toft Hansen
IDRI1002 Informatikk 1: Databaser

Sammendrag

I denne artikkelen gis det en introduksjon i sentrale datamodelleringsbegreper. Innholdet er knyttet til lærebokas kapittel 6 *Datamodellering*.

Bakgrunn

ER (Entity-Relationship)-modellen ble introdusert i 1976 av Peter Pin-Shan Chen i artikkelen «The Entity-Relationship Model – Toward A Unified View of Data». Den består av en mengde ulike begreper for å beskrive strukturen til en database.

Med ER-modellen lager vi et begrepsskjema som gir som resultatet en strukturert grafisk framstilling av en «miniverden» (forretningsdomene). Styrken i en ER-modell ligger i arbeidet som foregår uavhengig av teknisk kompetanse.

Grunnprinsippene er skjemaet som bygges opp av *entitetstyper* (eng. entity types), *sammenhengstyper* (eng. relationships) og *attributter* (egenskaper).

Utgangspunkt for ER-analysen

Følgende spørsmål er utgangspunktet for ER-analysen: (1) Hvilke krav er bestemmende for databasen? (2) Hvilke formål har en med databasen? (3) Hvordan skal databasen brukes? (4) Hvilke entitetstyper (objekter fra virkeligheten) skal registreres i databasen? (5) Hva slags entiteter (data) skal registreres for hver entitetstype? (6) Hva slags sammenhenger er det mellom de ulike entitetene?

Punkt 4 – 6 besvares ved hjelp av ER-diagrammet.

Entitetstype

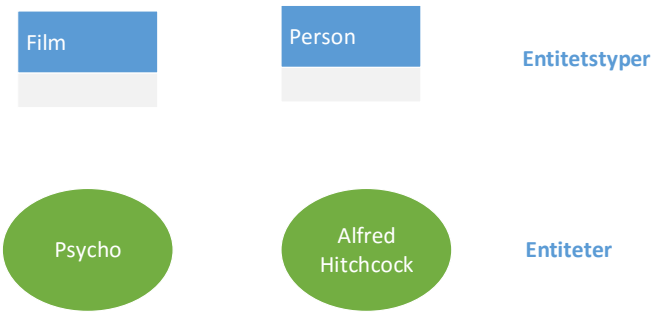
En *entitetstype* er en abstraksjon av virkeligheten. Det er enheter i den virkelige verden med visse fellestrekk, som regel en ting, et begrep eller en rolle.

En entitetstype er et substantiv i en problemtekst. Vi bruker entitetstyper til å «samle og skille» enheter fra hverandre. Vi skiller mellom *sterke* og *svake* entitetstyper. Det vanligste er sterke entitetstyper.

Figur 1. Ulike sterke entitetstyper

Entitet

En *entitet* er *én forekomst* (enkelttilfelle) av en entitetstype.



Figur 2. Entitetstyper med tilhørende entiteter

Attributter

Et *attributt* er en egenskap ved en entitetstype. Du finner attributter ved å analysere entitetene i entitetstypen og finne fram til felles og relevante egenskaper ved dem og hvilke data som skal lagres.

Figur 3. Entitetstype med attributter

Vi har ulike typer attributter: *nøkkelattributt* (identifikator, primærnøkkel), *én-verdi attributt*, *sammensatt attributt*, *flerverdi-attributt* og *avledet attributt*.

Nøkkelattributt

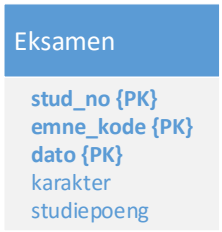
Et *nøkkelattributt* er et attributt som gir en entitet en *entydig* identifikasjon.

Figur 4: Entitetstypen Film med nøkkelattributtet *film_id*.

Ingen filmer har samme film_id.

Sammensatte nøkkelattributt

Et nøkkelattributt kan være sammensatt av flere attributter, men vi snakker fremdeles om ett nøkkelattributt. Et *sammensatt nøkkelattributt* bør ikke bestå av for mange attributter.



