

## i **Forside**

# UNIVERSITET I OSLO

## Det matematisk-naturvitenskapelige fakultet

### Eksamen i IN2090/INF1300

Dato: 6 desember 2019

Tid: 14:30–18:30

Sted: Silurveien 2 (Sal 3B, 3C, 3D, 4C, 4D)

Tillatte hjelpemidler:

- Læreboka Elmasri & Navathe: Fundamentals of Database Systems, Global Edition, 7th Edition (læreboken brukt i IN2090, fom. 2019).
- Læreboka Halpin & Morgan: Information Modelling and Relational Databases, Second Edition (læreboken brukt i INF1300 og IN2090 høsten 2018).
- 4 håndskrevne A4-sider med notater (2 ark hvis det er skrevet på begge sider)

Eksamen består av 3 deler (maksimal poengsum i parentes):

1. Modellering (40)
2. SQL (50)
3. Relasjonsmodellen (10)

I dette oppgavesettet skal du svare med digital håndtegning på modelleringsoppgavene, altså oppgave 1.1 og 1.2. Du bruker skisseark du får utdelt. Det er anledning til å bruke flere ark per oppgave. Se instruksjon for utfylling av skisseark i lenken under oppgavelinjen. På disse oppgavene kan du velge om du vil løse dem med modelleringsspråket ER eller ORM2.

Det er IKKE anledning til å bruke digital håndtegning på andre oppgaver enn oppgave 1.1 og 1.2. Det blir IKKE gitt ekstratid for å fylle ut informasjonsboksene på skisseark (oppgavekoder, kand.nr. o.l.).

På flervalgsoppgavene i oppgave 4, 12 og 13, er det minuspoeng for galt svar. Men den totale poengsummen for hver oppgaver blir aldri lavere enn 0.

## 1.1 Modelling: Kandidater

I denne (og neste) oppgave skal du lage en modell for en database som inneholder informasjon om intervjukandidater (for programvareutviklere) og intervjuene de deltar på. Du velger selv om du vil lage modellen i ER (første side i vedlagt dokument gir oversikt over ERs notasjon) eller ORM2 (side 2 over utover i vedlagt dokument gir oversikt over ORM2s notasjon).

Modellene skal tegnes på papir ved hjelp av Scantron. Du kan velge om du vil lage én stor modell for begge oppgavene, eller om du vil dele dem opp i to modeller. Dersom du velger å dele modellen opp i to, trenger du kun å tegne inn entitetene (og deres nøkler) for de entitetene du trenger fra oppgave 1 i modellen for oppgave 2. Du må gjerne inkludere kommentarer til modellene. Skriv og tegn tydelig.

Modellen vi skal lage i denne første oppgaven skal modellere kandidater:

1. For hver kandidat skal databasen kunne lagre et personnummer som er unikt for alle kandidater, et navn, en adresse, samt en email (som også er unik for hver kandidat).
2. Videre skal databasen inneholde informasjon om den nåværende arbeidsgiveren til hver kandidat. Dersom kandidaten er arbeidsledig skal ingen arbeidsgiver lagres. Dersom kandidaten har flere enn én arbeidsgiver (f.eks. dersom kandidaten har to deltidsjobber) er vi kun interessert i én av dem. (Altså, hver kandidat har maksimalt en arbeidsgiver i vår database). En arbeidsgiver kan ha mange kandidater som ansatte. For hver kandidat ønsker vi også å lagre kandidatens nåværende jobbtittel (f.eks. "produktansvarlig").
3. For hver arbeidsgiver er vi interessert i å lagre en unik ID, et navn, og kontaktinformasjon. Kontaktinformasjonen består igjen av telefonnummer, email og adresse.

*I denne oppgaven skal du svare med digital håndtegnning. Bruk eget skisseark (utdelt). Se instruksjon for utfylling av skisseark i lenken under oppgavelinjen.*

---

Maks poeng: 7

## 1.2 Modellering: Intervjuer

Vi skal fortsette på modellen om kandidater fra forrige oppgave. I denne oppgaven skal vi modellere intervjuene.

I likhet med forrige oppgave kan du velge mellom ER og ORM, og modellen skal lages på papir. Du kan velge om du vil lage én stor modell for begge oppgavene, eller om du vil dele dem opp i to modeller. Dersom du velger å dele modellen opp i to, trenger du kun å tegne inn entitetene (og deres nøkler) for de entitetene du trenger fra den forrige modellen.

Modellen du skal lage i denne oppgaven skal inneholde informasjon om intervjuer:

1. For hver kandidat skal databasen kunne lagre hvilke intervjuer en kandidat deltar på. Hvert intervju har et intervjunummer, samt en dato og et tidspunkt hvor intervjuet ble holdt. Intervjuer kan unikt identifiseres med kombinasjonen av dets intervjunummer og personnummeret til kandidaten som deltok på intervjuet. For eksempel: kandidaten med personnummer #123 deltar på intervju #1 holdt 18.12.19 klokken 14:00; kandidaten med personnummer #123 deltar på intervju #2 dato 21.12.2019 klokken 15:00; kandidaten med personnummer #456 deltar på intervju #1 dato 21.12.2019 klokken 15:00, osv.
2. Intervjuene blir holdt av en intervjuer på et bestemt emne (hint: bruk en ternær relasjon). Hvert emne har et unikt navn (f.eks. "datastrukturer") og en mengde tester (i databasen vil vi bare representere testene med et navn). Hver intervjuer har en ansatt-id og en mengde telefonnummer.
3. Databasen skal kunne lagre at mange intervju kan bli holdt av mange intervjuere for mange emner (med andre ord: gitt et intervju og en intervjuer, så kan vi ha mange emner; gitt et intervju og et emne, kan vi ha mange intervjuere; og gitt et emne og en intervjuer kan vi ha mange intervju). For eksempel: kandidat med personnummer #123 deltar på intervju #1 holdt av intervjuer med ansatt-id #111 om emne "datastrukturer"; kandidat med personnummer #456 deltar på intervju med nummer #1 holdt av intervjuer med ansattnummer #111 om emne "Java-programmering", osv.
4. Videre skal databasen kunne lagre at hver intervjuer er sertifisert på minst ett emne (men kan være sertifisert på flere). Hvert emne kan ha mange sertifiserte intervjuere. Sertifisering på et emne skjer på en bestemt dato og av en gitt sertifikatutsteder. (F.eks. intervjuer med ansatt-id #111 er sertifisert i emne "datastrukturer" fra dato 19.01.19 av Universitetet i Oslo.)
5. I tillegg har hvert emne nøyaktig én ansvarlig intervjuer (f.eks. den som er ansvarlig for å lage testene for emnet). En intervjuer kan være ansvarlig for maksimalt ett emne (men en intervjuer trenger ikke være ansvarlig for et emne).

