*Løsningsforslag*

# IDATT2103 Databaser

# Øving: Relasjonsmodellen, del 1

*Else Lervik*

---

## ***Flervalgsoppgave, se løsning i BlackBoard***

### *Innledende oppgave a: Sett opp mulige tabeller*

```
Borettslag(bolag_navn, bolag_adr, etabl_aar)
Bygning(bygn_adr, ant_etasjer)
Leilighet (leil_nr, ant_rom, ant_kvm, etasje)
Andelseier (and_eier_nr, fornavn, etternavn, telefon, ansiennitet)
```

Alle tabell- og kolonnenavn skrives sammenhengende. SQL er ikke case-sensitivt, vi bruker derfor understrekingstegn der vi i, for eksempel Java, heller ville brukt stor bokstav (databaser: *ant_etasjer*, programmering: *antEtasjer*). Unngå å bruke æ, ø og å i navn på tabeller og kolonner.

Merk at antall bygninger og antall leiligheter ikke er tatt med i attributtlistene over. Dette er såkalte *avledete attributter*, det vil si at vi kan regne dem ut.

### *Innledende oppgave b: Entitetsintegritet (primærnøkler)*

En enkelt kolonne eller en kombinasjon av kolonner som entydig identifiserer en rad i tabellen kalles kandidatnøkkel. Kandidatnøkler markeres her med *kursiv*:

```
Borettslag(bolag_navn, bolag_adr, etabl_aar)
Bygning(bygn_adr, ant_etasjer)
Leilighet (leil_nr, ant_rom, ant_kvm, etasje)
Andelseier (and_eier_nr, fornavn, etternavn, telefon, ansiennitet)
```

Borettslag: Borettslagsnavnet og –adressen vil hver for seg kunne identifisere borettslaget. Vi har to kandidatnøkler, velger å bruke navnet som primærnøkkel.

Bygning: Vi kan bruke adressen, men velger her å innføre et løpenummer som primærnøkkel i stedet. Det kunne vi også gjort for borettslag.

Leilighet: Her bruker vi leilighetsnummeret som primærnøkkel. Vi antar at dette er entydig for alle leilighetene i hele Bygg&Bo. Eventuelt kunne det vært entydig for hver bygning eller hvert borettslag (jmf. begrepet svak entitetstype, kapittel 6).

Andelseier: Attributtet and_eier_nr brukes som primærnøkkel.

Tabellene ser nå slik ut, med primærnøkkelen understreket:

```
Borettslag(bolag_navn, bolag_adr, etabl_aar)
Bygning(bygn_id, bygn_adr, ant_etasjer)
Leilighet (leil_nr, ant_rom, ant_kvm, etasje)
Andelseier (and_eier_nr, fornavn, etternavn, telefon, ansiennitet)
```

#### Konklusjon, primærnøkler

En bør alltid vurdere om det er fornuftig å innføre et løpenummer (MySQL: AUTO_INCREMENT, JavaDB: IDENTITY, Oracle: SQL-SEQUENCE) som primærnøkkel.

Spesielt ved oppdateringer kan det skape problemer dersom man bruker en informasjonsbærende primærnøkkel. En primærnøkkel blir ofte fremmednøkler i andre tabeller, og en oppdatering av primærnøkkelen (for eksempel endring av en adresse) må derfor også medføre at de tilsvarende fremmednøklene endres. Dette er ikke alltid trivielt, se oppgaven om ON UPDATE CASCADE til

slutt i dette løsningsforslaget. På den annen side vil fremmednøkler med informasjon ofte kunne gi en klient den informasjon som trengs, uten at denne trenger å slå opp i den tabellen som fremmednøkkelen refererer til.

Altså, som så ofte ellers i datamodelleringsverdenen, en må vurdere behovet for enkle og raske søk opp mot behovet for enkle og raske oppdateringer.

Vær også forsiktig med å bruke det 11-sifrede fødselsnummeret som primærnøkkel i person-tabeller. Kan man garantere at det er aktuelt å kreve dette nummeret for alle nåværende og framtidige anvendelser? Og hva med utlendinger?

Bruk SQL-kravet UNIQUE for informasjonsbærende attributter som skal være entydige og som ikke er primærnøkler.