Figur 5. Entitetstype med et sammensatt nøkkelattributt

I Figur 5 har entitetstypen Eksamen et sammensatt nøkkelattributt bestående av attributtene stud_no, emne_kode og dato. Samme student kan gå opp til eksamen i samme emne flere ganger, men ikke på samme dato. Det er kombinasjonen av verdiene i de 3 attributtene som er entydig.

Kandidatnøkkel

Vi finner nøkkelattributtet ved å finne aktuelle *kandidatnøkler*. En kandidatnøkkel er et attributt som kunne ha vært et nøkkelattributt med samme egenskaper som et nøkkel-attributt.

Figur 6. Entitetstypen Film

I entitetstypen Film i Figur 6 er attributtene tittel, år, lengde, produsent mulige kandidatnøkler. Men ingen av disse attributtene vil ha en entydig verdi for hver entitet. Filmer kan ha samme tittel (Psycho fra 1960 og Psycho fra 1998). Flere filmer utgis i løpet av ett år (Psycho og The Little Shop of Horrors fra 1960). Flere filmer kan ha samme lengde (The Birds og Citizen Kane, 119 min.). Flere filmer kan ha samme produsent (Psycho og Citizen Kane, Universal Pictures). Derfor kan ingen av disse attributtene være nøkkelattributt alene og er derfor heller ingen kandidatnøkkel.

Surrogatnøkkel

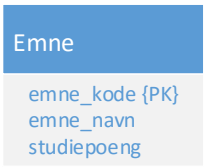
Hva med en kombinasjon av attributter? Tittel og år? Tittel, år og produsent? Tittel og lengde? Ingen av de mulige kandidatnøklene gir noen garanti for entydige verdier for hver entitet.

Løsningen blir å opprette attributtet film_id, en *surrogatnøkkel*. Film_id vil ha verdier som inneholder egne id-nummer for hver film, slik at verdien for hver film_id blir entydig.

Figur 7. Entitetstypen Film med surrogatnøkkel.

Én-verdi attributt

Én-verdi attributter kan kun ha én verdi (atomisk) for hver entitet.



Figur 8. Entitetstypen Emne med én-verdi attributter

Et emne kan for eksempel ha kun én emnekode, ett emnenavn og ett gitt antall studiepoeng.

Sammensatte attributt

Et *sammensatt attributt* består av flere attributter som naturlig hører sammen.

Figur 9. Entitetstypen Ansatt med et sammensatt attributt

Attributtet navn er sammensatt av attributtene etternavn og fornavn.

Flerverdi-attributt

Et *flerverdi-attributt* kan ha flere verdier for hver entitet.

Hvis det for hver kunde alltid er 2 epost adresser, er løsningen som er vist i Figur 10 grei.

Figur 10. Entitetstypen Kunde med flerverdi-attributtet epost

Hva hvis noen kunder ikke har epost adresse, kun én eller 3? Da er ikke løsningen i Figur 10 like grei. Det betyr at vi må vi finne en mer universell og fleksible løsning. En slik løsning er vist i Figur 11.

Figur 11. Optimal flerverdi-løsning

En entitet i entitetstypen Kunde kan ha ingen eller flere epost adresser.

Avledet attributt

Verdien til et *avledet attributt* avledes av verdien til ett eller flere andre attributter i samme entitetstype.

Figur 12. Entitetstype med avledet attributt

Verdien til attributtet varighet i entitetstypen Leasing_avtale blir avledet av forskjellen mellom verdiene i attributtene start og slutt.

Sammenhengstyper

En *sammenhengstype* kobler sammen entitetstyper i et ER-diagram slik at vi får uttrykt sammenhenger mellom entitetene i entitetstypene. En sammenhengstype designes ut fra en *handling* (et verb) i teksten (bor på, arbeider i, produserer, skriver på, etc.).

Sammenhengstyper er som regel *2-veis* (*todimensjonale, binære*) og kan ha egne attributter.

Hver sammenhengstype har et beskrivende navn som kalles *intensjon*.

Figur 13. Den binære sammenhengstypen «jobber i».

