Høgskolen i Østfold

EKSAMEN

Emnekode:	**Emnenavn:**
ITF10306	Databaser
Dato:	**Eksamenstid:**
13. mai 2016	Kl. 9.00 – kl. 13.00, 4 timer
Hjelpemidler:	**Faglærer:**
Syntaksoversikt (vedlagt oppgaven)	Edgar Bostrøm

Om eksamensoppgaven og poengberegning:

Oppgavesettet består av 4 sider inklusiv denne forsiden.

Vedlegget består av 6 sider.

Oppgavesettet består av 4 oppgaver.

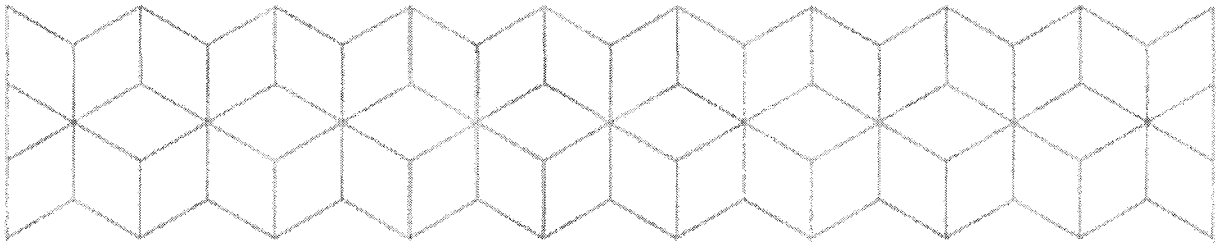
Kontroller at oppgavesettet er komplett før du begynner å besvare spørsmålene.

Alle oppgaver teller likt ved sensuren.

Les gjennom hele oppgavesettet før du starter!

Sensurfrist: 7. juni 2016

Karakterene er tilgjengelige for studenter på Studentweb senest 2 virkedager etter oppgitt sensurfrist. www.hiof.no/studentweb



Eiendomsmegling[1].

Vi skal se på et system for håndtering av eiendomssalg.

Jobben, dvs. å prøve å selge eiendommen, kalles av eiendomsmeglerne for et oppdrag. Som kjent selges eiendommer ved at det legges inn – forhåpendligvis flere - bud på eiendommen. Vanligvis er det slik at eiendommen selges på grunnlag av det høyeste budet, men det er i prinsippet ikke noen ting i veien for at den blir solgt til en annen av budgiverne, eller ikke solgt i det hele tatt. Dessuten kan samme eiendom bli solgt flere ganger, men da er det selvsagt et nytt oppdrag.

Framstillingen er naturligvis forenklet, og omfatter kun salg av bebygde eiendommer, ikke utleie eller salg av tomter. Vi tar bare for oss norske eiendommer. Vi antar også at systemet er laget bare for ett eiendomsmeglerfirma (men som selvsagt kan ha flere meglere ansatt). Vi gjør også begrensningen at det bare er personer som kan eie eiendommer og som legger inn bud. I virkeligheten kan naturligvis institusjoner, aksjeselskaper, kommuner etc. både eie eiendommer og legge inn bud.

Noen av tabellene som opplagt trengs i denne sammenhengen er gitt under.

Primærnøkkel er understreket, og fremmednøklene har samme navn som de kolonnene de refererer til. De fleste kolonnene burde være selvforklarende, men vi koster på oss et par forklaringer: Solgt er en kolonne som viser om oppdraget har ført til et salg eller ikke. Vi kan anta at denne har datatypen boolean el.l., evt. et heltall slik at solgt = 1 betyr at eiendommen er solgt, mens solgt = 0 betyr at den ikke er solgt. Eiendomnr antar vi er et entydig, fast nummer som er tildelt den enkelte eiendom som finnes i Norge[2]. PersonID i tabellen bud henviser til den personen som la inn budet.

KOMMUNE

Kommunenr	**Kommunenavn**
0101	Halden
0104	Moss
...	...

PERSON

PersonID	**Fornavn**	**Etternavn**	**Postnr**	**Poststed**
9012	Marianne	Hansen	...	...
9132	Mari	Jebsen	...	...
...	...	...	...	...

