Innledning

EKSAMEN

Emnekode: ITF10306
Emnenavn: Databaser
Dato: 21.05.19
Eksamenstid: 09.00 - 13.00.
Hjelpemidler: Syntaksoversikt (vedlagt oppgaven).
Faglærer: Edgar Bostrøm/Ida K. Thoresen
Sensurfrist: 11.06.19
Karakterene er tilgjengelige i Studentweb.

Om oppgavesettet
Oppgavesettet består av 3 tekstoppgaver og en flervalgsoppgave. Hver av de 4 oppgavene teller likt ved bedømningen.

Vedlegget består av 6 sider.

På flervalgsoppgaven er minst ett av svaralternativene korrekte, dette betyr at det kan være kun ett korrekt alternativ, men det kan også være flere eller alle.
For hvert korrekt alternativ gis det 1 poeng, mens det for galt svar gis -0,5 poeng.

Les gjennom hele oppgavesettet før du starter. Det vil kunne være informasjon underveis som er viktig for en senere oppgave. Være også svært nøye med å lese oppgaveteksten og kontrollere løsningen på hver deloppgave!

Vedlagt er en syntaksoversikt, som finnes ved siden av oppgavearket.

Du kan benytte utleverte ark der du ønsker å lage illustrasjoner, modeller etc. Når du gjør det husk å skrive **SE PAPIRARK** som svar på de oppgavene dette gjelder, så er du sikker på at sensor ser hva du har skrevet i tillegg til det som leveres elektronisk.

Karakterene er tilgjengelige for studenter på studentweb senest 2 virkedager etter sensurfrist.

Bompenger.

Noen av dere blir kanskje i dårlig humør av det, men mye av dagens oppgave handler altså om bompenger.
Vent med å deppe til etter eksamen!


Kr
Automatisk
bomstasjon
Automatic toll
800 m

1 Oppgave 1. SQL'e vi betale så mye?

Oppgave 1. SQL'e vi betale så mye? Tid: 1 time.

Vi snakker om bommer eller bomstasjoner, selv om det jo egentlig er steder med elektronisk registrering av passeringen. Dette svært forenklede systemet består av informasjon om bommene inkl. pris, dvs. hva en passering gjennom denne bommen koster, hvilke biler som passerer, og selve passeringen, inkl. tiden for passeringen.

Kjennetegn er det som vi mer folkelig kaller for bilnummer (f.eks. AA98765). Vi regner bare med norske kjøretøy.
Samme person (og firma) kan eie flere biler. For firmaer skrives firmanavnet inn som etternavn, mens fornavn-kolonnen ikke blir brukt. Tilsvarende bruker vi firmanr i fødselsnummer-kolonnen.
Tabellen Bil inneholder en Biltypekode, dette beskrives nærmere i oppgave 3.
De ulike bommene kan ha ulik pris for passering, men foreløpig regner vi at alle biltyper har samme pris for en gitt bom.
Passeringsnr i tabellen passering er en teller/løpenummer. Fremmednøklene i tabellen framkommer av korresponderende primærnøkler i de andre tabellene.

BIL

Kjennetegn Biltypekode Fødselsnr Etternavn Fornavn Adresse Postnr Poststed

BOM

Bomnr Bomnavn Pris

PASSERING

Passeringsnr Kjennetegn Bomnr Dag Måned År Tidspunkt

a. Skriv ut alt om bommer som inneholder 'tunnell' som en del av navnet sitt.

b. Skriv ut alle passeringer i 2019 for biler med kjennetegn som begynner på bokstavene AS og AA. Kjennetegn, bomnr og -navn, samt dato og tidspunkt skal skrives ut. Det skal sorteres slik at de nyeste passeringene kommer først.

c. Skriv ut alt fra tabellen Bil for de bilene som **ikke** har noen passeringer i det hele tatt i 2019.

d. Hvilken bom er den dyreste å passere gjennom? Bomnr, -navn og pris skal være med. Det kan godt være flere som er like dyre, og i så tilfelle skal alle disse være med.

e. Skriv ut antall passeringer for hver bomstasjon. Bomnr, -navn og antall passeringer skal med.

f. Skriv ut kjennetegn på biler som har betalt over 10.000 i bompenger i 2019.

g. Skriv ut kjennetegn, bomnr og -navn på de bilene hvor alle passeringene har skjedd på samme bom.