Multiplisitet

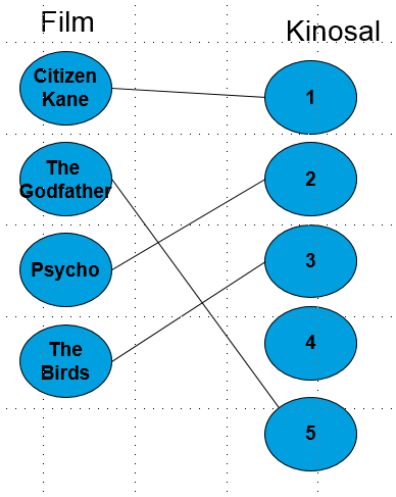
Multiplisiteten beskriver antall mulige sammenhenger, den minimale/maksimale forekomsten av koblinger mellom entiteter i en binær sammenhengstype.

Det er 3 former for multiplisitet: *én-til-én*, *én-til-mange* og *mange-til-mange*.

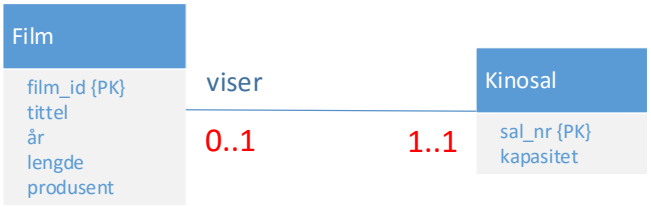
Én-til-én

En entitet av entitetstypen Film kan kun være koblet til minimum/maksimum én entitet av entitetstypen Kinosal. Én entitet av Kinosal kan være koblet til maksimum én entitet av Film eller ingen.

Denne multiplisiteten er illustrert som et *semantisk nett* i Figur 14 og som et ER-diagram i Figur 15.



Figur 14. Én-til-én sammenhengstype illustrert med et semantisk nett

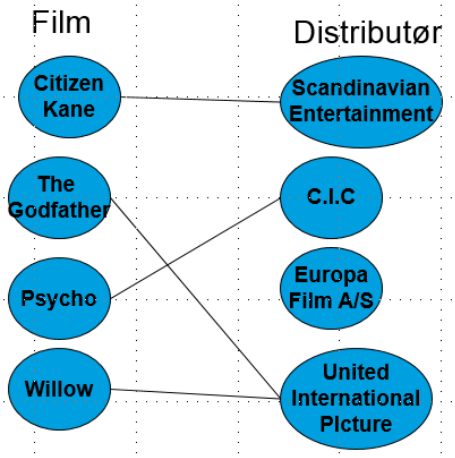


Figur 15. Én-til-én sammenhengstype illustrert som et ER-diagram med standard symboler

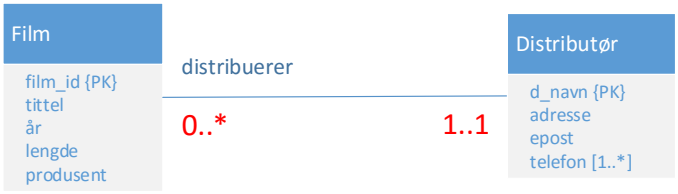
Én-til-mange

I en én-til-mange sammenhengsgtype kan hver entitet av entitetstypen Film kun være koblet til én entitet av entitetstypen Distributør, mens hver entitet av entitetstypen Distributør kan være koblet til flere entiteter av entitetstypen Film. Videre: én film kan bare distribueres av én distributør, mens én distributør kan distribuere ingen eller flere filmer.

Dette er illustrert som et semantisk nett i Figur 16 og som et ER-diagram i Figur 17.



Figur 16. Én-til-mange representert som et semantisk nett

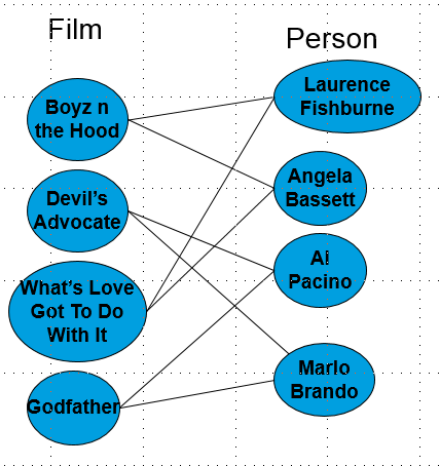


Figur 17. Én-til-mange sammenhengstype representert som et ER-diagram med standard symboler

Mange-til-mange

I en mange-til-mange sammenhengstype kan hver entitet i entitetstypen Person være koblet til flere entiteter i entitetstypen Film og motsatt. Videre: Én person kan spille i minst én film og maksimum mange. En film kan ha minst én person (skuespiller) og maksimum mange.