OPPDRAG

Oppdragsnr	**Eiendomnr**	**Gateadresse**	**Postnr**	**Poststed**	**Kommunenr**	**Prisantydning**	**Solgt**
21231	...	...	...	...	...	3300000	
21884	...	...	...	...	...	2990000	✔
21111	...	...	...	...	...	...	
....	....	...	...	...	...	...	...

BUD

Budnr	**Oppdragsnr**	**Budpris**	**PersonID**	**Dato**	**Tid**	**Forbehold**
88212	21231	1250000	9012	....	....	
88781	21231	1310000	1234	....	....	Kan ikke overta før oktober.
...	....	.....	....	....	....	....

Strukturen her er muligens ikke ideell, se senere oppgaver.

---

[1] Takk til NN hos Privatmegleren for nyttig informasjon i forbindelse med laging av oppgaven!

[2] I virkeligheten er det langt fra så enkelt! Eiendommer identifiseres med en kombinasjon av kommunenr, gårsdnr, bruksnr, evt. festenummer, evt. også seksjonsnr.

Oppgave 1. SQL-spørringer. Tid: 1 time.

a) Lag en spørring som plukker ut alt om alle bud på oppdrag nr 21231, med de høyeste budene først.

b) Lag en spørring som viser alle ***usolgte*** oppdrag i Fredrikstad med prisantydning på under 2 millioner (2000000) eller mer enn 5 millioner (5000000).

c) Lag en liste med gateadresse, postnr og poststed på oppdrag hvor det ikke er lagt inn noe bud.

d) Hva er høyeste bud i forbindelse med oppdrag 21884? Svaret skal inneholde budprisen og hvem (for- og etternavn) som har lagt inn budet.

e) Lag en liste over gjennomsnittlig prisantydning for usolgte eiendommer pr. kommune. Kommunenr, kommunenavn og den gjennomsnittlige prisantydning skal være med.

f) Det hender at samme eiendom blir omsatt flere ganger. Lag en liste over de eiendommene (med eiendomnr, gateadresse, postnr og poststed) hvor det finnes 3 eller flere oppdrag i databasen, sortert slik at de som er solgt flest ganger kommer øverst på lista.

enten:

g) Lag en spørring som lister opp alt om de eiendommene hvor forskjellen (opp eller ned) mellom prisantydning og høyeste bud er mer enn 20% av prisantydning.

eller:

h) Lag en spørring som lister ut alt om de eiendommene hvor forskjellen på høyeste og laveste bud er mer enn 500000.

Oppgave 2. Normalisering Tid: 1 time.

a) Forklar hva som er hensikten med normalisering. Forklar også kort hva denormalisering er og hvorfor det av og til gjøres.

b) Vi ser på tabellen Oppdrag. Anta at den er på minst 1NF.

- Angi alle determineringer (funksjonelle avhengigheter) som finnes i denne tabellen (både de som er ok og eventuelle determineringer som ikke er ok). Dette kan gjerne vises med et determineringsdiagram, men opplisting går også greit.
- Hvilken normalform er tabellen på? Begrunn svaret.
- Sett opp en normalisert versjon av denne strukturen (dvs. hvilke tabeller som framkommer som resultat av normaliseringen).

Oppgave 3. Datamodellering Tid: 1 time.

I denne oppgaven skal du tegne en datamodell for utvidelse av systemet. Både primærnøkler, fremmednøkler, minima og maksima skal være med. Verb/roller/relasjonsnavn som beskriver relasjonstypene/assosiasjonene skal være med der de ikke er helt opplagt. Velg selv hvilken type notasjon du bruker, men vær konsekvent!

- Ta utgangspunkt i innledningen og oppgave 1, men slik at eventuelle brudd på normalisering blir rettet opp (jf. også oppgave 2c).