Du kan bruke et eget ark hvis du ønsker å lage illustrasjoner, modeller etc. Hvis du gjør det, skriv **SE PAPIRARK** for de

Skriv ditt svar her...

2 Oppgave 2. Mer SQL. Normalisering

Oppgave 2. Mer SQL. Normalisering Tid: 1 time.

a. Lag en CREATE TABLE-setning for å lage tabellen PASSERING. Tidspunkt kan du regne f.eks. som en streng eller du kan definere det med datatypen time. Det er en fordel hvis du får med at dato må være et tall fra 1 til 31, og måned må være et tall fra 1 til 12.

b. Definer fremmednøkler for PASSERING. Bruk ALTER TABLE-setninger. Ta med endringstyper, vi bruker RESTRICT for sletting og CASCDE for endring.

c. Bilen med kjennetegn AA12345 er byttet ut med en ny bil med kjennetegn AA98765. Vi skal legge inn kjennetegnet for den nye i stedet for den gamle. Bruk en UPDATE-setning for dette.

d. Skriv ut bomnr, -navn og antall passeringer for den bomstasjonen som har flest passeringer. Det kan godt være flere som har like mange passeringer.

e. Skriv ut kjennetegn på biler som har passert både bomnr 13 og bomnr 17 på samme dato.

f. Lag en liste med alle brudd på 2NF (2. normalform) som finnes i tabellen BIL. Alternativt kan du bruke et determineringsdiagram for å vise bruddene.

g. Foreslå ny tabellstruktur for BIL (evt. fordelt på flere tabeller) slik at det ikke bryter med 2NF

Du kan bruke et eget ark hvis du ønsker å lage illustrasjoner, modeller etc. Hvis du gjør det, skriv **SE PAPIRARK** for de

Skriv ditt svar her...

3 Oppgave 3. Datamodellering

Oppgave 3. Datamodellering Tid: 1 time.

Vi skal gjøre en del utvidelser i forhold til det enkle systemet i oppgave 1.

- Det finnes jo mange bomselskap i landet, og de har ansvar for de ulike bommene som finnes. Vi skal ha med informasjon om dette, med BomselskapID, BomselskapNavn, og hvilket bomselskap hver bom hører til.

- Personer (og firmaer) kan opprette en avtale (bomavtale) med ett og bare ett bompengeselskap, man har i tilfelle et avtalenr og dato for når avtalen ble inngått. Denne avtalen kan gjelde flere biler. Det typiske er da at de får f.eks. rabatt (f.eks. 30%) på passeringer for dette bompengeselskapets bommer, men denne kan variere litt fra avtale til avtale, noen er jo flinke til å forhandle!! De som er ekstra flinke til å forhandle kan også få prosenter hos et annet bomselskap enn den de har avtale med. Rabattprosenten(e) skal inkluderes i systemet.

- I praksis har bommene ulike priser for ulike biltyper. For hver bom må det derfor finnes en oversikt over ulike biltyper med biltypekode og -betegnelse (f.eks. BB - Personbil, LA - Lastebil, EL – Elbil osv., og nye koder og -betegnelser må kunne legges inn). Dette må være knyttet til den enkelte bil. Vi antar at en bil kun er knyttet opp mot en biltypekode.

- Det må også finnes en oversikt over priser for hver Biltype på hver bomstasjon (f.eks. at det for passering gjennom bomstasjon nr. 2918 koster 25 kr. for BB, 60 kr. for LA, osv.). Dette vil dermed danne grunnlaget for å kunne ha ulike priser avhengig av hvilken biltype det er.

