NTNU
Kunnskap for en bedre verden

Øving 1: Relasjonsmodellen, del 1 (obligatorisk)

Else Lervik (redigert av Tore Mallaug), NTNU
Innleveringsfrist: se Blackboard
Tidligste godkjenning: datoer blir annonsert

Løsningsforslag legges ut i etterkant.

Alle obligatoriske øvinger må være godkjente for å få karakter i emnet.

Flervalgsoppgave

I Blackboard finner du testen ***Intro_til_databaser***. Denne må være godkjent i tillegg til oppgavene i dette dokumentet.

Problemstilling:
Database for boligbyggerlaget Bygg & Bo.

Du skal lage en database for boligbyggerlaget Bygg & Bo (BB). BB administrerer byggeaktiviteter og etablerte borettslag. Følgende data om borettslagene er nødvendig for administrasjonen, og skal lagres i en egen database:

BB eier mange borettslag. Hvert borettslag er bestemt ved navn, adresse, antall hus og/eller blokkenheter og etableringsår.

BB har et medlemsregister over samtlige andelseiere i borettslagene. Av disse utgjør leietakerne i ulike borettslag ca. 75 %. (Det er mulig å være medlem i et borettslag uten å eie en leilighet der, for eksempel kan det være folk som ønsker å flytte dit, men som venter på at en leilighet skal bli ledig for salg.)

Hvert borettslag er oppdelt i hus/blokker med egne adresser, og i hvert hus/blokk finnes det et antall leiligheter. Hus/blokk beskrives ved antall etasjer og antall leiligheter. Leilighetene beskrives vha av antall rom og antall m^2, etasje, og et leilighetsnummer.

En andelseier kan bare eie én leilighet. En leilighet eies av en andelseier, men i perioder kan den være uten eier.

Innledende oppgave a: Sett opp mulige tabeller

Sett opp noen mulige tabeller med fornuftige attributtnavn. Et minimum må være å opprette tabeller for henholdsvis borettslag, bygninger, andelseiere og leiligheter. Du bør tenke litt igjennom hvilke data som skal lagres før tabellene lages. Unngå å lagre samme data i flere tabeller for å minske faren for duplisering av data. Unngå også å lagre data som kan avledes (regnes ut) av andre data.

Eksempel på en tabell på relasjonell form:

```
ansatt(ansatt_nr, fornavn, etternavn, kontor_telefon, fodselsaar)
```

Innledende oppgave b: Entitetsintegritet (primærnøkkel)

For hver tabell du satte opp i oppgave a skal du nå finne mulige *kandidatnøkler*. For hver kandidatnøkkel bør du begrunne hvorfor du mener at dette er en nøkkel. Velg deretter en passende *primærnøkkel* for hver tabell.

Eksempel: I tabellen Ansatt foran er det mange mulige kandidatnøkler. Men hvilke av dem som er entydige er situasjonsbestemt. Dersom dette er en liten bedrift med få ansatte, er det mulig at attributtet etternavn eventuelt kombinasjonen av fornavn og etternavn er entydig, og derfor en kandidatnøkkel. Men dette er neppe tilfelle dersom det er mange ansatte. Flere kan hete det samme. Kontor_telefon er kanskje også en kandidatnøkkel, hvis hver ansatt har hvert sitt kontor. Men alt dette er så usikkert, at jeg har funnet det best å lage et eget attributt nummer, ansatt_nr, som kandidatnøkkel og som jeg har valgt til primærnøkkel. Det er vanlig å sette en strek under pimærnøkkelen:

```
ansatt(ansatt_nr, fornavn, etternavn, kontor_telefon, fodselsaar)
```

HUSK! Det er mulig å ha sammensatte nøkler.

Innledende oppgave c: Referanseintegritet (fremmednøkkel)

Inkluder *referanseintegritet* (fremmednøkler) i tabellene dine, slik at du får fram *en-til-en* og *en-til-mange*-sammenhengstyper mellom Borettslag, Bygning, Leilighet og Andelseier.

Oppgave som skal vises fram for godkjenning

a) Lag en figur svarende til klassediagram uten operasjoner. Sett på sammenhengene mellom klassene (en-til-mange, mange-til-mange).

b) Oversett til relasjonsmodellen.

c) Opprett en database og kjør CREATE TABLE setninger der du definerer primær- og fremmednøkler slik du er kommet fram til i de innledende oppgavene.

d) Lag INSERT-setninger der du legger inn noen gyldige data i tabellene. Lag også INSERT-setninger som gir brudd på entitets- og referanseintegritetsreglene.

e) Fra forelesningene har vi at en fremmednøkkel kan være NULL. Er det fornuftig å tillate dette for de fremmednøklene du har kommet fram til her? Kan primærnøkler være NULL?

f) Når vi skal implementere referanseintegriteten (fremmednøkkel) er det mulig å sette ON DELETE/UPDATE CASCADE (se under for nærmere forklaring). Vurder konsekvensen av dette for hver enkelt fremmednøkkel.

ON DELETE/UPDATE CASCADE

Anta at vi har en tabell Bok og en tabell Forlag, med en-til-mange-sammenheng fra forlag til bok. Et forlag gir altså ut mange bøker, mens en bok er gitt ut på eksakt ett forlag:

Ved oversetting til relasjonsmodellen får vi:

```
bok(bok_id, tittel, forlag_id*) // OBS! fremmednøkkel pga. en-til-mange-sammenhengen
forlag(forlag_id, forlagnavn)
```

At forlag_id er fremmednøkkel i tabellen Bok angir dere for eksempel slik i SQL:

```
ALTER TABLE bok ADD CONSTRAINT bok_fk FOREIGN KEY(forlag_id)
REFERENCES forlag;
```

Her defineres referanseintegritetsregelen bok_fk, med forlag_id som fremmednøkkel og med referanse til tabellen Forlag hvor forlag_id er primærnøkkel. Her kan vi også legge til ON DELETE CASCADE:

```
ALTER TABLE bok
ADD CONSTRAINT bok_fk FOREIGN KEY(forlag_id) REFERENCES forlag
(forlag_id) ON DELETE CASCADE;
```

Dette betyr at dersom et forlag_id fjernes (legges ned eller går konkurs) fra tabellen Forlag (mor-tabellen), så vil alle bøkene som er utgitt (registrert) på samme forlag også slettes fra tabellen Bok. Dette er ikke ønskelig.

Dersom vi *ikke* har med ON DELETE CASCADE vil vi ikke få lov å fjerne et forlag fra tabellen Forlag så fremt forlaget har gitt ut minst én bok.

Tilsvarende gjelder ON UPDATE CASCADE, men da er det oppdateringer som "kaskades".

Ekstraoppgave

Prøv ut ON DELETE CASCADE og ON UPDATE CASCADE.