- Det er behov for en del utvidelser:
  - Systemet skal alltid inneholde hvilken megler som er ansvarlig for hvert oppdrag (ansattnr, for- og etternavn). I noen tilfelle settes det i tillegg opp en annen person som saksbehandler for oppdraget, men det er relativt sjelden. Også andre ansatte og evt. freelancere tildeles egne ansattnr i systemet og lagres sammen med meglere.
  - Eieren eller eierne av eiendommen skal være med. Det vil ofte være flere eiere, men i så tilfelle er det en som er oppgitt som "hovedeier" – kalt fullmektig - i forbindelse med salget. I forhold til oppgjøret må man også ha med hvor stor eierandel (i prosent) som hver av eierne har. En som er eier av en eiendom vil senere ofte legge inn bud på en annen eiendom osv., så det lønner seg å knytte både eiere og budgivere til Person, jf. tabellene i innledningen.
  - Type eiendom skal være med (f.eks. E: enebolig, H: hytte, B: borettslagseiendom).
  - Når en eiendom er solgt, skal vi vite hvilket bud som "fikk tilslaget", eller - som eiendomsmeglerne sier - "budaksept". Vi antar at eiendommer som blir solgt, alltid blir solgt på grunnlag av et bud som er lagret i systemet.
  - Bransjen har klare regler for dokumentasjon av det arbeidet de gjør. Derfor foretas det en logging av alle aktiviteter som gjøres i forbindelse med oppdraget. Denne loggen skal inneholde type aktivitet (det er faste slike, som "Oppdrag startet", "Bestilt eierskifteforsikring", "Bestilt fotograf", "Visning", "Mottatt bud", "Oppgjør foretatt" m.fl.). Disse aktivitetstypene er nummerert, og man må kunne legge inn nye slike.
    For hver aktivitet skal man dermed ha med en dato da aktiviteten ble lagt inn, hvem (ansattnr) som la den inn, hvilken type aktivitet og eventuelt en kommentar.

Kommenter også kort om noe fra den opprinnelige tabellstrukturen i oppgave 1 bør endres. Drøft gjerne også ulike løsninger som du vurderte i denne prosessen.

Merk: Modellen er ikke så vanskelig, men inneholder mange detaljer. Les derfor godt gjennom oppgaven *flere ganger* og se at du har fått med deg alt.

Oppgave 4. Utsnitt Tid: 1 time.

Ved bedømmingen legges det mest vekt på den generelle forklaringen.

a) Forklar generelt hva et utsnitt er, ulike former for bruk av utsnitt, om oppdaterbarhet av utsnitt m.m.

b) I det mest vanlige systemet som er i bruk for eiendomsmeglere i Norge ligger alle som på en eller annen måte er med i prosessen i en felles "kontakt"-tabell, men slik at de har en kode som viser hvilken type kontakt (eier, megler, takstmann, kunde, grunneier osv.) de er.
   - Forklar hvorledes utsnitt / view kan brukes i denne sammenhengen for å få fram utsnitt som kun inneholder en av disse typene kontakt. Vurder kort fordeler og ulemper med denne måten å organisere data på i forhold til den som er foreslått i oppgave 3.
   - Tenk deg at Person var utvidet med Kontakttype, f.eks. M (megler), K (kunde) osv. Lag et utsnitt som du kaller Megler som plukker ut meglerne fra Person-tabellen.

SQL-syntaks – noen elementer

- Syntaksoversikten gjelder SQL2.
- Oversikten er ikke fullstendig og heller ikke helt presis, men er forhåpentligvis til hjelp.
- [ ] brukes om frivillige elementer, det er altså ikke med i SQL-språket.
- ¦ brukes som eller, det er altså ikke med i SQL-språket.
- { ….. } start, hhv. slutt, " ".
- < ….> brukes for å beskrive et språkelement. Disse beskrives eller er beskrevet tidligere i syntaksbeskrivelsen eller følger av det generelle mønsteret fra andre.
- **Fet skrift** brukes om faste språkelementer

1 CREATE / ALTER / DROP TABLE-SETNING

Create table

CREATE TABLE <tabellnavn> (<kommaseparert tabelldefinisjonsliste>);

<kommaseparert tabelldefinisjonsliste>:

- liste med en eller flere elementer som er enten <kolonnedefinisjon> eller <skrankedefinisjon>
- hvis listen består av flere elementer, er det komma mellom disse.
- listen må ha minst en <kolonnedefinisjon>, har som regel også minst en <skrankedefinisjon>

<kolonnedefinisjon>:

- <kolonnenavn> <datatype> [**NOT NULL**] [**DEFAULT** <verdi>] , samt eventuell <skrankedefinisjon>, men uten (den første) kommaseparerte kolonnelisten.

