i

Forside

UNIVERSITET I OSLO

Det matematisk-naturvitenskaplege fakultet

Eksamen i IN2090/INF1300

Dato: 6 desember 2019
Tid: 14:30–18:30
Stad: Silurveien 2 (Sal 3B, 3C, 3D, 4C, 4D)

Tillatte hjelpemidler:

- Læreboka Elmasri & Navathe: Fundamentals of Database Systems, Global Edition, 7th Edition (læreboka brukt i IN2090, fom. 2019).
- Læreboka Halpin & Morgan: Information Modelling and Relational Databases, Second Edition (læreboka brukt i INF1300 og IN2090 hausten 2018).
- 4 handskrevne A4-sider med notat (2 ark om det er skreven på begge sider)

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

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

I dette oppgåvesettet skal du svare med digital handteikning på modelleringsoppgåvene, altså oppgåve 1.1 og 1.2. Du bruker skisseark du får utdelt. Du har høve til å bruke fleire ark per oppgåve. Sjå instruksjon for utfylling av skisseark i lenka under oppgåvelinja. På desse oppgåvene kan du velje om du vil løyse dei med modelleringsspråket ER eller ORM2.

Det er IKKJE høve til å bruke digital handteikning på andre oppgåver enn oppgåve X og X. Det blir IKKJE gitt ekstratid for å fylle ut informasjonsboksane på skissearka (eingongskodar, kand.nr. o.l.).

På fleirvalgsoppgåvene i oppgåve 4, 12 og 13, er det minuspoeng for galt svar. Men den totale poengsumma for kvar oppgåve blir aldri lågare enn 0.

1.1 Modellering: Kandidater

I denne (og neste) oppgåve skal du lage ein modell for ein database som inneheld informasjon om intervjukandidatar (for programmvareutviklere) og intervjuene dei deltek på. Du veljer sjølv om du vil lage modellen i ER (oversikt over ER-notasjon finnar du i vedlagt dokument) eller ORM2 (oversikt over ORM2-notasjon finnar du i vedlagt dokument).

Modellene skal tegnes på papir ved hjelp av Scantron. Du kan velje om du vil lage ein stor modell for begge oppgåvene, eller om du vil dele dei opp i to modeller. Dersom du veljer å dele modellen opp i to, trenge du kun å teikne inn entitetane (og deira nøkler) for dei entitetane du trenge frå oppgåve 1 i modellen for oppgåve 2. Du må gjerne inkludere kommentarar til modellane. Skriv og teikn tydeleg.

Modellen vi skal lage i denne første oppgåven skal modellere kandidatar:

1. For kvar kandidat skal databasen kunne lagre eit personnummer som er unikt for alle kandidatar, eit namn, ein adresse, og ein email (som også er unik for kvar kandidat).
2. Vidare skal databasen innehalde informasjon om den noverande arbeidsgjevaren til kvar kandidat. Dersom kandidaten er arbeidslaus skal inga arbeidsgjevar lagres. Dersom kandidaten har fleire enn ein arbeidsgjevar (til dømes, dersom kandidaten har to deltidsjobbar) er vi kun interessert i ein av dei. (Altså, kvar kandidat har maksimalt ein arbeidsgjevar i vår database). Ein arbeidsgjevar kan ha mange kandidatar som ansatte. For kvar kandidat ynskjer vi også å lagre kandidaten sin noverande jobbtittel (f.eks. "produktansvarlig").
3. For kvar arbeidsgjevar er vi interessert i å lagre ein unik ID, eit namn, og kontaktinformasjon. Kontakinformasjonen består igjen av telefonnummer, email og addresse.

I denne oppgåva kan du svare med digital handteikning. Bruk eige skisseark (utdelt). Sjå instruksjon for utfylling av skisseark i lenka under oppgåvelinja.

Maks poeng: 7

1.2 Modellering: Intervjuer

Vi skal fortsetje på modellen om kandidatar fra forrige oppgåve. I denne oppgåva skal vi modellere intervjua.

I likskap med forrige oppgåve kan du velje mellom ER og ORM, og modellen skal lagast på papir. Du kan velje om du vil laga ein stor modell for begge oppgåvene, eller om du vil dele dei opp i to modellar. Dersom du veljar å dele modellane opp i to, trenge du kun å teikne inn entitetane (og deira nøkler) for dei entitetane du treng fra den forrige modellen.