Denne sammenhengstypen er illustrert med et semantisk nett i Figur 18 og som et ER-diagram i Figur 18.

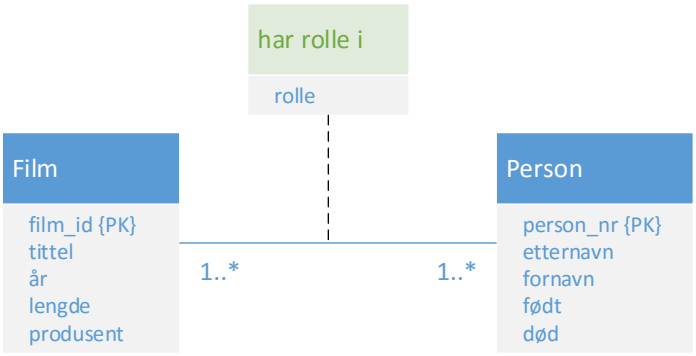


Figur 18. Mange-til-mange representert som et semantisk nett

Figur 19. Mange-til-mange sammenhengstype representert som et ER-diagram med standard symboler

Sammenhengstyper med attributt

Problem: En person innehar rollefigurer i flere filmer, en film har flere rollefigurer; attributtet «rolle» kan verken knyttes til entitetstypen Film eller Person. Løsningen blir å knytte den til sammenhengstypen mellom entitetstypene som vist i Figur 20.

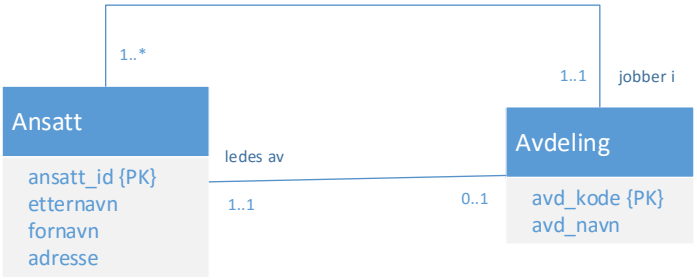


Figur 20. Sammenhengstype med ett attributt

Fler-rolle-problematikk

Vi kan ha flere sammenhengstyper som kobler sammen to entitetstyper. Problem: en ansatt jobber i en avdeling, en avdeling har flere ansatte, en avdeling har en avdelingsleder og en ansatt leder en avdeling.

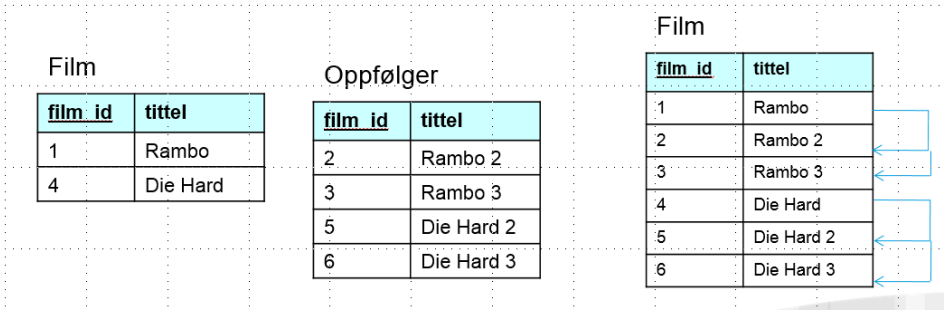
Vi løser dette som vist i Figur 21.



Figur 21. To entitetstyper med to sammenhengstyper

Rekursive sammenhengstyper

Det kan være sammenhenger mellom entiteter i samme entitetstype. Eksempel på dette kan være entitetstypen Film. En film har mange «oppfølgere». En oppfølger er en oppfølger til én original film. Dette er illustrert som tabeller i Figur 22.