<skrankedefinisjon>

- [**CONSTRAINT** <skrankenavn>] **PRIMARY KEY** (<kommaseparert kolonneliste>)
- [[**CONSTRAINT** <skrankenavn>] **FOREIGN KEY** (<kommaseparert kolonneliste>)] **REFERENCES** <tabell> (<kommaseparert kolonneliste>) [**ON UPDATE** <ref.oper.>] [**ON DELETE** <ref.oper.>]
- [**CONSTRAINT** <skrankenavn>] **UNIQUE** (<kommaseparert kolonneliste>)
- [**CONSTRAINT** <skrankenavn>] **CHECK** (<betingelse>)
- noen flere finnes.

<kommaseparert kolonneliste>:

- en eller flere kolonner. Hvis det er flere kolonner er disse adskilt med komma

<ref.oper.>: (dvs. referanseintegritetsoperasjon)

- {**RESTRICT** ¦ **NO ACTION** ¦ **CASCADE** ¦ **SET NULL**}

Alter table

ALTER TABLE <tabellnavn>
{**ADD** ¦ **DROP**} {[**COLUMN**][3] <kolonnedefinisjon> ¦ <skrankedefinisjon>};

Noen systemer mangler **DROP.**

Drop table

DROP TABLE <tabellnavn>;

[3] Skal være med for noen systemer, skal utelates for andre.

2 SELECT-SETNINGER.

Select-setning uten gruppering

SELECT [DISTINCT] <kommaseparert resultatiste>
FROM <kommaseparert tabellliste>
[**WHERE** <betingelse>]
[**ORDER BY** <ordnet kolonneliste med sortering>];

<kommaseparert resultatliste>:

- kommaseparert liste, hvor hvert element er en av
  - en kolonne
  - en beregning m.m.
  - en select-setninger som returnerer en verdi for hver verdi av de andre i listen.
- et element kan gis et eget navn (alias). Mest vanlig for å gi resultatet av en beregning et folkelig navn. Skrives <kolonne> / <beregning> **AS** <NyttNavn>.

<kommaseparert tabelliste>:

- enkleste form er en enkelt tabell eller en liste av tabeller med komma mellom
- et element i denne kan også være alias, på formen <tabellnavn> [**AS**] <aliasnavn>. Alias må brukes hvis man trenger to eller flere benevnelser for samme tabell.
- elemenene i denne kan være **INNER JOIN**, **LEFT [OUTER] JOIN** eller **RIGHT [OUTER] JOIN** .
  Eks.: <tabell1> **LEFT OUTER JOIN** <tabell2> **ON** <tabell1>.<kolonne1> = <tabell2>.<kolonne2>
- inner, left og right join kan også nestes i flere nivåer.

<betingelse>:

- består av en eller flere <enkeltbetingelse> evt. med **AND** eller **OR** mellom.
- paranteser brukes på vanlig måte, **AND** binder sterkere enn **OR**

<enkeltbetingelse>:

- er et utsagn som, for en gitt rad i from-setningen, resulterer i enten sant eller usant.
- ofte <kolonnenavn> = <verdi>, men kan også være >, >=, <>, <=
- hvis du ikke bruker INNER / LEFT / OUTER JOIN er det viktig å ha med <tabell1>.PK = <tabell2>.FK
- **BETWEEN** <startverdi> **AND** <sluttverdi>
- søking i starten av en streng (trunkert søking): <kolonne> **LIKE** '<startstreng>%'
- søking i om delstrengen finnes i kolonnen: <kolonne> **LIKE** '%<delstreng>%'
- **NOT** brukes til å negere en enkeltbetingelse eller sammensatt betingelse. Binder sterkere enn **AND** og **OR**.
- <kolonne> **IS [NOT] NULL** brukes for å sjekke om en kolonne er NULL, evt. ikke er NULL.
- delspørringer med IN / NOT IN:
  <kolonne> **[NOT] IN**
  (SELECT <enkeltkolonne> .....)
- delspørringer med EXISTS / NOT EXISTS:
  [NOT] EXISTS
  (**SELECT** .....)
- **ALL** og **ANY** brukes på resultatet av en delspørring.
  - **ALL** er sann hvis alle i delspørringen oppfyller kriteriet. Usant hvis delspørringen er tom.
  - **ANY** er sann hvis noen (en eller flere) oppfyller kravet. Sant hvis delspørringen er tom. **SOME** er ekvivalent med ANY.
  - Tips: **WHERE** <kolonne> >= **ALL** (**SELECT** <kolonneliste> .....) er det samme som **WHERE** <kolonne> = (**SELECT max**(<kolonne>) .....)