### *Innledende oppgave c: Referanseintegritet (fremmednøkkel)*

Tabellene er utvidet med referanseintegritet (stjerne) for å kunne kople opplysninger mellom tabellene.

```
Borettslag(bolag_navn, bolag_adr, etabl_aar)
Bygning(bygn_id, bygn_adr, ant_etasjer, bolag_navn*)
Leilighet (leil_nr, ant_rom, ant_kvm, etasje, bygn_id*, and_eier_nr*)
Andelseier (and_eier_nr, fornavn, etternavn, telefon, ansiennitet, bolag_navn*)
```

Vi har flere en-til-mange-sammenhenger, og fremmednøklene havner på mange-siden i forholdet:

- Et borettslag har mange bygninger (en-til-mange) - derfor blir bolag_navn fremmednøkkel i tabellen Bygning.
- En bygning har flere leiligheter (en-til-mange) - derfor blir bygn_id fremmednøkkel i tabellen Leilighet.
- Et borettslag har mange andelseiere (en-til-mange) - derfor blir bolag_navn fremmednøkkel i tabellen Andelseier.

Vi har en-til-en-sammenheng mellom Leilighet og Andelseier. På hvilken side skal vi plassere fremmednøkkelen?

En leilighet må ha eksakt én andelseier (en-til-en), men en andelseier trenger ikke å ha en leilighet - derfor blir andelseiernummer fremmednøkkel i tabellen Leilighet. Dette er antakelig den beste løsningen. Ved å sette UNIQUE-krav på denne fremmednøkkelen sikrer en at en andelseier, eier kun én leilighet.

Alternativt kan leil_nr bli fremmednøkkel i tabellen Andelseier. Andelseiere som ikke har leilighet vil ikke ha verdi for denne kolonnen.

Nå er tabellene knyttet sammen ved hjelp av fremmednøkler, og dermed kan vi kople informasjonen fra tabellene.

## Oppgavene som skulle vises fram for godkjenning

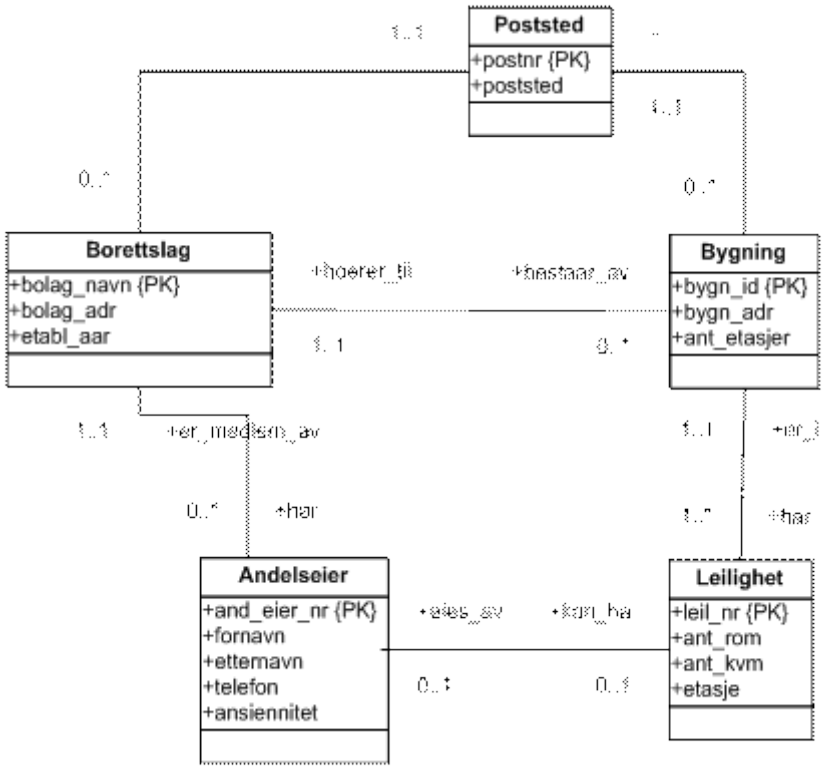
*Oppgave a) – UML-diagram (ER-modell)*

Se figur neste side.