Modellen du skal laga i denne oppgåven skal innehalde informasjon om intervju:

1. For kvar kandidat skal databasen kunne lagre kva for eit intrvju ein kandiat deltok på. Kvart intervju har eit intervjunummer, og ein dato og eit tidspunkt kor intervjuet blei haldt. Intervjuer kan unikt identifiserast med kombinasjonen av intervjuet sitt intervjunummer og personnummeret til kandidaten som deltok på intervjuet. For eksempel: kandidaten med personnummer #123 deltek på intervju #1 haldt 18.12.19 klokka 14:00; kandidaten med personnummer #123 deltek på intervju #2 dato 21.12.2019 klokka 15:00; kandidaten med personnummer #456 deltek på intervju #1 dato 21.12.2019 klokka 15:00, osv.
2. Intervjuene blir haldt av ein intervjuar på eit bestemt emne (hint: bruk ein ternær relasjon). Kvart emne har eit unikt namn (f.eks. "datastrukturer") og ein mengde testar (i databasen vil vi bare representere testane med eit namn). Kvar intervjuar har ein ansatt-id og ein mengde telefonnummer.
3. Databasen skal kunne lagre at mange intervju kan bli haldt av mange intervjuare for mange emner (med andre ord: gitt eit intervju og ein intervjuar, så kan vi ha mange emner; gitt eit intervju og eit emne, kan vi ha mange intervjuare; og gitt eit emne og ein intervjuar kan vi ha mange intervju). Til dømes: kandidat med personnummer #123 deltek på intervju #1 haldt av intervjuar med ansatt-id #111 om emne "datastrukturer"; kandidat med personnummer #456 deltek på intervju med nummer #1 haldt av intervjuar med ansattnummer #111 om emne "Java-programmering", osv.
4. Vidare skal databasen kunne lagre at kvar intervjuar er sertifisert på minst eitt emne (men kan være sertifisert på fleire). Kvart emne kan ha mange sertifiserte intervjuare. Sertifisering på eit emne skjer på ein bestemt dato og av ein gitt sertifikatutferdar. (F.eks. intervjuar med ansatt-id #111 er sertifisert i emne "datastrukturer" frå dato 19.01.19 av Universitetet i Oslo.)
5. I tillegg har kvart emne nøyaktig éin ansvarleg intervjuar (f.eks. den som er ansvarleg for å lage testane for emnet). Ein inertvjuar kan være ansvarleg for maksimalt eitt emne (men ein intervjura trenger ikkje være ansvarleg for eit emne).

I denne oppgåva kan du svare med digital handteikning. Bruk eige skisseark (utdelt). Sjå instruksjon for utfylling av skisseark i lenka under oppgåvelinja.