<ordnet kolonneliste med sortering>:

- som kolonneliste, men i sorteringsprioritet, og hver kolonne kan etterfølges av **ASC** eller **DESC**.
- hvis det ikke oppgis sortering, blir sorteringen i stigende rekkefølge.

Select-setning med gruppering / aggregering

For det som er felles for alle select-setning henvises det til 0.

SELECT <kommaseparert resultat- eller aggregeringsliste>
FROM <kommaseparert tabelliste>
[**WHERE** <betingelser>]
[**GROUP BY** <kommaseparert resultatliste>]
[**HAVING** <betingelse for gruppe>]
[**ORDER BY** <ordnet kolonneliste med sortering>[4]];

<kommaseparert resultat- eller aggregeringsliste>:

- NB! hvert element er enten et element fra group by-listen eller en <aggregeringsfunksjon>.

<aggregeringsfunksjon>:

- {**count(***)|**count**(<kolonne>)|**sum**(<kolonne>)|**max**(<kolonne>)|**min**(<kolonne>)|**avg**(<kolonne>)| mfl.}
- <kolonne> kan også være en beregning
- noen systemer har også mulighet for **count** (**distinct** <kolonne>)), teller altså opp antall ulike.
- hvis vi ikke har med **GROUP BY** gjelder aggregeringen for hele tabellen

<betingelse for gruppe>:

- bare aktuelt dersom man har **GROUP BY**.
- betingelse som gjelder gruppen, innholder ofte en aggregeringsfunksjon, f.eks. count(*) > 1, sum(<kolonne>) = (select sum(.....)
- kan inneholde **AND**, **OR**, **NOT** osv., på samme måte som <betingelse>

3 INSERT / UPDATE / DELETE

INSERT-setning

INSERT INTO <tabell> [(<kommaseparert kolonneliste>)]
{ **VALUES** (<kommaseparert verdiliste>) | <select-setning> } ;

UPDATE-setning

UPDATE <tabell>
SET <kommaseparert kolonne/verdi-liste>
[**WHERE** <betingelse>];

- I noen systemer kan <tabell> i stedet være en begrenset form for <kommaseparert tabellliste>

<kommaseparert kolonne/verdi-liste>:

- hvert element består av <kolonne> = <konstant> eller <kolonne> = <beregninget verdi, f.eks. på grunnlag av tidligere verdi>
- oftest bare en slik kolonne/verdi-kombinasjon, men kan være flere.

DELETE-setning

DELETE
FROM <tabell>
[**WHERE** <betingelse>];

[4] Når man bruker gruppering, kan order by også inneholde grupperingsfunksjoner, f.eks. order by sum(beløp) desc.

4 CREATE / DROP VIEW

Create view

CREATE VIEW <utsnittsnavn> [(<kommaseparert kolonneliste>)]
AS
<select-setning>;

- kolonnelisten er nødvendig hvis det ikke er fullt samsvar mellom kolonnenavn i select-setningen og utsnittet.

Drop view

DROP VIEW <utsnittsnavn>;

5 Indekser

CREATE [**UNIQUE**] **INDEX** <indeksnavn> **ON** <tabell> (<ordnet kolonneliste med sortering>);

DROP INDEX <indeksnavn>;

Noen systemer har andre mekanismer i tillegg.

6 GI / FRATA RETTIGHETER TIL TABELLER M.M.

GRANT <rettigheter> **ON** <tabell el.l.> **TO** <bruker/gruppeliste> [**WITH GRANT OPTION**];

REVOKE [<rettigheter> ¦ **GRANT OPTION**] **FROM** <tabell el.l.> **TO** <bruker/gruppeliste>;

<rettigheter>:
kommasepartert liste med en eller flere av **SELECT**, **INSERT**, **UPDATE** (<kolonnenavn>), **DELETE**, **ALL** m.fl..