Vi ser at vi har tatt med enitetstypen Poststed. Denne er ofte hendig i forbindelse med adresser. (Husk bare at utenlandske adresse må behandles spesielt.) Vi får dermed med postnr som fremmednøkkel i tabellene Borettslag og Bygning:

```
PoststedOv1(postnr, poststed)    // kaller relasjonen PoststedOv1 for å skille den
fra tilsv relasjoner i andre øvinger/eksempler
Borettslag(bolag_navn, bolag_adr, etabl_aar, postnr*)
Bygning(bygn_id, bygn_adr, ant_etasjer, bolag_navn*, postnr*)
```

```
Leilighet (leil_nr, ant_rom, ant_kvm, etasje, bygn_id*, and_eier_nr*)
Andelseier (and_eier_nr, fornavn, etternavn, telefon, ansiennitet, bolag_navn*)
```




*Oppgave b) – CREATE TABLE*

Vi må forholde oss til datatyper, og hvert eneste databasesystem har sin egen mengde lovlige datatyper, og det er ikke sikkert de har støtte for alle datatypene som inngår i ISO SQL-standarden. Men vi prøver alltid først en av dem, bare hvis det ikke går, tillater vi oss å bruke en proprietær type. Her er en liste over de datatypene du antakelig har mest brukt for:

Tekststrenger: CHAR – fast lengde, VARCHAR – variabel lengde, men maksverdi gitt
Tall: INTEGER, SMALLINT, REAL
Tidspunkter: DATE (kommer tilbake til denne i SQL-delen av faget)

Merk spesielt at BOOLEAN ikke fins, verken i Oracle eller Java DB. Men du kan bruke den i mySQL, da er den ekvivalent med en integer på 1 byte. Alle verdier forskjellig fra 0 betraktes som TRUE. Se http://troels.arvin.dk/db/rdbms/#data_types-boolean

Koden nedenfor er skrevet for mySQL. For å kjøre denne i andre systemer må genereringen av løpenummer endres.

Scriptet for å lage tabellene ser slik ut:

```
-- DROP-setninger i tilfelle vi må kjøre scriptet på nytt.
-- Referanseintegriteten krever at tabellene slettes i en bestemt rekkefølge.

DROP TABLE IF EXISTS leilighet;
DROP TABLE IF EXISTS andelseier;
DROP TABLE IF EXISTS bygning;
DROP TABLE IF EXISTS borettslag;
DROP TABLE IF EXISTS poststed;

-- Lager tabellene, legger inn NOT NULL- og UNIQUE-krav der det er naturlig
-- Vær forsiktig med å legge inn slike krav, det er vanskelig å forandre
```

```
-- dem i ettertid.

CREATE TABLE poststed(
  postnr SMALLINT PRIMARY KEY,
  poststed VARCHAR(20) NOT NULL);

CREATE TABLE borettslag(
  bolag_nr INTEGER NOT NULL AUTO_INCREMENT,
  bolag_navn VARCHAR(20) NOT NULL UNIQUE,
  bolag_adr VARCHAR(40) NOT NULL UNIQUE,
  etabl_aar SMALLINT NOT NULL,
  postnr SMALLINT NOT NULL,
  PRIMARY KEY(bolag_nr));

CREATE TABLE bygning(
  bygn_id INTEGER NOT NULL AUTO_INCREMENT,
  bygn_adr VARCHAR(40) NOT NULL,
  ant_etasjer INTEGER  DEFAULT 1,
  bolag_navn VARCHAR(20) NOT NULL,
  postnr SMALLINT NOT NULL,
  PRIMARY KEY (bygn_id));

CREATE TABLE leilighet(
  leil_nr INTEGER NOT NULL AUTO_INCREMENT,
  ant_rom SMALLINT NOT NULL,
  ant_kvm REAL NOT NULL,
  etasje SMALLINT DEFAULT 1,
  bygn_id INTEGER NOT NULL,
  and_eier_nr INTEGER NOT NULL UNIQUE,
  PRIMARY KEY (leil_nr));

CREATE TABLE andelseier(
  and_eier_nr INTEGER NOT NULL AUTO_INCREMENT,
  fornavn VARCHAR(30) NOT NULL,
  etternavn VARCHAR(30) NOT NULL,
  telefon CHAR(15),
  ansiennitet SMALLINT,
  bolag_navn VARCHAR(20) NOT NULL,
  PRIMARY KEY (and_eier_nr));

-- Fremmednøkler

ALTER TABLE borettslag
  ADD CONSTRAINT borettslag_fk1 FOREIGN KEY(postnr)
  REFERENCES poststed(postnr);

ALTER TABLE bygning
  ADD CONSTRAINT bygning_fk1 FOREIGN KEY(postnr)
  REFERENCES poststed(postnr);

ALTER TABLE bygning
  ADD CONSTRAINT bygning_fk2 FOREIGN KEY(bolag_navn)
  REFERENCES borettslag(bolag_navn);

ALTER TABLE leilighet
  ADD CONSTRAINT leilighet_fk1 FOREIGN KEY(bygn_id)
  REFERENCES bygning(bygn_id);

ALTER TABLE leilighet
  ADD CONSTRAINT leilighet_fk2 FOREIGN KEY(and_eier_nr)
  REFERENCES andelseier(and_eier_nr);

ALTER TABLE andelseier
  ADD CONSTRAINT andelseier_fk2 FOREIGN KEY(bolag_navn)
  REFERENCES borettslag(bolag_navn);
```