*I denne oppgaven skal du svare med digital håndtegning. Bruk eget skisseark (utdelt). Se instruksjon for utfylling av skisseark i lenken under oppgavelinjen.*

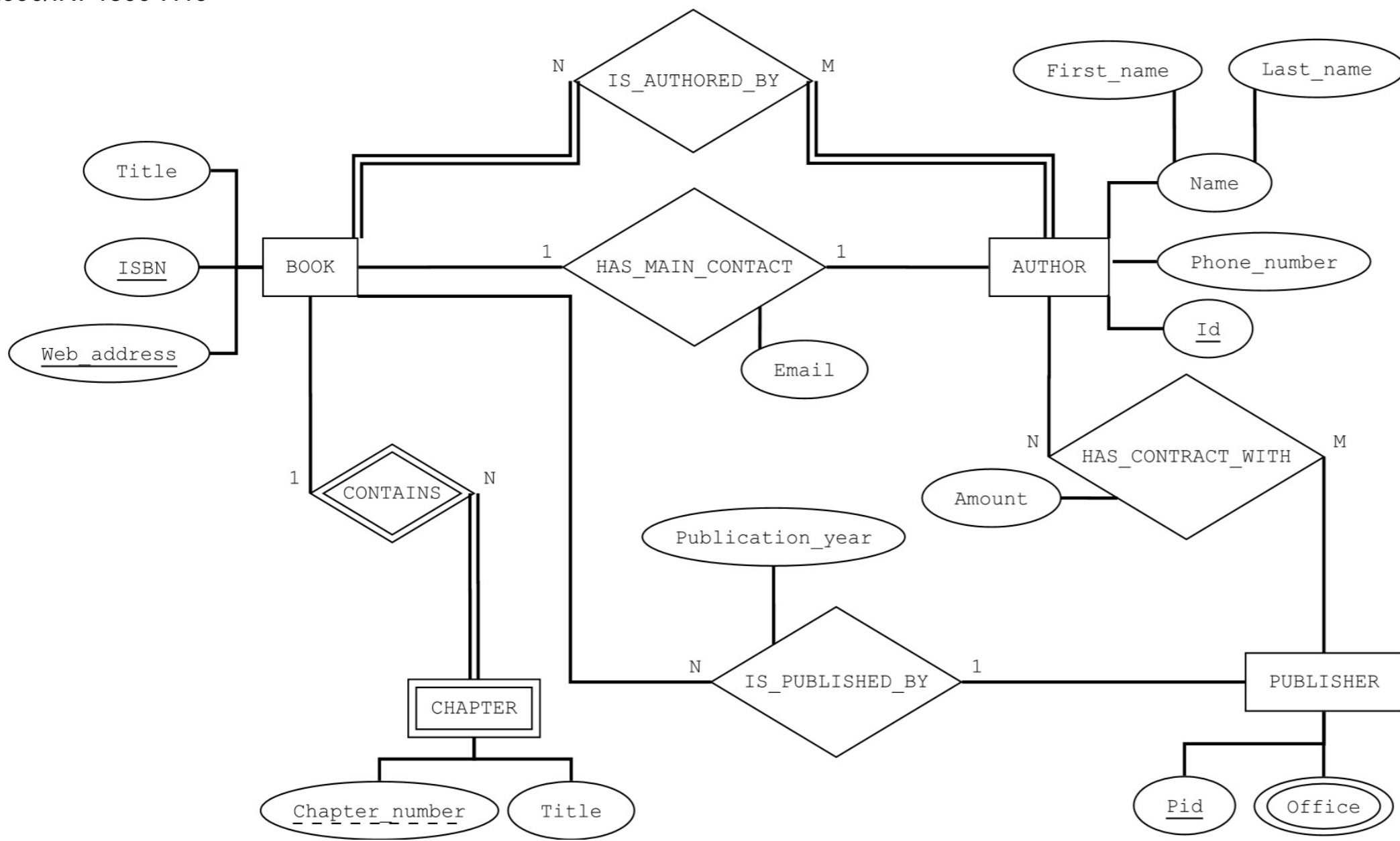
Maks poeng: 18

## 1.3 Realisering

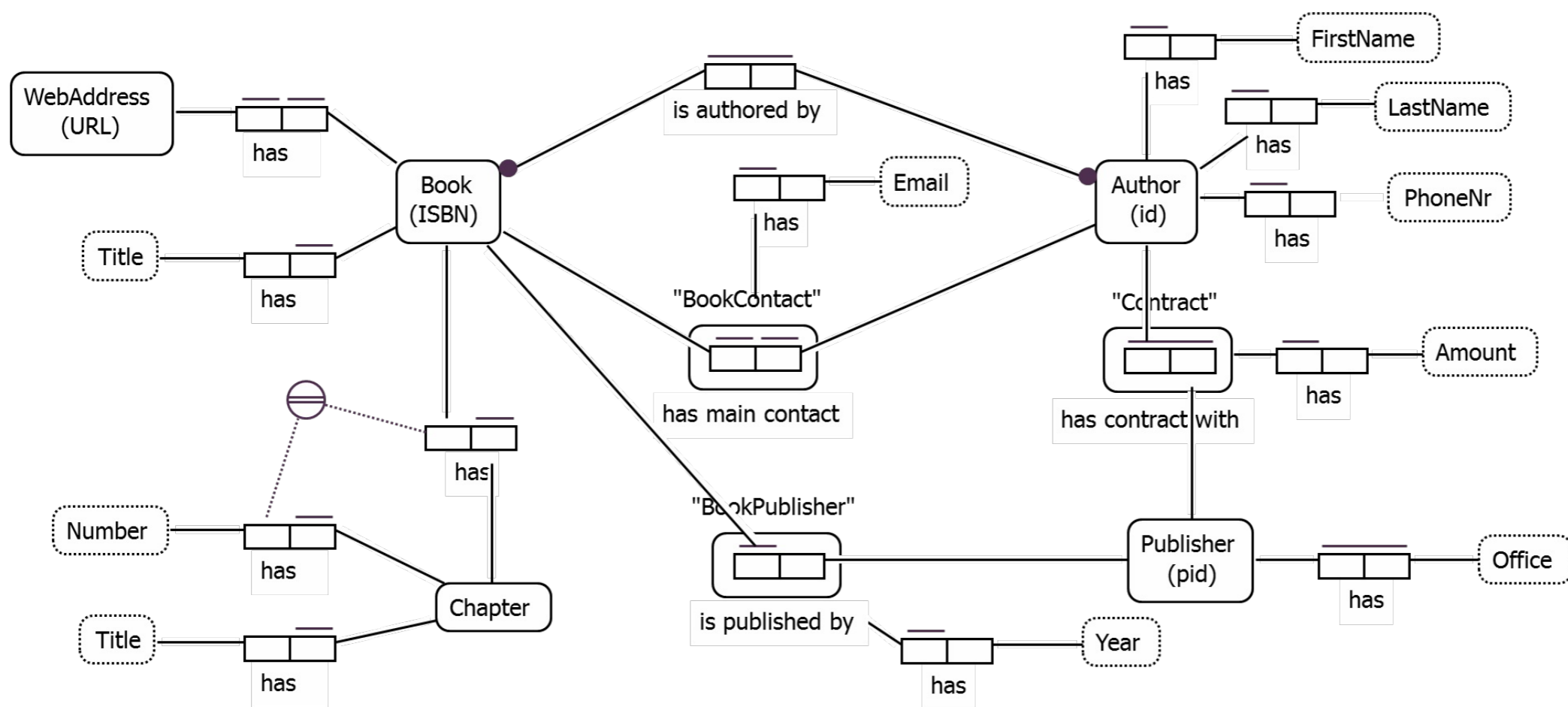
Realiser enten ER-modellen eller ORM2-modellen vist under til et databaseskjema ved å bruke algoritmen for realisering. Forklar eventuelle valg du tar underveis.

Bruk understreking for å markere kandidatnøkler. Bruk i tillegg **fet skrift** for å markere primærnøkkelen for hver relasjon (du finner knapper for understreking og fet skrift i menyen over). List også opp alle fremmednøkler på formen  $T(A) \rightarrow P(B)$  (her er attributten(e) A i T fremmednøkkel som referere til relasjon P sin(e) B attributt(er)).









ER-modell:



ORM2-modell:



**Skriv ditt svar her...**

Format - | **B** *I* U  $x_2$   $x^2$  |  $I_x$  |    |   |  $\Omega$   |  |  $\Sigma$  | 

Words: 0

Maks poeng: 15

## 2.1 Skranker og SQL

Gitt en database laget av følgende SQL-script:

```
CREATE TABLE art (
  aid int PRIMARY KEY,
  navn text NOT NULL,
  type text CHECK (type = 'Pattedyr' OR type = 'Fisk' OR type = 'Fugl')
);
```

```
CREATE TABLE dyr (
  did int PRIMARY KEY,
  navn text NOT NULL,
  mor_til int UNIQUE REFERENCES dyr(did),
  aid int REFERENCES art(aid)
);
```

```
INSERT INTO art VALUES
(0, 'Katt', 'Pattedyr'),
(1, 'Hund', 'Pattedyr'),
(2, 'Gris', 'Pattedyr'),
(3, 'Spurv', 'Fugl');
```

```
INSERT INTO dyr VALUES
(0, 'Doglas', NULL, 1),
(1, 'Mons', NULL, 0),
(2, 'Plutina', 0, 1),
(3, 'Princess', NULL, 2),
(4, 'Caterine', 1, 0);
```

For hver av SQL-kommandoene under, avgjør om de er lovlige i henhold til databaseskjemaet over (altså, kommandoen lykkes uten error) eller ulovlige i henhold til databaseskjemaet over (altså kommandoen feiler og gir error).