<bruker/gruppeliste>:
kommasepart liste med en eller flere brukere eller grupper. I tillegg finnes ofte noen standardgrupper, som PUBLIC og DBA.

Noen variasjoner og begrensninger fra et system til et annet.

7 NOEN ANDRE ELEMENTER

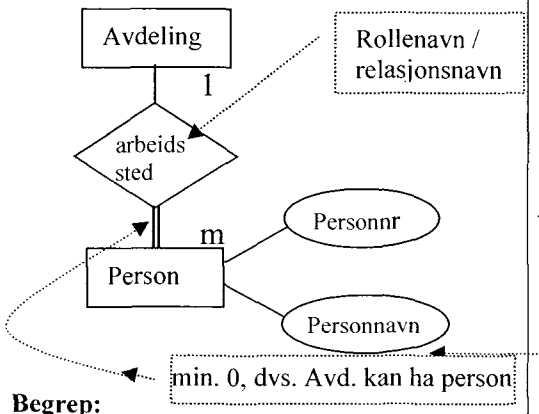
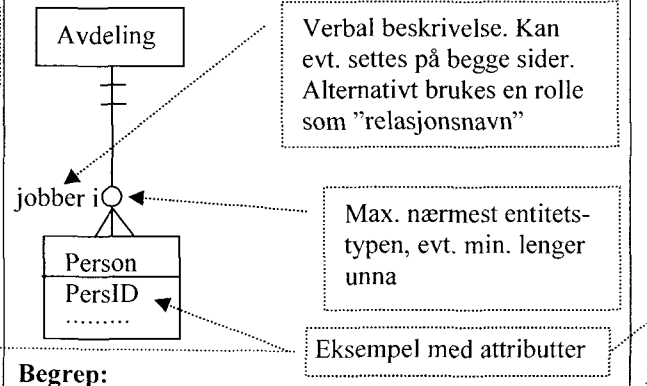
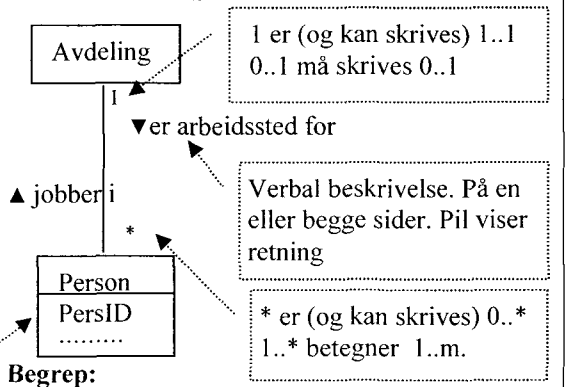
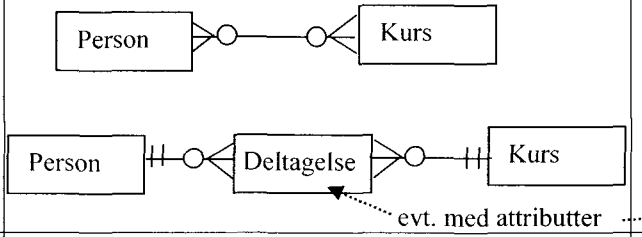
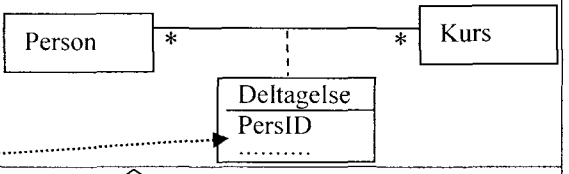
Muligheter for å lage / ta bort brukere etc., **CREATE USER**, gjerne sammen med **IDENTIFIED BY** <passord>. Tilsvarende **DROP USER**.

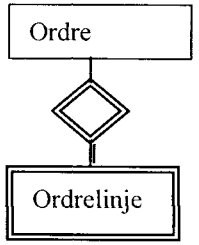
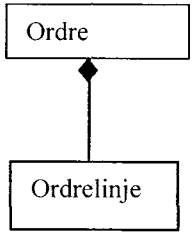
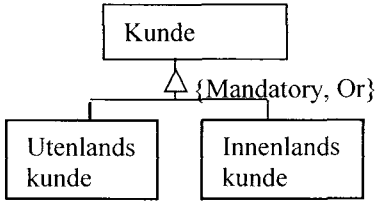
Muligheter for å lage nye databaser, **CREATE DATABASE** <databasenavn>

I noen systemer: laging av typer, domener etc.