Lag en datamodell hvor du tar utgangspunkt i det som er sagt i oppgave 1 og 2, og du legger til (evt. endrer) ut fra det som er beskrevet over. Det er en fordel hvis både min. og max. er med i modellen, og tilsvernde med verb/roller der det kan klargjøre modellen.
Kommentér der du mener det kan være tvil om hvordan strukturen bør være.

(Det legges mest på modellen, men kommentarer vil også telle med.)

Du kan bruke et eget ark hvis du ønsker å lage illustrasjoner, modeller etc. Hvis du gjør det, skriv **SE PAPIRARK** for de

Skriv ditt svar her...

i Oppgave 4 - Flervalgsoppgave

På hver oppgave er minst ett av svaralternativene korrekte, dette betyr at det kan være kun ett korrekt alternativ, men det kan også være flere eller alle.

For hvert korrekt svaralternativ gis det 1 poeng, mens det for galt svar gis -0,5 poeng.

4

Oppgave 4.a

Hvilke strukturer av datamodeller har vi?
Velg ett eller flere alternativer

- Fysisk nivå
- Konseptuelt nivå
- Praktisk nivå
- Logisk nivå

5

Oppgave 4.b

Hva er **galt** om fremmednøkler?
Velg ett eller flere alternativer

- Primærnøkkelen fra hovedtabellen blir fremmednøkkel i detaljeringstabellen
- Uten fremmednøkler hadde vi ikke klart å koble informasjon fra forskjellige tabeller sammen
- Fremmednøklene må aldri settes på mangesiden i en relasjon
- I en identifiserende struktur benyttes fremmednøkkelen som en del av primærnøkkelen i detaljeringstabellene

6

Oppgave 4.c

Vi kan gjøre flere typer kontroll av lovlige verdier. Hvilke?
Velg ett eller flere alternativer

- Kontroll av maks antall
- Kontroll i flere dimensjoner
- Kontroll i relasjoner
- Kontroll i ett nivå

7

Oppgave 4.d

Hva er korrekt om nettverk?
Velg ett eller flere alternativer

- Både rettede og urettede grafer er en type nettverk
- En relasjonsdatabase kan beskrive visse nettverk
- Et nettverk er et system av noder og kanter
- En matrise kan benyttes for å beskrive et nettverk

8

Oppgave 4.e

Hva må være oppfylt for at en tabell skal være på, 2NF, 2. normalform?
Velg ett eller flere alternativer

- Deler av nøkkelen skal ikke kunne determinere ikke-nøkkelattributter
- Alle attributter skal være atomiske
- Enhver determinant er en kandidatnøkkel
- Ikke-nøkkelattributter skal ikke være transitivt avhengige av primærnøkkelen

9

Oppgave 4.f

Hvordan kobler vi data i flere tabeller sammen i en relasjonsdatabase?
Velg ett eller flere alternativer

- Vi kobler data sammen basert på kolonnenavnene
- Vi kobler data sammen basert på hvordan dataene ser ut
- Vi kobler data sammen basert på likhet i verdier
- Vi kobler data sammen basert på rekkefølgen av radene i tabellen

10

Oppgave 4.g

Hvilke egenskaper regnes som faste egenskaper i en relasjonsdatabase?
Velg ett eller flere alternativer

- Alle kolonner må inneholde en verdi
- En kolonne skal inneholde kun én verdi
- Relasjonsdatabaser består kun av tabeller
- Radene i tabellene er usorterte

11

Oppgave 4.h

Hva er korrekt om følgende relasjonsoperatorer i forbindelse med relasjonsdatabaser?
Velg ett eller flere alternativer

- Ved en projeksjon oppretter vi en ny relasjon ut i fra bestemte attributter i en eksisterende relasjon
- Ved en forening oppretter vi en ny relasjon ved å sette sammen hver tuppel i en relasjon med hver tuppel i en annen relasjon
- Når vi benytter en seleksjon(restriksjon) oppretter vi en ny relasjon ut i fra bestemte tupler i en eksisterende relasjon
- Ved et produkt oppretter vi en ny relasjon med sammensetting av tupler fra to relasjoner på et felles attributt.