	Ulovlig	Lovlig
DROP TABLE art CASCADE;	<input type="radio"/>	<input type="radio"/>
INSERT INTO dyr VALUES (6, 'Timmy', NULL, 2);	<input type="radio"/>	<input type="radio"/>
UPDATE dyr SET mor_til = 3 WHERE did = 0;	<input type="radio"/>	<input type="radio"/>
UPDATE dyr SET mor_til = 4 WHERE mor_til IS NULL;	<input type="radio"/>	<input type="radio"/>
DELETE FROM dyr WHERE did = 3;	<input type="radio"/>	<input type="radio"/>
INSERT INTO art VALUES (4, 'Flue', 'Innsekt');	<input type="radio"/>	<input type="radio"/>
INSERT INTO dyr VALUES (7, 'Mona', 1, 0);	<input type="radio"/>	<input type="radio"/>
DELETE FROM dyr WHERE navn = 'Mons';	<input type="radio"/>	<input type="radio"/>

Maks poeng: 10

## 2.2 Band etter 2000

I denne (og de neste) oppgaven skal du bruke følgende databaseskjema:

Sjanger(sjangerID, navn)

Band(bandID, navn, startet, sjangerID)

Person(personID, navn, født)

Medlem(personID, bandID)

Album(albumID, bandID, navn, utgitt)

Sang(sangID, navn, spilletid, albumID)

med følgende fremmednøkler:

Band(sjangerID) refererer til Sjanger(sjangerID)

Meldem(personID) refererer til Person(personID)

Medlem(bandID) refererer til Band(bandID)

Album(bandID) refererer til Band(bandID)

Sang(albumID) refererer til Album(albumID)

En sjanger består av en unik sjangerID og et navn (f.eks. 'pop' eller 'metal'); et band har en unik bandID, et navn, en dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet spiller i; en person har en unik personID, et navn og en dato personen er født; personer kan være medlem i et band og er beksrevet i relasjonen Medlem (merk at en person kan være medlem i mange band og et band kan ha mange medlemmer); et album består av en unik albumID, et navn, bandIDen til bandet som lagde albummet, og en dato for når albummet ble utgitt; en sang har en unik sangID, et navn, en spilletid i sekunder, og en albumID som sier hvilket album sangen er en del av.

For eksempel kan databasen inneholde følgende data:

Sjanger

sjangerID	Navn
0	Pop
1	Rock
2	Metal

Band

bandID	navn	startet	sjangerID
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violet	1989-04-23	1

Person

personID	navn	født
0	Peter Smith	1963-02-09
1	Mary Green	1978-08-16
2	Bettany Evans	1981-09-09

Medlem

personID	bandID
0	0
1	2
2	1
1	0

Album

albumID	bandID	navn	utgitt
0	0	Surfin	1986-02-07
1	1	Love	1999-11-09
2	0	Fog on the grass	1990-03-13

Sang

sangID	navn	spilletid	albumID
0	Board	231	0
1	Miss you	126	1
2	Sharks around	322	0
3	Fog on the grass	401	2
4	Biking in the sun	209	2

**Oppgave:** Skriv en spørring som finner alle band som enten ble startet etter år 2000 eller inneholder strengen 'King' i navnet. Skriv ut navnet på bandet og datoen bandet ble startet.

**Skriv ditt svar her...**

1

Maks poeng: 5

## 2.3 Timer Pop-musikk fra 90s

I denne (og de neste) oppgaven skal du bruke følgende databaseskjema:

Sjanger(sjangerID, navn)

Band(bandID, navn, startet, sjangerID)

Person(personID, navn, født)

Medlem(personID, bandID)

Album(albumID, bandID, navn, utgitt)

Sang(sangID, navn, spilletid, albumID)

med følgende fremmednøkler:

Band(sjangerID) refererer til Sjanger(sjangerID)

Medlem(personID) refererer til Person(personID)

Medlem(bandID) refererer til Band(bandID)

Album(bandID) refererer til Band(bandID)

Sang(albumID) refererer til Album(albumID)

En sjanger består av en unik sjangerID og et navn (f.eks. 'pop' eller 'metal'); et band har en unik bandID, et navn, en dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet spiller i; en person har en unik personID, et navn og en dato personen er født; personer kan være medlem i et band og er



beksrevet i relasjonen Medlem (merk at en person kan være medlem i mange band og et band kan ha mange medlemmer); et album består av en unik albumID, et navn, bandIDen til bandet som lagde albummet, og en dato for når albummet ble utgitt; en sang har en unik sangID, et navn, en spilletid i sekunder, og en albumID som sier hvilket album sangen er en del av.

For eksempel kan databasen inneholde følgende data:

## Sjanger

sjangerID	Navn
0	Pop
1	Rock
2	Metal

## Band

bandID	navn	startet	sjangerID
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violet	1989-04-23	1

## Person

personID	navn	født
0	Peter Smith	1963-02-09
1	Mary Green	1978-08-16
2	Bettany Evans	1981-09-09

## Medlem

personID	bandID
0	0
1	2
2	1
1	0

## Album

albumID	bandID	navn	utgitt
0	0	Surfin	1986-02-07
1	1	Love	1999-11-09
2	0	Fog on the grass	1990-03-13

## Sang

sangID	navn	spilletid	albumID
0	Board	231	0
1	Miss you	126	1
2	Sharks around	322	0
3	Fog on the grass	401	2
4	Biking in the sun	209	2

**Oppgave:** Skriv en spørring som finner antall timer med musikk fra sjangeren 'Pop' laget av band startet mellom år 1990 og 2000. Merk: En time er 3600 sekunder.

Skriv ditt svar her...

1	
---	--

Maks poeng: 5

## 2.4 Personer født på interessant dato

I denne (og de neste) oppgaven skal du bruke følgende databaseskjema:

Sjanger(sjangerID, navn)

Band(bandID, navn, startet, sjangerID)

Person(personID, navn, født)

Medlem(personID, bandID)

Album(albumID, bandID, navn, utgitt)

Sang(sangID, navn, spilletid, albumID)

med følgende fremmednøkler:

Band(sjangerID) refererer til Sjanger(sjangerID)

Medlem(personID) refererer til Person(personID)

Medlem(bandID) refererer til Band(bandID)

Album(bandID) refererer til Band(bandID)

Sang(albumID) refererer til Album(albumID)

En sjanger består av en unik sjangerID og et navn (f.eks. 'pop' eller 'metal'); et band har en unik bandID, et navn, en dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet spiller i; en person har en unik personID, et navn og en dato personen er født; personer kan være medlem i et band og er beksrevet i relasjonen Medlem (merk at en person kan være medlem i mange band og et band kan ha mange medlemmer); et album består av en unik albumID, et navn, bandIDen til bandet som lagde albummet, og en dato for når albummet ble utgitt; en sang har en unik sangID, et navn, en spilletid i sekunder, og en albumID som sier hvilket album sangen er en del av.