Figur 22. Filmer med oppfølgingsfilmer

Når vi designer ER-diagrammet for «film med oppfølgere», får vi ER-diagrammet som vist i Figur 23 og som tabell i Figur 24.

Figur 23. Entitetstypen Film med rekursjon (sammenhengstype til seg selv)

Film

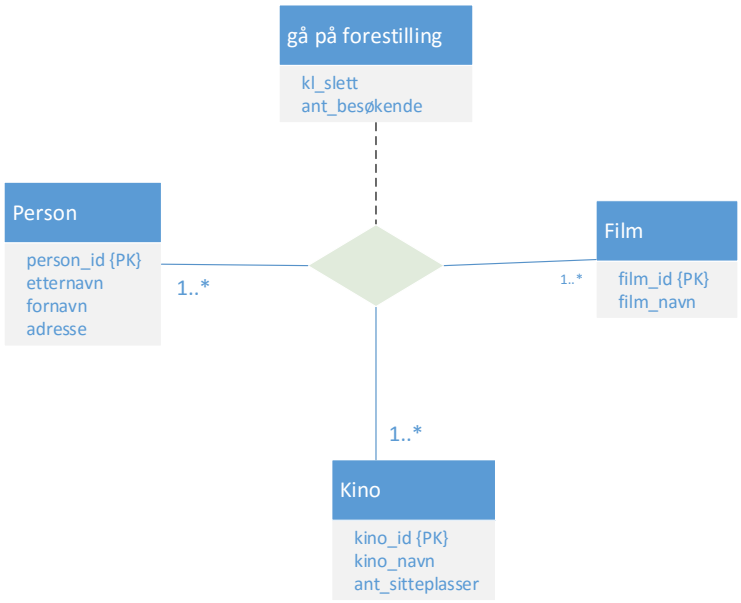
film_id	tittel	film_id_org*
1	Rambo	NULL
2	Rambo 2	1
3	Rambo 3	1
4	Die Hard	NULL
5	Die Hard 2	4
6	Die Hard 3	4

Figur 24. Entitetstypen Film oversatt til tabell (* betyr fremmednøkkel – forklares senere)

Multiveis sammenhengstyper

Normalt er binære sammenhengstyper tilstrekkelig. I noen tilfeller må tre – *trinær* – eller flere entitetstyper kobles sammen for å uttrykke et mer komplekst sammenhengstype-forhold.

Hvis vi skal lagre informasjon om forestillinger på en kino må vi designe sammenhengen mellom hvem som går på forestillingen, hvilken film som blir vist og på hvilken kino forestillingen blir vist. Dette kan da designes med bruk av *multiveis sammenhengstyper*.



Figur 25. Multiveis sammenhengstype

Avhengigheter i sammenhengstyper

Avhengigheten avgjør om *eksistensen* til en entitet avhenger av sammenhengen til en annen entitet i en tilkoblet entitetstype. Når vi har *total avhengighet* mellom entiteter i to tilkoblete entitetstyper, har vi *eksistensavhengighet*. I andre tilfeller er avhengigheten *delvis*.

Eksistensavhengighet

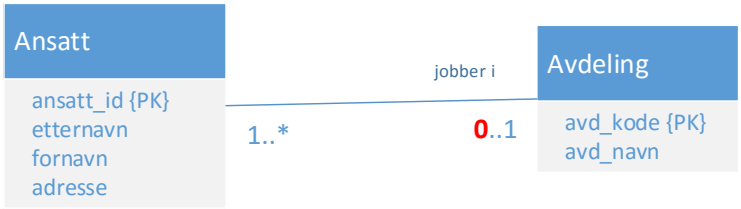
Eksistensavhengigheten finner vi alltid på én-siden i en én-til-mange sammenhengstype.

Eksempel: en ansatt må ha minst én postadresse – for hver ansatt som registreres må det registreres minst ett postnummer – entitetstypen Ansatt er eksistensavhengig av entitetstypen Sted. Dette er vist i Figur 26.

Figur 26. En ansatt er eksistensavhengig av en postadresse

Delvis avhengighet

Delvis avhengighet finner vi også på én-siden i en én-til-mange sammenhengstype. Eksempel: en ansatt jobber i én avdeling, men det kan være ansatte som ikke er knyttet til noen avdeling. Vi kan m.a.o. registrere ansatte uavhengig om de tilhører en avdeling eller ikke. Da blir en ansatt delvis avhengig av en avdeling. Dette er vist i Figur 27.