12

Oppgave 4.i

Hva er **ikke** korrekt når vi snakker om en visning(view)?
Velg ett eller flere alternativer

- En visning kalles gjerne en virtuell tabell
- Det lagres data i en visning
- En visning er et søkbart objekt i en database som defineres av en spørring
- Man kan ikke spørre mot en visning likt som man kan mot en tabell

13

Oppgave 4.j

Hva er korrekt når vi snakker om alias i database sammenheng?
Velg ett eller flere alternativer

- Et alias benyttes for å gi en tabell et midlertidig navn
- Aliaset eksisterer kun så lenge spørringen benyttes
- Et alias er et annet navn på en tabell
- Alias benyttes gjerne for å gi kolonner et mer forståelig navn

14

Oppgave 4.k

Hva stemmer om følgende måter å gjenfinne data på?
Velg ett eller flere alternativer

- Ved å benytte indekser oppretter vi en hjelpetabell som inneholder indeksverdiene og som benyttes for raskt oppslag i grunntabellen som er indeksert.
- Ved et full søk (lineær søk) har vi sorterte data og halverer datamengden vi i søker i hver gang vi søker. Vi vet verdien vi leter etter, sjekker verdien midt i søkemengden og fortsetter søket i den delen av mengden vi vet vår data befinner seg i.
- Ved et binærsøk har vi usorterte data og må derfor søke gjennom alle radene i tabellen for å finne den dataen vi leter etter.
- Hashing er en metode som benytter en hashingfunksjon for å gi alle mulige verdier et resultat innenfor et ønsket intervall. Hensikten med dette er å enkelt finne data med like eller ulike verdier.

15

Oppgave 4.l

Hvilke egenskaper kan en transaksjon ha?
Velg ett eller flere alternativer

- Consistency - databasen må være konsistent før og etter en endt transaksjon
- Isolation - transaksjoner kan ikke arbeide med data som allerede er i midlertidig endret tilstand
- Atomicity - alt eller ingenting blir gjennomført
- Durability - resultatet av den utførte transaksjonen er endelig/vil ikke forsvinne

16

Oppgave 4.m

Hva er korrekt om en delt lås, leselås?
Velg ett eller flere alternativer

- En delt lås benyttes kun ved lesing
- Forutsetter at det ikke er andre låser på objektet
- Godtar at andre transaksjoner kan lese, men ikke endre dataene dersom en transaksjon holder delt lås på et objekt.
- En leselås godtar at flere transaksjoner leser de samme dataene

17

Oppgave 4.n

Hvilke måter kan vi gjenopprette en database på?
Velg ett eller flere alternativer

- Foroverrulling
- RAID
- Bakoverrulling
- Fjerne låser

Question 1
Attached

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

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> (det finnes noen flere enn de som er omtalt her)

- [**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>**)**

<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**][1] <kolonnedefinisjon> ¦ <skrankedefinisjon>}**;**

Noen systemer mangler **DROP.**

Drop table

DROP TABLE <tabellnavn>**;**

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

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>]**;**

<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>

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>];

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>**;**

Indekser

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

Noen systemer har andre mekanismer i tillegg.

Gi / frata rettigheter til tabeller, laging av brukere, databaser 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.