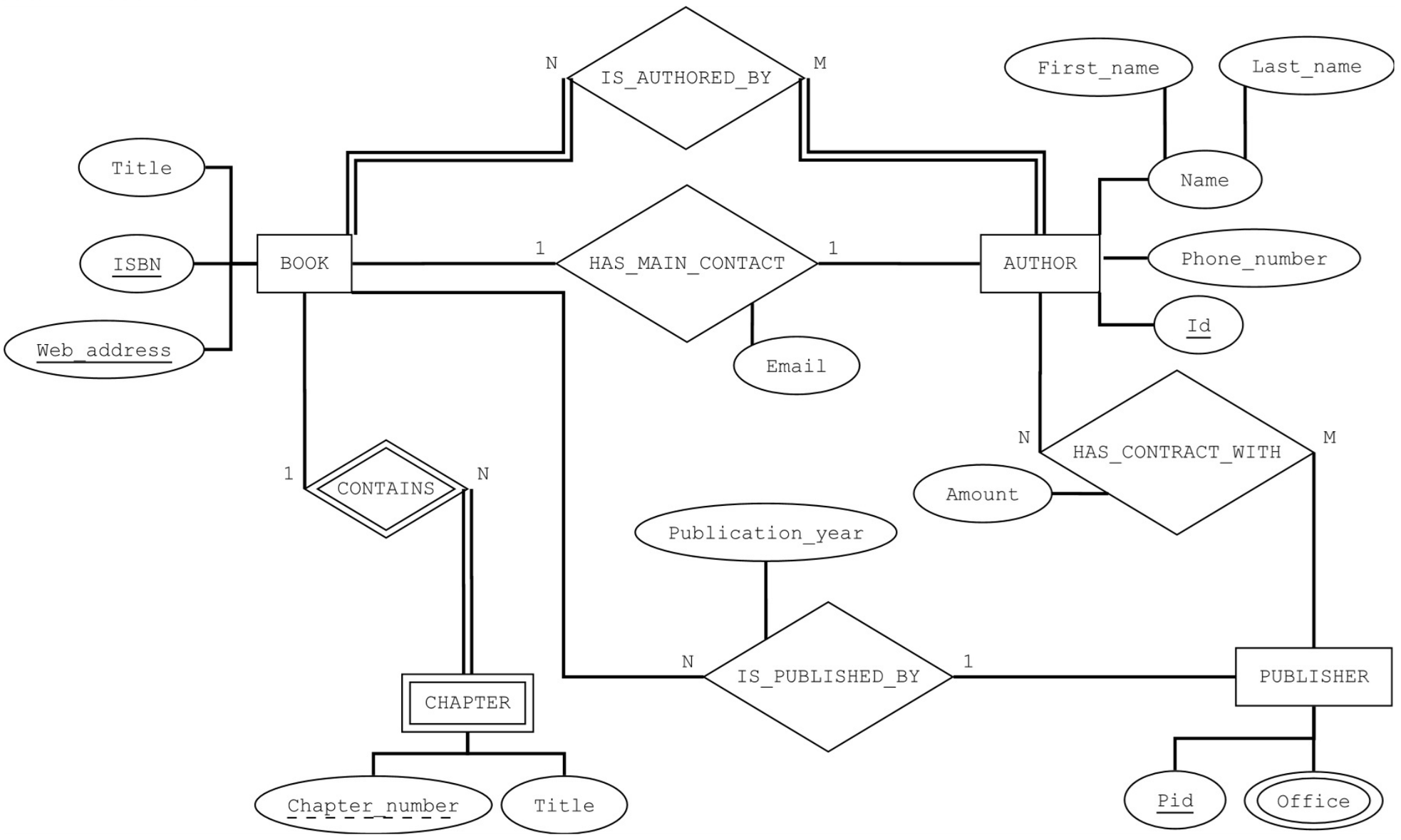
Maks poeng: 18

1.3 Realisering

Realiser anten ER-modellen eller ORM2-modellen vist under til eit databaseskjema ved å bruke algoritmen for realisering. Forklar eventuelle val du tek underveis.