For eksempel kan databasen inneholde følgende data:

Sjanger	
sjangerID	Navn
0	Pop
1	Rock
2	Metal

## Band

bandID	navn	startet	sjangerID
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violet	1989-04-23	1

## Person

personID	navn	født
0	Peter Smith	1963-02-09
1	Mary Green	1978-08-16
2	Bettany Evans	1981-09-09

## Medlem

personID	bandID
0	0
1	2
2	1
1	0

## Album

albumID	bandID	navn	utgitt
0	0	Surfin	1986-02-07
1	1	Love	1999-11-09
2	0	Fog on the grass	1990-03-13

## Sang

sangID	navn	spilletid	albumID
0	Board	231	0
1	Miss you	126	1
2	Sharks around	322	0
3	Fog on the grass	401	2
4	Biking in the sun	209	2

**Oppgave:** Skriv en spørring som finner navnet på alle personer født på en dato hvor det enten ble startet et nytt band, eller ble gitt ut et nytt album.

Skriv ditt svar her...

1	
---	--

Maks poeng: 5

## 2.5 Sanger per band

I denne (og de neste) oppgaven skal du bruke følgende databaseskjema:

Sjanger(sjangerID, navn)

Band(bandID, navn, startet, sjangerID)

Person(personID, navn, født)

Medlem(personID, bandID)

Album(albumID, bandID, navn, utgitt)

Sang(sangID, navn, spilletid, albumID)

med følgende fremmednøkler:

Band(sjangerID) refererer til Sjanger(sjangerID)

Medlem(personID) refererer til Person(personID)

Medlem(bandID) refererer til Band(bandID)

Album(bandID) refererer til Band(bandID)

Sang(albumID) refererer til Album(albumID)

En sjanger består av en unik sjangerID og et navn (f.eks. 'pop' eller 'metal'); et band har en unik bandID, et navn, en dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet spiller i; en person har en unik personID, et navn og en dato personen er født; personer kan være medlem i et band og er baksrevet i relasjonen Medlem (merk at en person kan være medlem i mange band og et band kan ha mange medlemmer); et album består av en unik albumID, et navn, bandIDen til bandet som lagde albummet, og en dato for når albummet ble utgitt; en sang har en unik sangID, et navn, en spilletid i sekunder, og en albumID som sier hvilket album sangen er en del av.

For eksempel kan databasen inneholde følgende data:

Sjanger	
sjangerID	Navn
0	Pop
1	Rock
2	Metal

## Band

bandID	navn	startet	sjangerID
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violet	1989-04-23	1

## Person

personID	navn	født
0	Peter Smith	1963-02-09
1	Mary Green	1978-08-16
2	Bettany Evans	1981-09-09

## Medlem

personID	bandID
0	0
1	2
2	1
1	0

## Album

albumID	bandID	navn	utgitt
0	0	Surfin	1986-02-07
1	1	Love	1999-11-09
2	0	Fog on the grass	1990-03-13

## Sang

sangID	navn	spilletid	albumID
0	Board	231	0
1	Miss you	126	1
2	Sharks around	322	0
3	Fog on the grass	401	2
4	Biking in the sun	209	2

**Oppgave:** Skriv en spørring som finner antall sanger hvert band har laget (altså antall sanger på alle deres album til sammen) for band som har laget færre enn 3 sanger. Skriv ut bandIDen, navnet på bandet og antall sanger.

Merk: Det kan finnes band som ikke har gitt ut noen album ennå eller band som kun har utgitt album uten sanger, disse skal også med i resultatet med antall sanger lik 0.

Skriv ditt svar her...

1	
---	--

Maks poeng: 5

## 2.6 Slett tomme album

I denne (og de neste) oppgaven skal du bruke følgende databaseskjema:

Sjanger(sjangerID, navn)

Band(bandID, navn, startet, sjangerID)

Person(personID, navn, født)

Medlem(personID, bandID)

Album(albumID, bandID, navn, utgitt)

Sang(sangID, navn, spilletid, albumID)

med følgende fremmednøkler:

Band(sjangerID) refererer til Sjanger(sjangerID)

Medlem(personID) refererer til Person(personID)

Medlem(bandID) refererer til Band(bandID)

Album(bandID) refererer til Band(bandID)

Sang(albumID) refererer til Album(albumID)

En sjanger består av en unik sjangerID og et navn (f.eks. 'pop' eller 'metal'); et band har en unik bandID, et navn, en dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet spiller i; en person har en unik personID, et navn og en dato personen er født; personer kan være medlem i et band og er baksrevet i relasjonen Medlem (merk at en person kan være medlem i mange band og et band kan ha mange medlemmer); et album består av en unik albumID, et navn, bandIDen til bandet som lagde albummet, og en dato for når albummet ble utgitt; en sang har en unik sangID, et navn, en spilletid i sekunder, og en albumID som sier hvilket album sangen er en del av.

For eksempel kan databasen inneholde følgende data:

Sjanger	
sjangerID	Navn
0	Pop
1	Rock
2	Metal

## Band

bandID	navn	startet	sjangerID
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violet	1989-04-23	1

## Person

personID	navn	født
0	Peter Smith	1963-02-09
1	Mary Green	1978-08-16
2	Bettany Evans	1981-09-09

## Medlem

personID	bandID
0	0
1	2
2	1
1	0

## Album

albumID	bandID	navn	utgitt
0	0	Surfin	1986-02-07
1	1	Love	1999-11-09
2	0	Fog on the grass	1990-03-13

## Sang

sangID	navn	spilletid	albumID
0	Board	231	0
1	Miss you	126	1
2	Sharks around	322	0
3	Fog on the grass	401	2
4	Biking in the sun	209	2

**Oppgave:** Skrive en SQL-kommando som sletter alle album som ikke har noen tilhørende sanger.

**Skriv ditt svar her...**

1	
---	--

## 2.7 Nyeste album

I denne (og de neste) oppgaven skal du bruke følgende databaseskjema:

Sjanger(sjangerID, navn)

Band(bandID, navn, startet, sjangerID)

Person(personID, navn, født)

Medlem(personID, bandID)

Album(albumID, bandID, navn, utgitt)

Sang(sangID, navn, spilletid, albumID)

med følgende fremmednøkler:

Band(sjangerID) refererer til Sjanger(sjangerID)

Medlem(personID) refererer til Person(personID)

Medlem(bandID) refererer til Band(bandID)

Album(bandID) refererer til Band(bandID)

Sang(albumID) refererer til Album(albumID)

En sjanger består av en unik sjangerID og et navn (f.eks. 'pop' eller 'metal'); et band har en unik bandID, et navn, en dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet spiller i; en person har en unik personID, et navn og en dato personen er født; personer kan være medlem i et band og er beksrevet i relasjonen Medlem (merk at en person kan være medlem i mange band og et band kan ha mange medlemmer); et album består av en unik albumID, et navn, bandIDen til bandet som lagde albummet, og en dato for når albummet ble utgitt; en sang har en unik sangID, et navn, en spilletid i sekunder, og en albumID som sier hvilket album sangen er en del av.

For eksempel kan databasen inneholde følgende data:

Sjanger

sjangerID	Navn
0	Pop
1	Rock
2	Metal

Band

bandID	navn	startet	sjangerID
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violet	1989-04-23	1

Person

personID	navn	født
0	Peter Smith	1963-02-09
1	Mary Green	1978-08-16
2	Bettany Evans	1981-09-09

Medlem

personID	bandID
0	0
1	2
2	1
1	0

...



Album

albumID	bandID	navn	utgitt
0	0	Surfin	1986-02-07
1	1	Love	1999-11-09
2	0	Fog on the grass	1990-03-13

Sang

sangID	navn	spilletid	albumID
0	Board	231	0
1	Miss you	126	1
2	Sharks around	322	0
3	Fog on the grass	401	2
4	Biking in the sun	209	2

**Oppgave:** Skriv en SQL-kommando som lager et view med navn "nyeste\_album" som inneholder de 10 nyeste albummene. Viewet skal for hvert av albummene vise albummets navn og navnet på bandet som utga albummet, datoen albummet er gitt ut, samt antall sanger på albummet. Sorter viewet etter når albummet er utgitt, hvor det nyeste er først.