### *Oppgave c) – INSERT*

```
-- Legger inn gyldige data

INSERT INTO poststed VALUES(2020, 'Skedsmokorset');
INSERT INTO poststed VALUES(6408, 'Aureosen');
INSERT INTO poststed VALUES(7033, 'Trondheim');
INSERT INTO poststed VALUES(7020, 'Trondheim');

INSERT INTO borettslag VALUES(DEFAULT, 'Tertitten', 'Åsveien 100', 1980, 7020);
```

```
INSERT INTO borettslag VALUES(DEFAULT, 'Sisiken', 'Brurød', 1990, 7033);
INSERT INTO borettslag VALUES(DEFAULT, 'Lerken', 'Storgt 5', 2000, 6408);

INSERT INTO andelseier VALUES(DEFAULT, 'Even', 'Trulsbo', '56667743', 3, 'Tertitten');
INSERT INTO andelseier VALUES(DEFAULT, 'Anna', 'Olsen', '45674588', 10, 'Tertitten');
INSERT INTO andelseier VALUES(DEFAULT, 'Ingrid', 'Olsen', '45785388', 8, 'Tertitten');
INSERT INTO andelseier VALUES(DEFAULT, 'Arne', 'Torp', '78565388', 7, 'Tertitten');
INSERT INTO andelseier VALUES(DEFAULT, 'Arne', 'Martinsen', '78555388', 4, 'Sisiken');

INSERT INTO bygning VALUES(DEFAULT, 'Åsveien 100a', 3, 'Tertitten', 7020);
INSERT INTO bygning VALUES(DEFAULT, 'Åsveien 100b', 3, 'Tertitten', 7020);
INSERT INTO bygning VALUES(DEFAULT, 'Åsveien 100c', 6, 'Tertitten', 7020);
INSERT INTO bygning VALUES(DEFAULT, 'Storgt 10', 2, 'Sisiken', 7020);

-- bruker defaultverdien for antall etasjer, da er antall felter færre enn alle, og de må
oppgis eksplisitt:
INSERT INTO bygning(bygn_id, bygn_adr, bolag_navn, postnr)
            VALUES(DEFAULT, 'Åsveien 100', 'Tertitten', 7020);

INSERT INTO leilighet VALUES(DEFAULT, 5, 110, 3, 1, 1);
INSERT INTO leilighet VALUES(DEFAULT, 5, 110, 3, 1, 2);
INSERT INTO leilighet VALUES(DEFAULT, 2, 50, 1, 3, 3);

-- bruker defaultverdien for etasje:
INSERT INTO leilighet(leil_nr, ant_rom, ant_kvm, bygn_id, and_eier_nr)
            VALUES(DEFAULT, 5, 110, 1, 4);


-- Forsøk på å legge inn ugyldige data:

-- 1. Brudd på entitetsintegriteten, dvs at vi forsøker å legge inn en rad
--    med en primærnøkkelverdi som fins fra før

INSERT INTO poststed VALUES(7020, 'Trondheim');

Feilmelding: 1062 - Duplicate entry '7020' for key 'PRIMARY'

-- 2. Brudd på referanseintegriteten, dvs forsøker å bruke en fremmednøkkelverdi
--    som ikke fins som primærnøkkelverdi i den tabellen den refererer til.
--
--    Forsøker å legge inn en leilighet i bygning nr 10, men denne bygningen fins ikke:
INSERT INTO leilighet VALUES(DEFAULT, 5, 110, 3, 10, 3);

Feilmelding: Duplicate entry '3' for key 'and_eier_nr'

-- 3. Brudd på UNIQUE-krav:
INSERT INTO leilighet VALUES(DEFAULT, 5, 110, 3, 1, 4);

Feilmelding: Duplicate entry '4' for key 'and_eier_nr'
```

*Oppgave d)*

Borettslag:

- Fremmednøkkel postnr: Bør ikke være NULL, pga at alle adresser i Norge har et postnr.