Figur 27. En ansatt er delvis avhengig av en avdeling

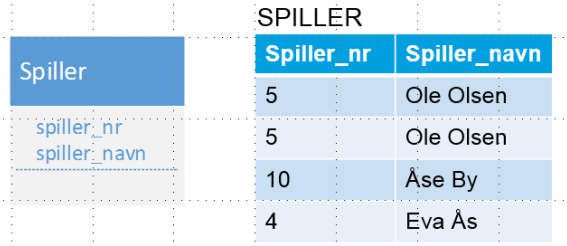
Sterke entitetstyper

En *sterk entitetstype* har et nøkkelattributt (identifikator) som entydig identifiserer hver entitet. Entitetstypen Film i Figur 28 er et eksempel på en sterk entitetstype.

Figur 28. Den sterke entitetstypen Film.

Svake entitetstyper

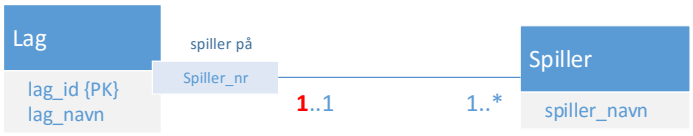
En *svak entitetstype* har ikke noe nøkkelattributt (identifikator) som entydig identifiserer hver entitet i entitetstypen. Eksempel: på et fotballag har hver spiller et spillernummer. Alle fotballag bruker samme nummerering. Det kan være spillere på forskjellige lag som har samme navn og samme nummer. Dette er illustrert i Figur 29.



Figur 29. Spiller illustrert som entitetstype og tabell

Identitetsavhengighet

En svak entitetstype kan ikke eksistere uten å være identifisert med en *eierentitetstype*. For å identifisere en entitet i entitetstypen Spiller, må vi inkludere attributtet spiller_nr i eierentitetstypen Lag. Dette er vist i Figur 30.



Figur 30. Spiller er identitetsavhengig av Lag

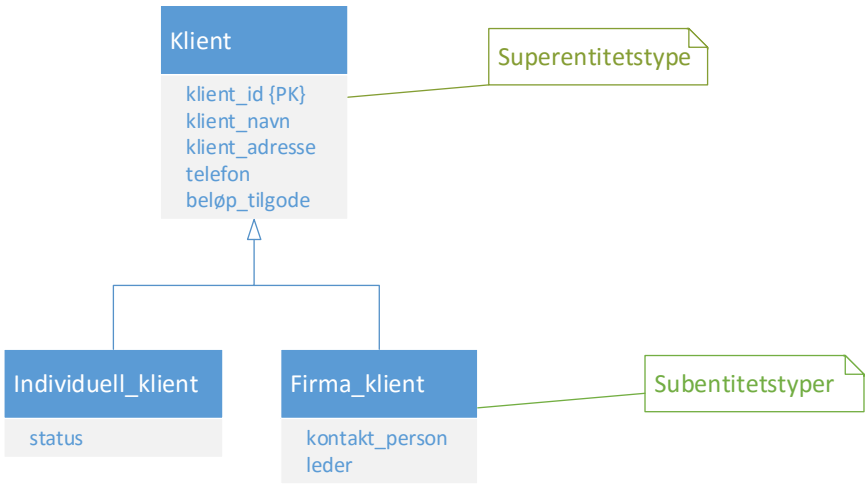
Entiteter i entitetstypen Spiller er *identitetsavhengige* av entiteter i entitetstypen Lag. Derfor vil alltid minimumsmultiplisiteten på én-siden (eiersiden) være 1.

EER (Enhanced Entity Relationship)-modell

Dette er en utvidelse av Chen sin opprinnelige ER-modell med objektorienterte prinsipper. Utvidelsen blir brukt for å designe et nøyaktigere databaseskjema og reflekterer data-egenskaper og begrensninger (skranker) mer presist.