(Merk: Du kan her anta at alle album har minst én sang)

**Skriv ditt svar her...**

1

Maks poeng: 5

## 2.8 Super-album

I denne (og de neste) oppgaven skal du bruke følgende databaseskjema:

Sjanger(sjangerID, navn)

Band(bandID, navn, startet, sjangerID)

Person(personID, navn, født)

Medlem(personID, bandID)

Album(albumID, bandID, navn, utgitt)

Sang(sangID, navn, spilletid, albumID)

med følgende fremmednøkler:

Band(sjangerID) refererer til Sjanger(sjangerID)  
 Meldem(personID) refererer til Person(personID)  
 Medlem(bandID) refererer til Band(bandID)  
 Album(bandID) refererer til Band(bandID)  
 Sang(albumID) refererer til Album(albumID)

En sjanger består av en unik sjangerID og et navn (f.eks. 'pop' eller 'metal'); et band har en unik bandID, et navn, en dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet spiller i; en person har en unik personID, et navn og en dato personen er født; personer kan være medlem i et band og er beksrevet i relasjonen Medlem (merk at en person kan være medlem i mange band og et band kan ha mange medlemmer); et album består av en unik albumID, et navn, bandIDen til bandet som lagde albummet, og en dato for når albummet ble utgitt; en sang har en unik sangID, et navn, en spilletid i sekunder, og en albumID som sier hvilket album sangen er en del av.

For eksempel kan databasen inneholde følgende data:

#### Sjanger

sjangerID	Navn
0	Pop
1	Rock
2	Metal

#### Band

bandID	navn	startet	sjangerID
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violet	1989-04-23	1

#### Person

personID	navn	født
0	Peter Smith	1963-02-09
1	Mary Green	1978-08-16
2	Bettany Evans	1981-09-09

#### Medlem

personID	bandID
0	0
1	2
2	1
1	0

#### Album

albumID	bandID	navn	utgitt
0	0	Surfin	1986-02-07
1	1	Love	1999-11-09
2	0	Fog on the grass	1990-03-13

#### Sang

sangID	navn	spilletid	albumID
0	Board	231	0
1	Miss you	126	1
2	Sharks around	322	0
3	Fog on the grass	401	2
4	Biking in the sun	209	2

**Oppgave:** Skriv navnet på alle band som har gitt ut et super-album. Et super-album er et album med spilletid

på mer enn en time (3600 sekunder). Skriv ut bandIDen, navnet på bandet, samt antall super-album bandet har utgitt.

Hint: Det kan være lurt å først finne "albumID" til alle super-album i en egen spørring (f.eks. med WITH). Deretter kan man finne bandet som har laget albummene, og telle opp.

**Skriv ditt svar her...**

1	
---	--

Maks poeng: 10

### 3.1 FDer

Gitt følgende relasjon:

$R(A, B, C, D, E, F, G)$

og følgende funksjonelle avhengigheter (FDer):

$A \rightarrow C$

$C \rightarrow B, D$

$A, B \rightarrow D, F$

$B, E \rightarrow G$

Huk av attributtene som er med i tillukningen til {A}.

- B
- E
- G
- F
- A
- C
- D

**Huk av attributtene som er med i tillukningen til {B}.**

- D
- F
- G
- B
- C
- A
- E

**Huk av attributtene som må være med i alle kandidatnøkler.**

- C
- D
- F
- G
- E
- B
- A

---

Maks poeng: 6

## 3.2 Normalformer

Gitt følgende relasjon

$R(\underline{A}, B, C, D, E, F)$

hvor kandidatnøkkelen er understreket. (Vi har altså kun én kandidatnøkkel,  $\{A, B\}$ )

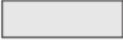






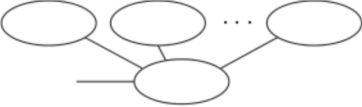




For hver av FDene under, anta at FDen gjelder for relasjonen R over, og bruk algoritmen for normalformer til å avgjøre hvilken normalform FDen (alene) tilsier at R er på.

	3NF	2NF	BCNF	1NF
A -> C	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
B -> D	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
A, B -> F	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
A, D -> E	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Maks poeng: 4

**Question 1**  
Attached



Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of $E_2$ in $R$
	Cardinality Ratio 1: N for $E_1 : E_2$ in $R$
	Structural Constraint (min, max) on Participation of $E$ in $R$

**Figure 3.14**  
Summary of the notation for ER diagrams.

# ORM 2 Graphical Notation

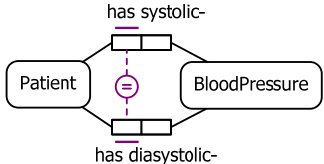
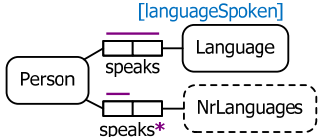
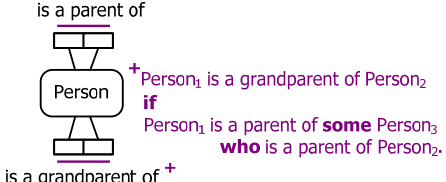
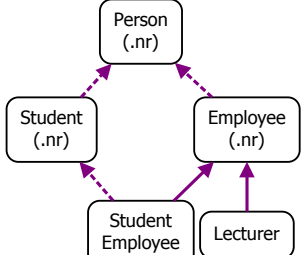
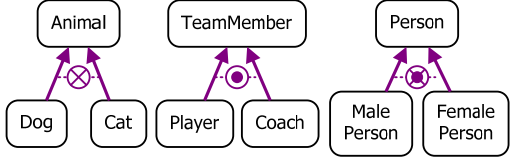
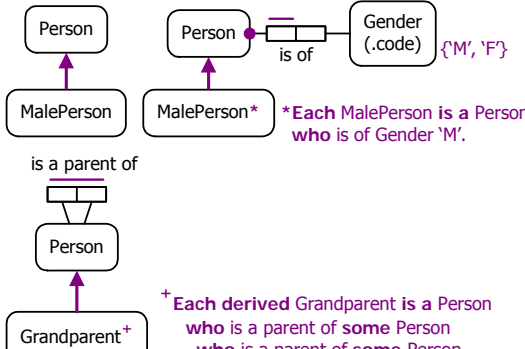
Terry Halpin

Construct	Examples	Description/Notes
Entity Type		Named soft rectangle, named hard rectangle, or named ellipse. The soft rectangle shape is the default.
Value Type		Named, dashed, soft rectangle (or hard rectangle or ellipse).
Entity type with popular reference mode	 	Abbreviation for injective reference relationship to value type, e.g.
Entity type with unit-based reference mode	 	Abbreviation for reference type, e.g. Optionally, unit type may be displayed.
Entity type with general reference mode	 	Abbreviation for reference type, e.g.
Independent Object Type		Instances of the type may exist, without playing any elementary fact roles
External Object Type		This notation is tentative (yet to be finalized)
Predicate (unary, binary, ternary, etc.)		Ordered set of 1 or more role boxes with at least one predicate reading in mixfix notation. If shown, object placeholders are denoted by "...". If placeholders are not shown, unaries are in prefix and binaries are in infix notation.
Duplicate type or predicate shape		If an object type or predicate shape is displayed more than once (on the same page or different pages) it is shadowed.
Unary fact type		The smokes role may be played by instances of the Person object type
Binary fact type		By default, predicate readings (binary or longer) are read left-to-right or top-to-bottom. An arrow-tip is used to display a different reading direction. Role names may be displayed in square brackets beside their role. Forward and inverse readings for binaries may be shown together, separated by "/".