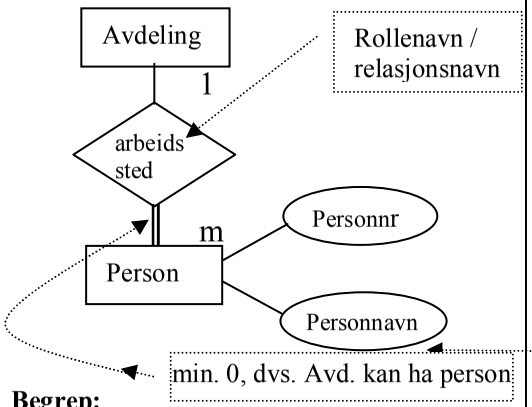
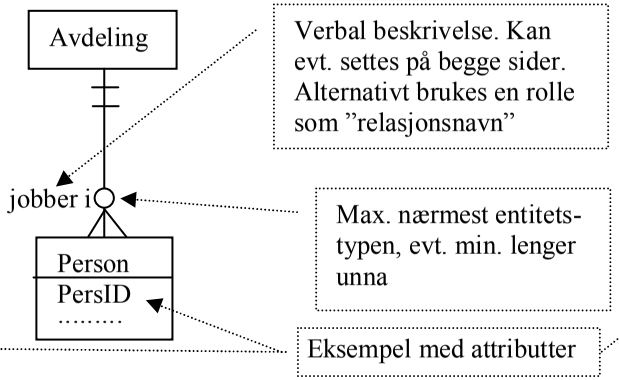
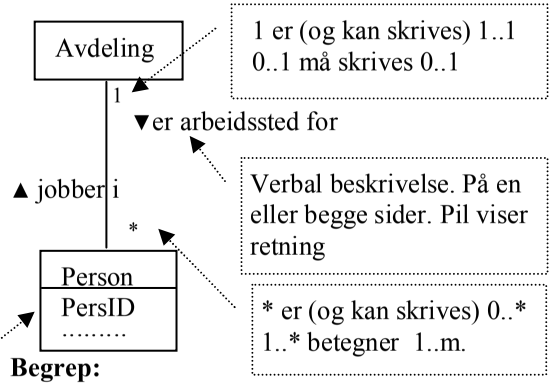
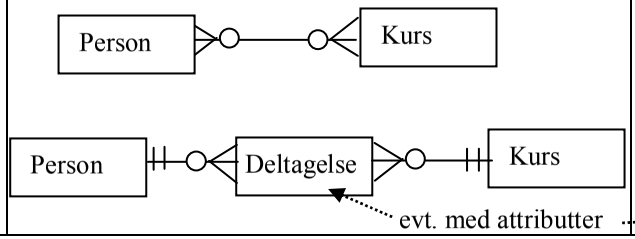
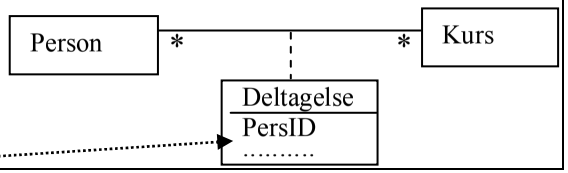
Noen variasjoner og begrensninger fra et system til et annet.

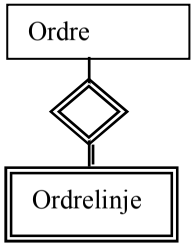
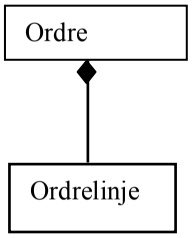
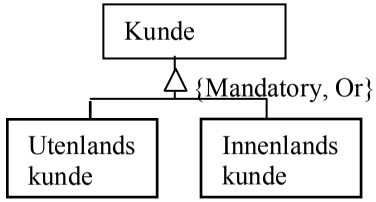
Annet:

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.

	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)	 kalles komposisjon. Finnes også en mindre sterk kobling som kalles aggregering (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.

Question 2
Attached

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

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> (det finnes noen flere enn de som er omtalt her)

- [**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>**)**

<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**][1] <kolonnedefinisjon> ¦ <skrankedefinisjon>}**;**

Noen systemer mangler **DROP.**

Drop table

DROP TABLE <tabellnavn>**;**

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

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>]**;**

<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>

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>];

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>**;**

Indekser

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

Noen systemer har andre mekanismer i tillegg.

Gi / frata rettigheter til tabeller, laging av brukere, databaser 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.