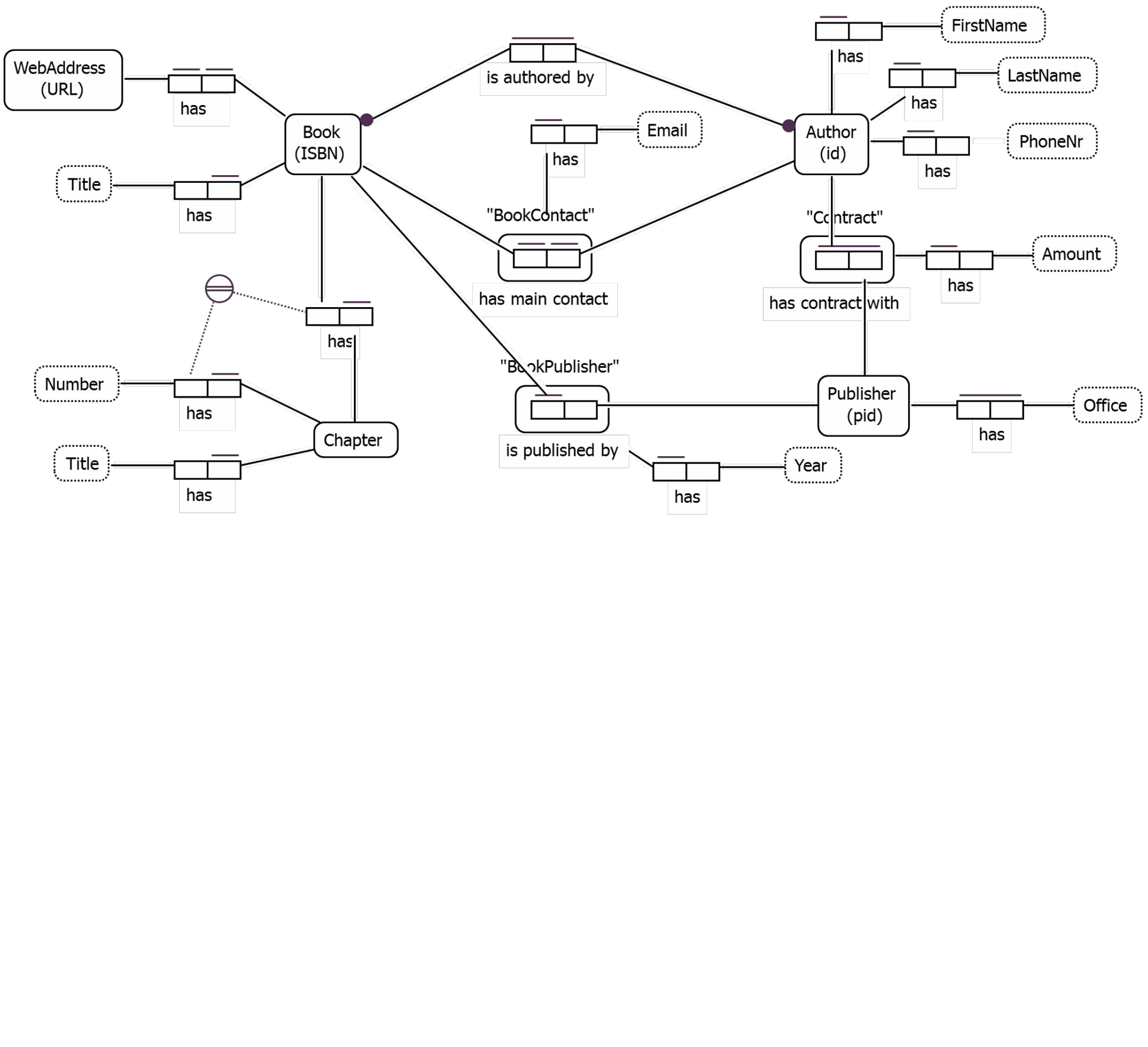
Bruk understreking for å markere kandidatnøklar. Bruk i tillegg **fet skrift** for å markere primærnøkkelen for kvar relasjon (du finner knappar for understreking og fet skrift i menyen over). List også opp alle fremmednøklar på formen T(A) -> P(B) (her er attributtan(e) A i T fremmednøkkel som referere til relasjon P sin(e) B attributt(ar)).

ER-modell:


IS_AUTHORED_BY
N
M
First_name
Last_name
Name
Title
ISBN
Web_address
BOOK
1
HAS_MAIN_CONTACT
1
AUTHOR
Phone_number
Id
Email
1
CONTAINS
N
N
HAS_CONTRACT_WITH
M
Amount
Publication_year
N
IS_PUBLISHED_BY
1
PUBLISHER
CHAPTER
Chapter_number
Title
Pid
Office

ORM2-modell:

-


WebAddress (URL)
has
Book (ISBN)
is authored by
Email
has
Author (id)
FirstName
has
LastName
has
PhoneNr
has
Title
has
"BookContact"
"Contract"
Amount
has
has main contact
has contract with
has
Number
has
Chapter
Title
has
"BookPublisher"
Publisher (pid)
Office
has
is published by
Year
has

Skriv svaret ditt her...

Format

Words: 0

Maks poeng: 15

2.1 Skranker og SQL

Gitt ein database laga av føljande 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 kvar av SQL-kommandoane under, avgjer om dei er lovlege i henhald til databaseskjemaet over (altså, kommandoen lykkast utan error) eller ulovlege i henhald til databaseskjemaet over (altså kommandoen feilar og gir error).

	Lovleg	Ulovleg
`UPDATE dyr SET mor_til = 4 WHERE mor_til IS NULL;`	○	○
`INSERT INTO dyr VALUES (6, 'Timmy', NULL, 2);`	○	○
`UPDATE dyr SET mor_til = 3 WHERE did = 0;`	○	○
`DELETE FROM dyr WHERE did = 3;`	○	○
`INSERT INTO art VALUES (4, 'Flue', 'Innsekt');`	○	○
`DELETE FROM dyr WHERE navn = 'Mons';`	○	○
`INSERT INTO dyr VALUES (7, 'Mona', 1, 0);`	○	○
`DROP TABLE art CASCADE;`	○	○

Maks poeng: 10

2.2 Band etter 2000

I denne (og dei neste) oppgåven skal du bruke føljande databaseskjema:

Sjanger(sjangerID, namn)
Band(bandID, namn, starta, sjangerID)
Person(personID, namn, født)
Medlem(personID, bandID)
Album(albumID, bandID, namn, utgitt)
Song(songID, namn, spilletid, albumID)

med føljande fremmednøklar:

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

Ein sjanger består av ein unik sjangerID og eit namn (f.eks. 'pop' eller 'metal'); eit band har ein unik bandID, eit namn, ein dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet speler i; ein person har ein unik personID, eit namn og ein dato personen er født; personer kan vere medlem i eit band og er beksreven i relasjonen Medlem (merk at ein person kan vere medlem i mange band og eit band kan ha mange medlemmer); eit album består av ein unik albumID, et namn, bandIDen til bandet som laga albummet, og ein dato for når albummet blei utgitt; ein song har ein unik sangID, eit namn, ein spilletid i sekundar, og ein albumID som seier kva for eit album songen er ein del av.

For eksempel kan databasen innehalde føljande data:

Sjanger

sjangerID	**namn**
0	Pop
1	Rock
2	Metal

Band

bandID	**namn**	**starta**	**sjangerID**
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violett	1989-04-23	1

Person

personID	**namn**	**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**	**namn**	**utgitt**
0	0	Surfin	1986-02-07
1	1	Love	1999-11-09
2	0	Fog on the grass	1990-03-13

Song

sangID	namn	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

Oppgåve: Skriv ein spørring som finnar alle band som anten blei starta etter år 2000 eller innehalder strengen 'King' i namnet. Skriv ut namnet på bandet og datoen bandet blei starta.

Skriv svaret ditt her...

```
1
```

Maks poeng: 5

2.3 Timer Pop-musikk fra 90s

I denne (og dei neste) oppgåven skal du bruke føljande databaseskjema:

Sjanger(sjangerID, namn)
Band(bandID, namn, starta, sjangerID)
Person(personID, namn, født)
Medlem(personID, bandID)
Album(albumID, bandID, namn, utgitt)
Song(songID, namn, spilletid, albumID)

med føljande fremmednøklar:

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

Ein sjanger består av ein unik sjangerID og eit namn (f.eks. 'pop' eller 'metal'); eit band har ein unik bandID, eit namn, ein dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet speler i; ein person har ein unik personID, eit namn og ein dato personen er født; personer kan vere medlem i eit band og

er beksreven i relasjonen Medlem (merk at ein person kan vere medlem i mange band og eit band kan ha mange medlemmer); eit album består av ein unik albumID, et namn, bandIDen til bandet som laga albummet, og ein dato for når albummet blei utgitt; ein song har ein unik sangID, eit namn, ein spilletid i sekundar, og ein albumID som seier kva for eit album songen er ein del av.

For eksempel kan databasen innehalde føljande data:

Sjanger

sjangerID	**namn**
0	Pop
1	Rock
2	Metal

Band

bandID	**namn**	**starta**	**sjangerID**
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violett	1989-04-23	1

Person

personID	**namn**	**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**	**namn**	**utgitt**
0	0	Surfin	1986-02-07
1	1	Love	1999-11-09
2	0	Fog on the grass	1990-03-13

Song

sangID	**namn**	**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

Oppgåve: Skriv ein spørring som finnar talet timar med musikk frå sjangeren 'Pop' laga av band starta mellom år 1990 og 2000. Merk: Ein time er 3600 sekundar.

Skriv svaret ditt her...

```
1
```

Maks poeng: 5

2.4 Personer født på interessant dato

I denne (og dei neste) oppgåven skal du bruke føljande databaseskjema:

Sjanger(sjangerID, namn)
Band(bandID, namn, starta, sjangerID)
Person(personID, namn, født)
Medlem(personID, bandID)
Album(albumID, bandID, namn, utgitt)
Song(songID, namn, spilletid, albumID)

med føljande fremmednøklar:

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

Ein sjanger består av ein unik sjangerID og eit namn (f.eks. 'pop' eller 'metal'); eit band har ein unik bandID, eit namn, ein dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet speler i; ein person har ein unik personID, eit namn og ein dato personen er født; personer kan vere medlem i eit band og er beksreven i relasjonen Medlem (merk at ein person kan vere medlem i mange band og eit band kan ha mange medlemmer); eit album består av ein unik albumID, et namn, bandIDen til bandet som laga albummet, og ein dato for når albummet blei utgitt; ein song har ein unik sangID, eit namn, ein spilletid i sekundar, og ein albumID som seier kva for eit album songen er ein del av.