Datamodellnotasjon i 3 dialekter: Chen, kråkefot og nedskalert UML.

En del detaljer og variasjoner er utelatt. Lær deg og bruk en av disse.

	Chens ER	**Kråkefot**	**nedskalert UML**
Grunnleggende. For alle dialekter: • attributter kan tas med eller utelates (avh. av hvor langt i prosessen og hvor stor modellen blir) • ditto for domener/datatyper • det finnes varianter for å vise min./max.	 **Begrep:** Entitet(styper), relasjon(styper), attributter.	 **Begrep:** Entitet(styper), relasjon(styper), attributter.	 **Begrep:** Entitet(styper) eller objektklasser, (multiplisitets)assosiasjoner, attributter.
Er repetisjoner tillatt?	Ja, på konseptuelt nivå	Nei – splittes ut i egne entitetstyper	Ja, på konseptuelt nivå
Eventuelle primær- og fremmednøkler	Tas gjerne ikke med	Hvis det tas med: Markeres f.eks. med primærnøkkel: understreking fremmednøkkel: prikket linje, *, el.l.	Hvis det tas med: markeres gjerne med {PK} hhv. {FK} bak attributtnavnet. Hvis (del av) begge deler: {PK,FK}
Entitetisering	Kan gjøres, men vanligvis settes det bare på attributter på relasjonen. Bare nødvendig ved 2. ordens entitetisering (entitetisering av noe som allerede er entitetisert eller kunne vært entitetisert).	Gjøres dersom "relasjonen skal inneholde attributter". 	Kan gjøres, men bare nødvendig ved det som ellers ville vært 2. ordens entitetisering. Assosiative entitetstyper m/ attributter kan legges på: 
n-ære relasjonstype / assosiasjoner (n >2)	Innebygdt i notasjonen, ingen forskjell på binære og n-ære.	Evt. entitetisering gjøres først, deretter henges nye entitetstyper på den nye entitetstypen.	Bruk  for å knytte dem sammen. Assosiative entitetstyper kan brukes

Avhengighet av andre entitetstyper (en entitet er avhengig av eksistensen av en annen entitet)	 kalles svak entitet / weak entity	Markeres ved at fremmednøkkelen er en del av primærnøkkelen (på mange-siden)	 ≈ komposisjon. Finnes også en mindre sterk kobling som kalles aggregering (hel-del, markeres med ◇ i stedet for ◆) .
Arv	Finnes ikke, må i tilfelle beskrives som 1:1, men gir ikke egentlig arv.	Finnes ikke, må i tilfelle beskrives som 1: 1, men gir ikke egentlig arv.	 I tillegg: kan beskrive kombinasjoner av mandatory/optional og om en overordnet kan kobles til max. 1 eller til flere underordnede (or eller and), se over. Kan også være arv med"ett barn", f.eks. bare "Kunde" og "Utenlandskunde".
Forhold til normalisering	Må evt. gjøre utsplittinger av repetisjoner	Er normalisert	Må gjøre evt. utsplittinger av repetisjoner
Overføring til relasjonsdatabaser	Overføres til kråkefot el.l. først (fra konseptuelt til logisk nivå) Alternativt: Legg på primær- og fremmednøkler Evt. repetisjoner må tas bort. Entitetstyper blir til tabeller. Relasjoner som gjelder 1:m tas bort, relasjoner som gjelder m:m blir egne tabeller.	Evt. mange-til-mange må entitetiseres. Ellers: entitetstyper blir til tabeller	Evt. repetisjoner må tas bort. Entietstyper/objektklasser blir til tabeller. Assosiasjonsattributter i m:m blir egne tabeller, andre m:m entitetiseres. Høyere ordens relasjonstyper blir til tabeller. Arv må omformuleres (flere alternativer finnes, ingen er helt gode). Dersom man bruker ORDB-utvidelser i systemer som har dette, kan arv implementeres.