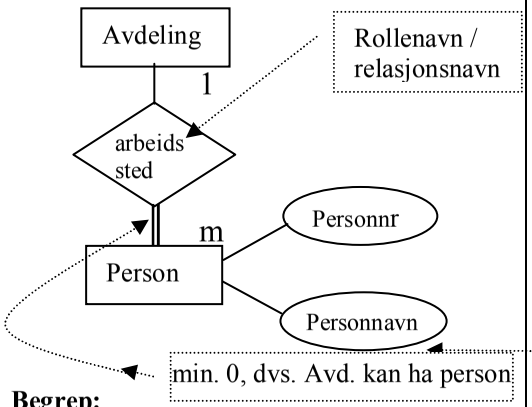
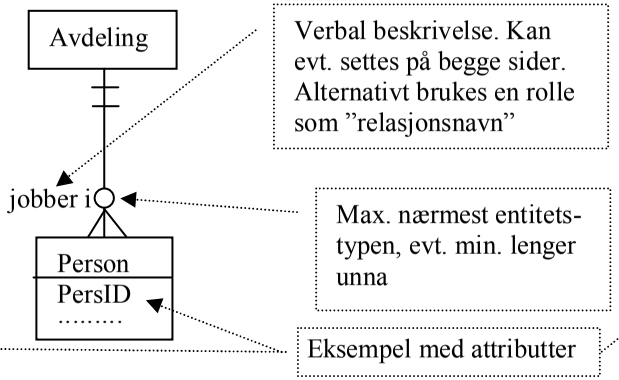
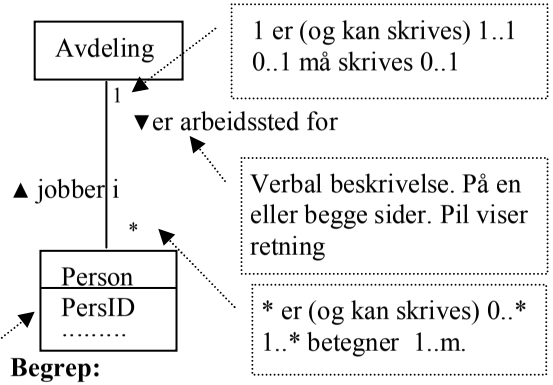
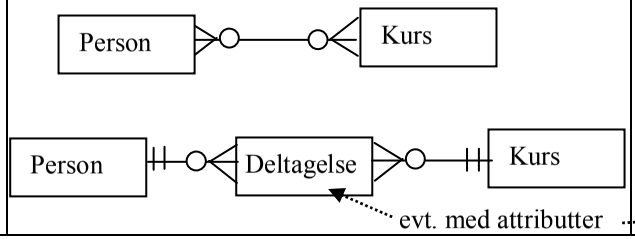
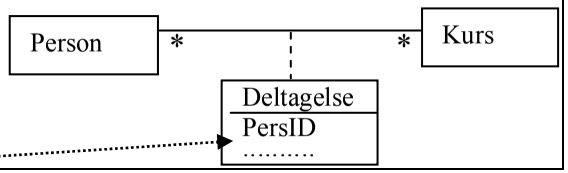
Noen variasjoner og begrensninger fra et system til et annet.

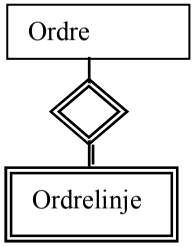
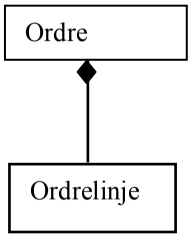
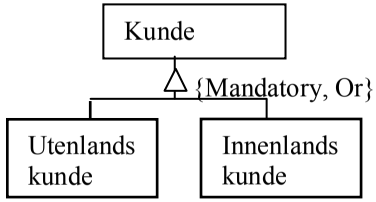
Annet:

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.

	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)	 kalles komposisjon. Finnes også en mindre sterk kobling som kalles aggregering (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.

Question 3
Attached

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

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> (det finnes noen flere enn de som er omtalt her)

- [**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>**)**

<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**][1] <kolonnedefinisjon> ¦ <skrankedefinisjon>}**;**

Noen systemer mangler **DROP.**

Drop table

DROP TABLE <tabellnavn>**;**

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

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>]**;**

<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>

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>];

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>**;**

Indekser

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

Noen systemer har andre mekanismer i tillegg.