Bygning:

- Fremmednøkkel postnr: Bør ikke være NULL, av samme grunn som for Borettslag.
- Fremmednøkkel bolag_navn: Kan ikke være NULL, ettersom en bygning må tilhøre et borettslag.

Leilighet:

- Fremmednøkkel bygn_id: Kan ikke være NULL, ettersom en leilighet må ligge i en bygning.
- Fremmednøkkel and_eier_id: Kan være NULL, ettersom en leilighet ikke må ha en andelseier.

Andelseier:

- Fremmednøkkel bolag_navn: Kan ikke være NULL, ettersom andelsieren må være med i et borettslag.

Dette er i samsvar med vurderinger vi har gjort tidligere, det svarer til 1..1-kravene i ER-modellen.

### Konklusjon – fremmednøkler NULL og/eller UNIQUE?

Dette framgår av ER-modellen. Fremmednøkler som er en følge av 1..1-multiplisitet (eksistensavhengighet) kan ikke ha verdien NULL, de som er en følge av 0..1-multiplisitet kan være NULL.

Bruk UNIQUE på fremmednøkler som er en følge av en en-til-en-sammenhengstype. SQL-syntaks:

```
CONSTRAINT leilighet_unique1 UNIQUE(and_eier_nr)
```

Primærnøkler skal sikre entitetsintegriteten, og de kan aldri være NULL.

*Oppgave e)*

ON DELETE CASCADE betyr følgende i hvert enkelt tilfelle:

Borettslag:

- Fremmednøkkel postnr: Dersom postnummeret slettes, vil også borettslaget slettes. Ikke lurt! Borettslaget må isteden få et nytt postnummer.

Bygning:

- Fremmednøkkel postnr: Dersom postnummeret slettes, vil også bygningn slettes. Ikke lurt! Bygningen må isteden få et nytt postnummer.
- Fremmednøkkel bolag_navn: Dersom borettslaget slettes, vil også alle tilhørende bygninger fjernes fra databasen. Dette er rimelig.

Leilighet:

- Fremmednøkkel bygn_id: Dersom bygningen slettes, vil også alle leilighetene i bygningen fjernes fra databasen. Dette er rimelig.
- Fremmednøkkel and_eier_id: Dersom andelseieren melder seg ut av Bygg&Bo, vil leiligheten han/hun hadde fjernes fra databasen. Dette er ikke rimelig. Isteden må leiligheten overføres til en annen andelseier.

Andelseier:

- Fremmednøkkel bolag_navn: Dersom borettslaget slettes, vil alle andelseierne også fjernes fra databasen. Dette er ikke rimelig, mange av dem vil kanskje heller overføres til et annet borettslag.

ON UPDATE CASCADE betyr følgende i hvert enkelt tilfelle:

Borettslag:

- Fremmednøkkel postnr: Dersom postnummeret endres, vil også borettslaget få nytt postnr. Lurt! Dersom endringen av postnummer er *så* enkel.

Bygning:

- Fremmednøkkel postnr: Samme som for borettslag.
- Fremmednøkkel bolag_navn: Dersom borettslaget endresr navn, vil også alle bygninger få nytt navn på feltet bolag_navn. Lurt!
- Leilighet: Fremmednøkkel bygn_id: Dersom bygningen endrer id, vil også alle leilighetene få forandret bygn_id. Lurt!

Fremmed

- Fremmednøkkel and_eier_id: Dersom andelseieren endrer nummer, vil også leiligheten han/hun eier få ny and_eier_id. Lurt!

Andelseier:

- Fremmednøkkel bolag_navn: Dersom borettslaget forandrer navn, vil også bolag_navn for alle andesleierne forandres. Lurt!

### Konklusjon ON DELETE/UPDATE CASCADE

ON UPDATE CASCADE er nesten alltid lurt. MySQL og PostgreSql har støtte for denne funksjonaliteten, men ikke Oracle eller Java DB. Funksjonaliteten må i de siste tilfellene programmeres. Varierende støtte for ON UPDATE CASCADE kan være en grunn til at man ikke bør bruke informasjonsbærende attributter som primærnøkler. Ved å bruke løpenummer er sjansen mindre for at de behøver å forandres.

ON DELETE CASCADE må brukes med stor forsiktighet. I praksis slettes sjelden data fra en database, man er ofte interessert i historikk. Istedenfor å slette data har man gjerne et attributt som forteller når borettslaget ble oppløst, personen meldt ut, bygningen solgt, etc. Dog har alle de nevnte databasesystemene støtte for denne funksjonaliteten.