For eksempel kan databasen innehalde føljande data:

Sjanger

sjangerID	**namn**
0	Pop
1	Rock
2	Metal

Band

bandID	namn	starta	sjangerID
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violett	1989-04-23	1

Person

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

Song

sangID	namn	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

Oppgåve: Skriv ein spørring som finnar namnet på alle personar født på ein dato kor det anten blei starta eit nytt band, eller blei gitt ut eit nytt album.

Skriv svaret ditt her...

```
1
```

Maks poeng: 5

2.5 Sanger per band

I denne (og dei neste) oppgåven skal du bruke føljande databaseskjema:

Sjanger(sjangerID, namn)
Band(bandID, namn, starta, sjangerID)
Person(personID, namn, født)
Medlem(personID, bandID)
Album(albumID, bandID, namn, utgitt)
Song(songID, namn, spilletid, albumID)

med føljande fremmednøklar:

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

Ein sjanger består av ein unik sjangerID og eit namn (f.eks. 'pop' eller 'metal'); eit band har ein unik bandID, eit namn, ein dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet speler i; ein person har ein unik personID, eit namn og ein dato personen er født; personer kan vere medlem i eit band og er beksreven i relasjonen Medlem (merk at ein person kan vere medlem i mange band og eit band kan ha mange medlemmer); eit album består av ein unik albumID, et namn, bandIDen til bandet som laga albummet, og ein dato for når albummet blei utgitt; ein song har ein unik sangID, eit namn, ein spilletid i sekundar, og ein albumID som seier kva for eit album songen er ein del av.

For eksempel kan databasen innehalde føljande data:

Sjanger

sjangerID	namn
0	Pop
1	Rock
2	Metal

Band

bandID	namn	starta	sjangerID
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violett	1989-04-23	1

Person

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

Song

sangID	namn	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

Oppgåve: Skriv ein spørring som finnar talet songer kvart band har laga (altså talet songer på alle deira album til saman) for band som har laga færre enn 3 songer. Skriv ut bandIDen, namnet på bandet og talet songer.

Merk: Det kan finnast band som ikkje har gitt ut nokon album ennå eller band som kun har utgitt album utan songer, desse skal også med i resultatet med talet songer lik 0.

Skriv svaret ditt her...

```
1
```

Maks poeng: 5

2.6 Slett tomme album

I denne (og dei neste) oppgåven skal du bruke føljande databaseskjema:

Sjanger(sjangerID, namn)
Band(bandID, namn, starta, sjangerID)
Person(personID, namn, født)
Medlem(personID, bandID)
Album(albumID, bandID, namn, utgitt)
Song(songID, namn, spilletid, albumID)

med føljande fremmednøklar:

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

Ein sjanger består av ein unik sjangerID og eit namn (f.eks. 'pop' eller 'metal'); eit band har ein unik bandID, eit namn, ein dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet speler i; ein person har ein unik personID, eit namn og ein dato personen er født; personer kan vere medlem i eit band og er beksreven i relasjonen Medlem (merk at ein person kan vere medlem i mange band og eit band kan ha mange medlemmer); eit album består av ein unik albumID, et namn, bandIDen til bandet som laga albummet, og ein dato for når albummet blei utgitt; ein song har ein unik sangID, eit namn, ein spilletid i sekundar, og ein albumID som seier kva for eit album songen er ein del av.

For eksempel kan databasen innehalde føljande data:

Sjanger

sjangerID	**namn**
0	Pop
1	Rock
2	Metal

Band

bandID	namn	starta	sjangerID
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violett	1989-04-23	1

Person

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

Song

sangID	namn	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

Oppgåve: Skrive ein SQL-kommando som slettar alle album som ikkje har nokre tilhøyrande songer.
Skriv svaret ditt her...

```
1
```

Maks poeng: 5

2.7 Nyeste album

I denne (og dei neste) oppgåven skal du bruke føljande databaseskjema:

Sjanger(sjangerID, namn)
Band(bandID, namn, starta, sjangerID)
Person(personID, namn, født)
Medlem(personID, bandID)
Album(albumID, bandID, namn, utgitt)
Song(songID, namn, spilletid, albumID)

med føljande fremmednøklar:

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

Ein sjanger består av ein unik sjangerID og eit namn (f.eks. 'pop' eller 'metal'); eit band har ein unik bandID, eit namn, ein dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet speler i; ein person har ein unik personID, eit namn og ein dato personen er født; personer kan vere medlem i eit band og er beksreven i relasjonen Medlem (merk at ein person kan vere medlem i mange band og eit band kan ha mange medlemmer); eit album består av ein unik albumID, et namn, bandIDen til bandet som laga albummet, og ein dato for når albummet blei utgitt; ein song har ein unik sangID, eit namn, ein spilletid i sekundar, og ein albumID som seier kva for eit album songen er ein del av.