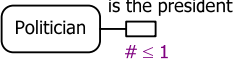


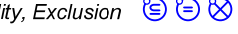










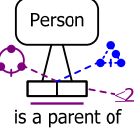
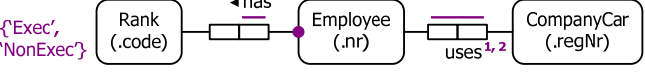
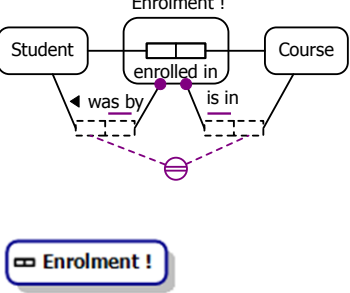


Construct	Examples	Description/Notes
Ternary fact type	<p>Person [player] Sport Country ... played ... for ...  Person ... introduced ... to ...  Date Food Cat ... ate ... on ...  [Cat] ate [Food] on [Date]</p>	<p>Role names may be added in square brackets.</p> <p>Arrow-tips are used to reverse the default left-right or top-down reading order.</p> <p>Reading orders other than forward and reverse are shown using named placeholders.</p>
Quaternary fact type	<p>Person City Date Food ... in ... on .. ate ...</p>	<p>The above notes for the ternary case apply here also.</p> <p>Fact types of higher arity (number of roles) are also permitted.</p>
Objectification (a.k.a. nesting)	<p>Student enrolled in Course  "Enrolment!"  resulted in Grade</p>	<p>The enrolment fact type is objectified as an entity type whose instances can play roles.</p> <p>In this example, the objectification type is independent, so we can know about an enrolment before the grade is obtained.</p>
Internal uniqueness constraint (UC) on unaries	<p>Person smokes Person smokes</p>	<p>These are equivalent (by default, predicates are assumed to be populated with sets, so no whole fact may be duplicated).</p>
Internal UC on binaries	<p>Gender is of Person was born in Country  Language speaks Person is president of Country</p>	<p>The examples show the 4 possible patterns:</p> <p>1:n (one-to-many); n:1 (many-to-one); m:n (many-to-many); 1:1 (one-to-one)</p>
Internal UC on ternaries.  For n-aries (n > 1) each UC must span at least n-1 roles	<p>Team got in Competition  Person played for Country</p>	<p>The first example has two, 2-role UCs: the top UC forbids ties; the other UC ensures that each team gets only place per competition (a dotted line excludes its role from the UC).</p> <p>The second example has a spanning UC (many-to-many-to-many).</p>
Simple mandatory role constraint	<p>Person was born in Country  Person was born in Country</p>	<p>The example constraint means that each person was born in some country.</p> <p>The mandatory role dot may be placed at either end of the role connector.</p>
Inclusive-or constraint (disjunctive mandatory role)	<p>Visitor has Passport  Visitor has DriverLicence</p>	<p>The constraint is displayed as a circled dot connected to the constrained roles. The example constraint means that each visitor referenced in the model must have a passport or a driver licence (or both).</p>
Preferred internal UC	<p>Country has / is of CountryCode</p>	<p>A double bar on a UC indicates it underlies the preferred reference scheme.</p>

Construct	Examples	Description/Notes
External UC (double-bar indicates preferred identifier)		Here, each state is primarily identified by combining its country and state code. Each combination of country and state name also applies to only one state.
Object Type Value Constraint		<p><i>Enumerations</i></p> <p>Ranges are inclusive of end values by default. Round brackets are used to exclude an end value. Square brackets may be added to explicitly declare inclusion, e.g. the constraint on PositiveScore may also be specified as {(0..100]}.</p> <p>Multiple combinations are allowed.</p>
Role value constraint		As for object type value constraints, but connected to the constrained role. Here, an age of a person must be at most 140 years.
Subset constraint		The arrow points from the subset end to the superset end (e.g. if a person smokes then that person is cancer prone). The role sequences at both ends must be compatible. A connection to the junction of 2 roles constrains that role pair.
Join subset constraint		The constrained role pair at the superset end is projected from a role path that involves a conceptual join on Language. The constraint declares that if an advisor serves in a country then that advisor must speak a language that is often used in that country.
Exclusion constraint		These constraints mean that no person is both married and widowed, and no person reviewed and authored the same book. Exclusion may apply between 2 or more compatible role sequences, possibly involving joins.
Exclusive-or constraint		An exclusive-or constraint is simply the conjunction of an inclusive-or constraint and an exclusion constraint. Also known as an xor constraint.

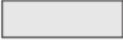








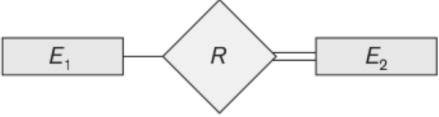

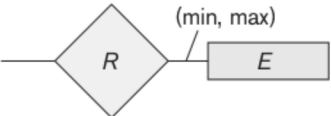
Construct	Examples	Description/Notes
Equality constraint		<p>This constraint means that a patient's systolic BP is recorded if and only if his/her diastolic BP is recorded.</p> <p>An equality constraint may apply between 2 or more compatible role sequences, possibly involving joins.</p>
Derived fact type, and derivation rule	 <p><b>*For each Person,</b> nrLanguages = count(languageSpoken).</p>	<p>A fact type is either asserted, derived, or semiderived.</p> <p>A derived fact type is marked with an asterisk <b>"*"</b>. A derivation rule is supplied. A double asterisk <b>"**"</b> indicates derived and stored (eager evaluation).</p>
Semiderived fact type, and derivation rule	 <p><b>+Person<sub>1</sub> is a grandparent of Person<sub>2</sub></b> <b>if</b> <b>Person<sub>1</sub> is a parent of some Person<sub>3</sub></b> <b>who is a parent of Person<sub>2</sub>.</b></p>	<p>A fact type is semiderived if some of its instances may be derived, and some of its instances may be simply asserted.</p> <p>It is marked by <b>"+"</b> (half an asterisk). <b>"**"</b> indicates semiderived and stored (eager evaluation for derived instances).</p>
Subtyping		<p>All subtypes are proper subtypes. An arrow runs from subtype to supertype. A solid arrow indicates a path to the subtype's preferred identifier (e.g. here, student employees are primarily identified by their employee number). A dashed arrow indicates the supertype has a different preferred identifier.</p>
Subtyping constraints		<p>A circled <b>"X"</b> indicates the subtypes are mutually exclusive. A circled dot indicates the supertype equals the union of the subtypes. The combination (xor constraint) indicates the subtypes partition the supertype (exclusive and exhaustive).</p>
Subtype derivation status	 <p><b>*Each MalePerson is a Person</b> <b>who is of Gender 'M'.</b></p> <p><b>+ Each derived Grandparent is a Person</b> <b>who is a parent of some Person</b> <b>who is a parent of some Person.</b></p>	<p>A subtype may be</p> <ul style="list-style-type: none"> <li>• asserted,</li> <li>• derived (denoted by <b>"*"</b>),</li> <li>• or semiderived (denoted by <b>"+"</b>).</li> </ul> <p>If the subtype is asserted, it has no mark appended and has no derivation rule.</p> <p>If the subtype derived or semiderived, a derivation rule is supplied.</p>

Construct	Examples	Description/Notes
<p>Internal frequency constraint</p>		<p>This constrains the number of times an occurring instance of a role or role sequence may appear in each population. Here: each jury has exactly 12 members; each panel that includes an expert includes at least 4 and at most 7 experts; each expert reviews at most 5 papers; each paper that is reviewed is reviewed by at least 2 experts; and each department and year that has staff numbers recorded in the quaternary appears there twice (once for each gender).</p>
<p>External frequency constraint</p>		<p>The example constraint has the following meaning. In this context, each combination of student and course relates to at most two enrolments (i.e. a student may enroll at most twice in the same course)</p>
<p>Ring constraints</p>		<p>A ring predicate <math>R</math> is locally reflexive if and only if, for all <math>x</math> and <math>y</math>, <math>xRy</math> implies <math>xRx</math>. E.g. “knows” is locally but not globally reflexive.</p> <p>Reflexive, symmetric and transitive properties may also be enforced using semiderivation rather than by constraining asserted fact types.</p> <p>The example constrains the subtyping relationship in ORM to be both acyclic (no cycles can be formed by a chain of subtyping connections) and strongly intransitive (no object type <math>A</math> can be both a direct subtype of another type <math>B</math> and an indirect subtype of <math>B</math>, where indirect subtyping means there is a chain of two or more subtyping relationships that lead from <math>A</math> to <math>B</math>).</p> <p>Ring constraints may be combined only if they are compatible, and one is not implied by the other. ORM tools ensure that only legal combinations are allowed.</p>
<p>Value-comparison constraints</p>		<p>The example constraint verbalizes as:  <b>For each Project,</b>  <b>existing</b> enddate <math>\geq</math> startdate.</p>