Subtyping

Ofte har en entitetstype noen entiteter med attributter som ikke alle de andre entitetene i entitetstypen har. Vi kan da definere spesialtilfelle-entitetstyper, såkalte *subentitetstyper*. Subentietstypens entiteter arver alle superentitetens attributter. Vi får et spesialtilfelle av sammenhengstypen én-til-én. Dette er vist i Figur 31.

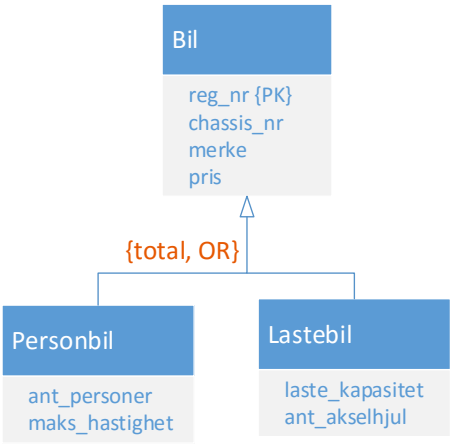


Figur 31. ER-diagram med subtyping

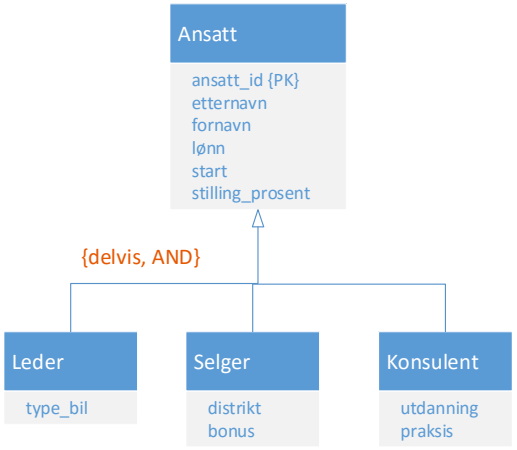
En subentitetstype kan også ha egne sammenhengstyper.

Subtyping har følgende 4 regler: *disjunkt* (OR) – en entitet kan kun tilhøre én subentitetstype, *overlappende* (AND) – en entitet kan tilhøre mer enn én subentitstype, *partiell (delvis) deltakelse* – en entitet trenger ikke å tilhøre noen subentitetstyper og *total deltakelse* – en entitet må tilhøre minst én subentitetstype.

Ut fra disse reglene får vi da følgende 4 typer spesialiseringer: *disjunkt og total*, *disjunkt og partiell*, *overlappende og total* og *overlappende og partiell*. To av disse er vist i Figur 32 og 33.



Figur 32. Total og disjunkt subtyping



Figur 33. Delvis og overlappende subtyping

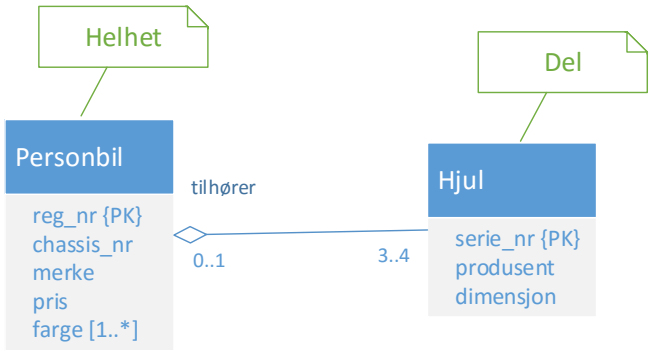
Figur 32 tolkes slik: *en bil er enten en person- eller lastebil, men ikke begge deler. En bil er ikke bare en bil, men må være enten en person- eller lastebil.*

Figur 33 tolkes slik: *en ansatt kan være både leder, selger og konsulent. En ansatt kan også bare være en generell ansatt og ikke noe av subtypene.*

Aggregering

Aggregering er et konseptuelt begrep. Den beskriver en ting og dens deler. Aggregeringen beskriver forholdet mellom det «hele og deler av helheten». Den skiller hvem av entitetstypene som organisasjonsmessig er overordnet.

Konseptuelt beskrives en aggregering med en åpen diamant på enden av sammenhengstypen som er helheten. Dette er vist i Figur 34.