Gi / frata rettigheter til tabeller, laging av brukere, databaser 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.

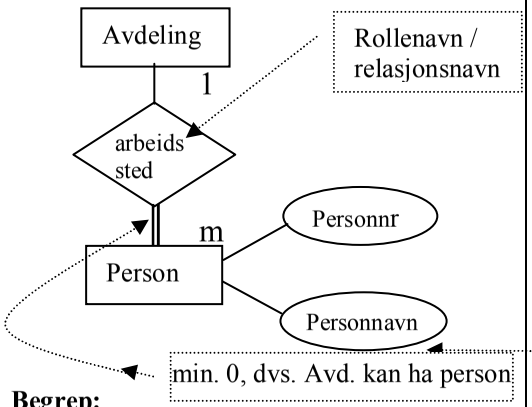
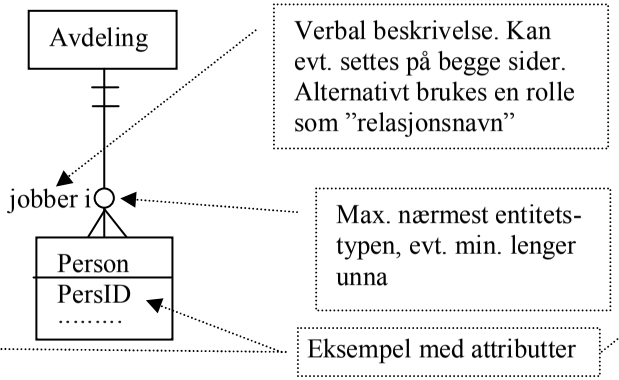
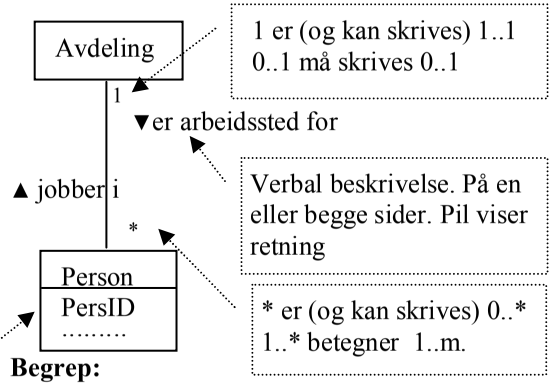
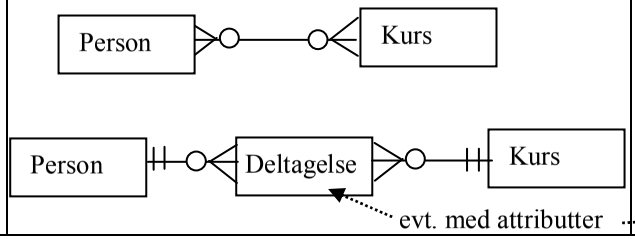
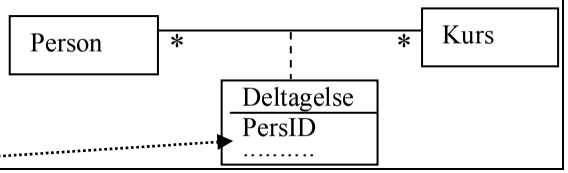
Noen variasjoner og begrensninger fra et system til et annet.

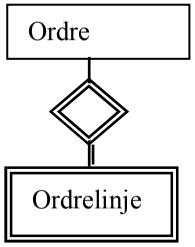
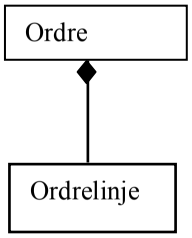
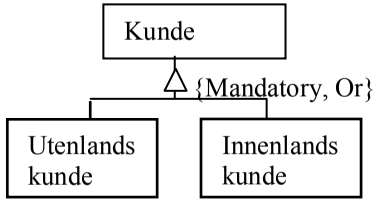
Annet:

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.

	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)	 kalles komposisjon. Finnes også en mindre sterk kobling som kalles aggregering (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.