Construct	Examples	Description/Notes
Object cardinality constraint		The example constraints ensure there is exactly one president and at most 100 senators (at any given time),
Role cardinality constraint		The example constraint ensures that at most one politician plays the role of president (at any given time).
Deontic constraints	<p>Uniqueness </p> <p>Mandatory </p> <p>Subset, Equality, Exclusion </p> <p>Frequency </p> <p>Irreflexive  Acyclic </p> <p>Asymmetric  Asym-Intrans </p> <p>Intransitive  Acyclic-Intrans </p> <p>Antisymmetric  Symmetric </p> <p>Strongly Intransitive  etc.</p> <p>e.g.</p> 	<p>Unlike alethic constraints, deontic constraint shapes are colored blue rather than violet. Most include “o” for “obligatory”. Deontic ring constraints instead use dashed lines.</p> <p>In the parenthood example, the alethic frequency constraint ensures that each person has at most two parents, the alethic ring constraint ensures that parenthood is acyclic, and the deontic ring constraint makes it obligatory for parenthood to be strongly intransitive.</p>
Textual constraints	 <p><sup>1</sup> Each Employee who has Rank 'NonExec' uses at most one CompanyCar.  <sup>2</sup> Each Employee who has Rank 'Exec' uses some CompanyCar.</p>	First-order constraints with no graphic notation may be expressed textually in the FORML 2 language. These examples use footnoting to capture a restricted uniqueness constraint and a restricted mandatory role constraint.
Objectification display options: link fact types, and compact display.		Internally, link fact types connect objectified associations to their component object types. By default, display of link fact types is suppressed. If displayed, link predicate shapes use dashed lines instead of solid lines. Objectification object types may also be displayed without their defining components, using an object type shape containing a small predicate shape, as shown in this Enrolment example.

**Question 2**  
Attached



Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of $E_2$ in $R$
	Cardinality Ratio 1 : N for $E_1 : E_2$ in $R$
	Structural Constraint (min, max) on Participation of $E$ in $R$

**Figure 3.14**  
Summary of the notation for ER diagrams.

# ORM 2 Graphical Notation

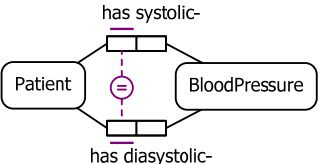
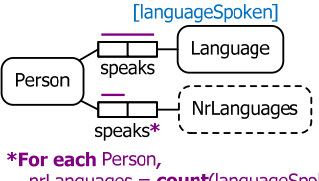
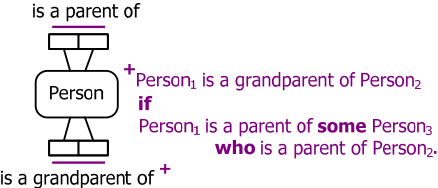
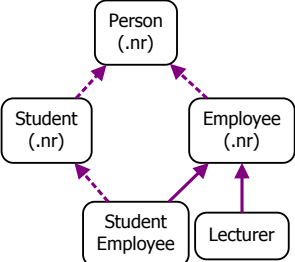
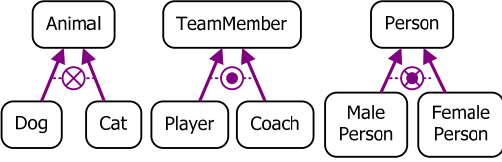
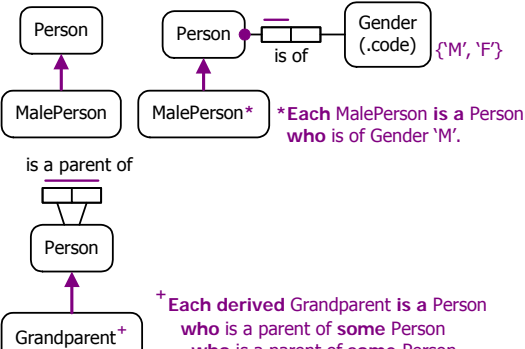
Terry Halpin

Construct	Examples	Description/Notes
Entity Type		Named soft rectangle, named hard rectangle, or named ellipse. The soft rectangle shape is the default.
Value Type		Named, dashed, soft rectangle (or hard rectangle or ellipse).
Entity type with popular reference mode	 	Abbreviation for injective reference relationship to value type, e.g.
Entity type with unit-based reference mode	 	Abbreviation for reference type, e.g. Optionally, unit type may be displayed.
Entity type with general reference mode	 	Abbreviation for reference type, e.g.
Independent Object Type		Instances of the type may exist, without playing any elementary fact roles
External Object Type		This notation is tentative (yet to be finalized)
Predicate (unary, binary, ternary, etc.)		Ordered set of 1 or more role boxes with at least one predicate reading in mixfix notation. If shown, object placeholders are denoted by "...". If placeholders are not shown, unaries are in prefix and binaries are in infix notation.
Duplicate type or predicate shape		If an object type or predicate shape is displayed more than once (on the same page or different pages) it is shadowed.
Unary fact type		The smokes role may be played by instances of the Person object type
Binary fact type		By default, predicate readings (binary or longer) are read left-to-right or top-to-bottom. An arrow-tip is used to display a different reading direction. Role names may be displayed in square brackets beside their role. Forward and inverse readings for binaries may be shown together, separated by "/".


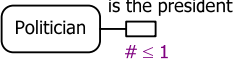


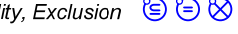










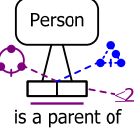
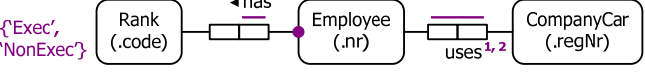
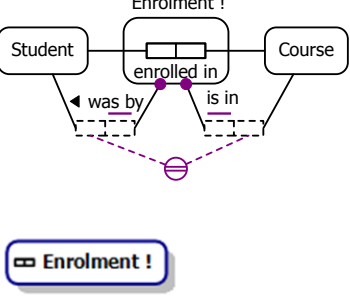


Construct	Examples	Description/Notes
Ternary fact type	<p>Person [player] Sport Country ... played ... for ...  Person ... introduced ... to ...  Date Food Cat ... ate ... on ...  [Cat] ate [Food] on [Date]</p>	<p>Role names may be added in square brackets.</p> <p>Arrow-tips are used to reverse the default left-right or top-down reading order.</p> <p>Reading orders other than forward and reverse are shown using named placeholders.</p>
Quaternary fact type	<p>Person City Date Food ... in ... on .. ate ...</p>	<p>The above notes for the ternary case apply here also.</p> <p>Fact types of higher arity (number of roles) are also permitted.</p>
Objectification (a.k.a. nesting)	<p>"Enrolment !"  Student enrolled in Course  resulted in Grade</p>	<p>The enrolment fact type is objectified as an entity type whose instances can play roles.</p> <p>In this example, the objectification type is independent, so we can know about an enrolment before the grade is obtained.</p>
Internal uniqueness constraint (UC) on unaries	<p>Person smokes</p>	<p>These are equivalent (by default, predicates are assumed to be populated with sets, so no whole fact may be duplicated).</p>
Internal UC on binaries	<p>Gender is of Person was born in Country  Language speaks Person is president of Country</p>	<p>The examples show the 4 possible patterns:</p> <p>1:n (one-to-many); n:1 (many-to-one); m:n (many-to-many); 1:1 (one-to-one)</p>
Internal UC on ternaries.  For n-aries (n > 1) each UC must span at least n-1 roles	<p>Team got in Competition  Person played for Country</p>	<p>The first example has two, 2-role UCs: the top UC forbids ties; the other UC ensures that each team gets only place per competition (a dotted line excludes its role from the UC).</p> <p>The second example has a spanning UC (many-to-many-to-many).</p>
Simple mandatory role constraint	<p>Person was born in Country</p>	<p>The example constraint means that each person was born in some country.</p> <p>The mandatory role dot may be placed at either end of the role connector.</p>
Inclusive-or constraint (disjunctive mandatory role)	<p>Visitor has Passport DriverLicence</p>	<p>The constraint is displayed as a circled dot connected to the constrained roles. The example constraint means that each visitor referenced in the model must have a passport or a driver licence (or both).</p>
Preferred internal UC	<p>Country has / is of CountryCode</p>	<p>A double bar on a UC indicates it underlies the preferred reference scheme.</p>