Figur 34. Aggregering vist med standard symboler

Et hjul er en del av en personbil eller bare et hjul uten noen tilknytning. En personbil består av fra 3 til 4 hjul. Hvis personbilen «skrapes» (slettes), kan hjulene «leve» videre som uavhengige hjul.

Komposisjon

En *komposisjon* er en sterk aggregering. Delene er strengt koblet til den utøvende helheten.

Konseptuelt beskrives en komposisjon med en *fylt diamant* på helhet-siden av sammenhengstypen. Dette forteller at entiteter tilhørende entitetstypen på diamantsiden har, eller eier entiteter av entitetstypen på del-siden som komponenter.

Opprettelsen og eksistensen av de eide entitetene følger eier-entitetens eksistens. Derfor brukes en komposisjon bare når det har konsekvenser ved endringer av databasetilstanden når brukeren utfører datamanipuleringskommandoene INSERT, UPDATE, DELETE og CREATE.

Figur 35. Komposisjon vist med standard symboler

Sletter vi katalogen, vil produktene heller ikke lenger være relevante.

Designprinsipp

Når vi designer et ER-diagram må vi forsøke å unngå *redundans* (dobbeltlagring). Vi bør heller ikke bruke entitetstyper når attributter gjør nytten. Svake entitetstyper er også noe som bør unngås og det samme gjelder overforbruk av subtyping.

Unngå redundans

Redundans oppstår når vi sier det samme på to eller flere måter. Dette sløser med plass og oppmuntrer til inkonsistens: to entiteter med samme verdier kan bli *inkonsistente* om vi endrer én, men glemmer å endre den andre relaterte versjonen.

Når vi gir navnet på læreren ett sted (i entitetstypen Lærer) og én gang, er dette «god design» (Figur 36).

Figur 36. Godt ER-design uten redundans

Vi får et «dårlig design» når vi repeterer navnet på læreren én gang for hvert emne. Og personens epost adresse og telefon forsvinner fra databasen om vedkommende midlertidig ikke underviser. Dette er vist i Figur 37.

Figur 37. Dårlig ER-design med redundans

Entitetstyper versus attributter

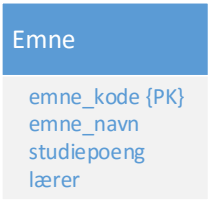
En entitetstype skal tilfredsstille «navnet på noe» og det *skal ha minst ett attributt i tillegg til nøkkelattributtet*.

Vi kan illustrere dette med et eksempel fra en «undervisningssituasjon». En lærer underviser i emner og bør være egen entitetstype pga. av ikke-nøkkel-attributtene, etternavn, fornavn, adresse, epost og telefon. Emne fortjener å være egen entitetstype fordi det er mange-siden i en én-til-mange sammenhengstype. Dermed unngår vi en kopi av læreropplysninger for hvert emne.

Figur 38. Riktig valg av entitetstyper

Hvis vi ikke hadde ekstrainformasjonen om en lærer foruten navnet ville det ikke ha vært nødvendig å opprette en egen entitetstype for lærer fordi vi kun lagrer verdier for et lærernavn.

Merk! Forutsetningen, som må være tilstede, er én lærer per emne.



Figur 39. Attributtet lærer istedenfor entitetstypen Lærer

Unngå overforbruk av svake entitetstyper

I praksis blir det laget nøkkelattributt for de fleste entitetstyper som for eksempel personid, ansnr, studid etc.

Når trenger vi da svake entitetstyper? Jo, når det ikke finnes en global autoritet som kan generere entydige id-er. Dette har vi illustrert med eksemplet med spillernummeret til en fotballspiller (Figur 29 og 30).

Unngå overforbruk av subtyping

Subtyping må ha sin begrunnelse i den virkelige verden og må kunne gi mening. For all del må subtypene ha noe felles.

Det kan også være feil å bruke subtyping. Bildekk er ikke en subtype av bil, men en del.

Litteraturliste

Connolly, Thomas og Begg, Carolyn 2010. *Database systems. A Practical Approach to Design, Implementation, and Management*. Fifth Edition. Pearson.
Elmasri, Ramez og Navathe, Shamkant B. 2011. *Database Systems. Models, Languages, Design, and Application Programming*. Sixth Edition. Pearson.