Construct	Examples	Description/Notes
External UC (double-bar indicates preferred identifier)		Here, each state is primarily identified by combining its country and state code. Each combination of country and state name also applies to only one state.
Object Type Value Constraint		Enumerations
		Ranges are inclusive of end values by default. Round brackets are used to exclude an end value. Square brackets may be added to explicitly declare inclusion, e.g. the constraint on PositiveScore may also be specified as {(0..100]}.
		Multiple combinations are allowed.
Role value constraint		As for object type value constraints, but connected to the constrained role. Here, an age of a person must be at most 140 years.
Subset constraint		The arrow points from the subset end to the superset end (e.g. if a person smokes then that person is cancer prone). The role sequences at both ends must be compatible. A connection to the junction of 2 roles constrains that role pair.
Join subset constraint		The constrained role pair at the superset end is projected from a role path that involves a conceptual join on Language. The constraint declares that if an advisor serves in a country then that advisor must speak a language that is often used in that country.
Exclusion constraint		These constraints mean that no person is both married and widowed, and no person reviewed and authored the same book. Exclusion may apply between 2 or more compatible role sequences, possibly involving joins.
Exclusive-or constraint		An exclusive-or constraint is simply the conjunction of an inclusive-or constraint and an exclusion constraint. Also known as an xor constraint.

Construct	Examples	Description/Notes
Equality constraint		<p>This constraint means that a patient's systolic BP is recorded if and only if his/her diastolic BP is recorded.</p> <p>An equality constraint may apply between 2 or more compatible role sequences, possibly involving joins.</p>
Derived fact type, and derivation rule	 <p><b>*For each Person,</b> nrLanguages = count(languageSpoken).</p>	<p>A fact type is either asserted, derived, or semiderived.</p> <p>A derived fact type is marked with an asterisk <b>"*"</b>. A derivation rule is supplied. A double asterisk <b>"**"</b> indicates derived and stored (eager evaluation).</p>
Semiderived fact type, and derivation rule	 <p><b>+Person<sub>1</sub> is a grandparent of Person<sub>2</sub></b> <b>if</b> <b>Person<sub>1</sub> is a parent of some Person<sub>3</sub></b> <b>who is a parent of Person<sub>2</sub>.</b></p>	<p>A fact type is semiderived if some of its instances may be derived, and some of its instances may be simply asserted.</p> <p>It is marked by <b>"+"</b> (half an asterisk). <b>"++"</b> indicates semiderived and stored (eager evaluation for derived instances).</p>
Subtyping		<p>All subtypes are proper subtypes. An arrow runs from subtype to supertype. A solid arrow indicates a path to the subtype's preferred identifier (e.g. here, student employees are primarily identified by their employee number). A dashed arrow indicates the supertype has a different preferred identifier.</p>
Subtyping constraints		<p>A circled <b>"X"</b> indicates the subtypes are mutually exclusive. A circled dot indicates the supertype equals the union of the subtypes. The combination (xor constraint) indicates the subtypes partition the supertype (exclusive and exhaustive).</p>
Subtype derivation status	 <p><b>*Each MalePerson is a Person</b> <b>who is of Gender 'M'.</b></p> <p><b>+ Each derived Grandparent is a Person</b> <b>who is a parent of some Person</b> <b>who is a parent of some Person.</b></p>	<p>A subtype may be</p> <ul style="list-style-type: none"> <li>• asserted,</li> <li>• derived (denoted by <b>"*"</b>),</li> <li>• or semiderived (denoted by <b>"+"</b>).</li> </ul> <p>If the subtype is asserted, it has no mark appended and has no derivation rule.</p> <p>If the subtype derived or semiderived, a derivation rule is supplied.</p>

Construct	Examples	Description/Notes
<p>Internal frequency constraint</p>		<p>This constrains the number of times an occurring instance of a role or role sequence may appear in each population. Here: each jury has exactly 12 members; each panel that includes an expert includes at least 4 and at most 7 experts; each expert reviews at most 5 papers; each paper that is reviewed is reviewed by at least 2 experts; and each department and year that has staff numbers recorded in the quaternary appears there twice (once for each gender).</p>
<p>External frequency constraint</p>		<p>The example constraint has the following meaning. In this context, each combination of student and course relates to at most two enrolments (i.e. a student may enroll at most twice in the same course)</p>
<p>Ring constraints</p>		<p>A ring predicate <math>R</math> is locally reflexive if and only if, for all <math>x</math> and <math>y</math>, <math>xRy</math> implies <math>xRx</math>. E.g. “knows” is locally but not globally reflexive.</p> <p>Reflexive, symmetric and transitive properties may also be enforced using semiderivation rather than by constraining asserted fact types.</p> <p>The example constrains the subtyping relationship in ORM to be both acyclic (no cycles can be formed by a chain of subtyping connections) and strongly intransitive (no object type <math>A</math> can be both a direct subtype of another type <math>B</math> and an indirect subtype of <math>B</math>, where indirect subtyping means there is a chain of two or more subtyping relationships that lead from <math>A</math> to <math>B</math>).</p> <p>Ring constraints may be combined only if they are compatible, and one is not implied by the other. ORM tools ensure that only legal combinations are allowed.</p>
<p>Value-comparison constraints</p>		<p>The example constraint verbalizes as:  <b>For each Project,</b>  <b>existing</b> enddate <math>\geq</math> startdate.</p>

Construct	Examples	Description/Notes
Object cardinality constraint		The example constraints ensure there is exactly one president and at most 100 senators (at any given time),
Role cardinality constraint		The example constraint ensures that at most one politician plays the role of president (at any given time).
Deontic constraints	<p>Uniqueness </p> <p>Mandatory </p> <p>Subset, Equality, Exclusion </p> <p>Frequency </p> <p>Irreflexive  Acyclic </p> <p>Asymmetric  Asym-Intrans </p> <p>Intransitive  Acyclic-Intrans </p> <p>Antisymmetric  Symmetric </p> <p>Strongly Intransitive  etc.</p> <p>e.g.</p> 	<p>Unlike alethic constraints, deontic constraint shapes are colored blue rather than violet. Most include “o” for “obligatory”. Deontic ring constraints instead use dashed lines.</p> <p>In the parenthood example, the alethic frequency constraint ensures that each person has at most two parents, the alethic ring constraint ensures that parenthood is acyclic, and the deontic ring constraint makes it obligatory for parenthood to be strongly intransitive.</p>
Textual constraints	 <p><sup>1</sup> Each Employee who has Rank 'NonExec' uses at most one CompanyCar.  <sup>2</sup> Each Employee who has Rank 'Exec' uses some CompanyCar.</p>	First-order constraints with no graphic notation may be expressed textually in the FORML 2 language. These examples use footnoting to capture a restricted uniqueness constraint and a restricted mandatory role constraint.
Objectification display options: link fact types, and compact display.		Internally, link fact types connect objectified associations to their component object types. By default, display of link fact types is suppressed. If displayed, link predicate shapes use dashed lines instead of solid lines. Objectification object types may also be displayed without their defining components, using an object type shape containing a small predicate shape, as shown in this Enrolment example.