For eksempel kan databasen innehalde føljande data:

Sjanger

sjangerID	namn
0	Pop
1	Rock
2	Metal

Band

bandID	namn	starta	sjangerID
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violett	1989-04-23	1

Person

personID	namn	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

Al

Album

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

Song

sangID	namn	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

Oppgåve: Skriv ein SQL-kommando som lagar eit view med namn "nyeste_album" som inneheld dei 10 nyaste albummane. Viewet skal for kvart av albummane vise albummet sitt namn og namnet på bandet som utga albummet, datoen albummet er gitt ut, og talet songer på albummet. Sorter viewet etter når albummet er utgitt, kor det nyaste er først.

(Merk: Du kan her anta at alle album har minst éin song)

Skriv svaret ditt her...

```
1
```

Maks poeng: 5

2.8 Super-album

I denne (og dei neste) oppgåven skal du bruke følgande databaseskjema:

Sjanger(sjangerID, namn)
Band(bandID, namn, starta, sjangerID)
Person(personID, namn, født)
Medlem(personID, bandID)
Album(albumID, bandID, namn, utgitt)
Song(songID, namn, spilletid, albumID)

med følgande fremmednøklar:

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

Ein sjanger består av ein unik sjangerID og eit namn (f.eks. 'pop' eller 'metal'); eit band har ein unik bandID, eit namn, ein dato som sier når bandet ble startet og en sjangerID som peker på sjangeren bandet speler i; ein person har ein unik personID, eit namn og ein dato personen er født; personer kan vere medlem i eit band og er beksreven i relasjonen Medlem (merk at ein person kan vere medlem i mange band og eit band kan ha mange medlemmer); eit album består av ein unik albumID, et namn, bandIDen til bandet som laga albummet, og ein dato for når albummet blei utgitt; ein song har ein unik sangID, eit namn, ein spilletid i sekundar, og ein albumID som seier kva for eit album songen er ein del av.

For eksempel kan databasen innehalde føljande data:

Sjanger

sjangerID	**namn**
0	Pop
1	Rock
2	Metal

Band

bandID	**namn**	**starta**	**sjangerID**
0	Blue Floyd	1985-01-19	1
1	Bettany Swords	1991-12-01	0
2	Shallow Violett	1989-04-23	1

Person

personID	**namn**	**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**	**namn**	**utgitt**
0	0	Surfin	1986-02-07
1	1	Love	1999-11-09
2	0	Fog on the grass	1990-03-13

Song

sangID	**namn**	**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

Oppgåve: Skriv namnet på alle band som har gitt ut eit super-album. Eit super-album er eit album med

speletid på meir enn ein time (3600 sekundar). Skriv ut bandIDen, namnet på bandet, samt talet super-album bandet har utgitt.

Hint: Det kan vere lurt å først finne "albumID" til alle super-album i ein eiga spørring (f.eks. med WITH). Deretter kan man finne bandet som har laga albummane, og telje opp.

Skriv svaret ditt her...

```
1
```

Maks poeng: 10

3.1 FDer

Gitt føljande relasjon:

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

og føljande funksjonelle avhengigheitar (FDer):

$$A \rightarrow C$$
$$C \rightarrow B, D$$
$$A, B \rightarrow D, F$$
$$B, E \rightarrow G$$

Huk av dei attributtane som er del i tillukninga til {A}.

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

Huk av dei attributtane som er del i tillukninga til {B}.

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

Huk av dei attributtane som må vere del i kvar kandidatnøkkel.

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

Maks poeng: 6

3.2 Normalformer

Gitt føljande relasjon

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

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

For kvar av FDane under, anta at FDen gjeld for relasjonen R over, og bruk algoritmen for normalformar til å avgjere kva for ein normalform FDen (åleine) tilseier at R er på.

	BCNF	2NF	3NF	1NF
A, B -> F	○	○	○	○
A, D -> E	○	○	○	○
A -> C	○	○	○	○
B -> D	○	○	○	○

Maks poeng: 4

Question 1
Attached

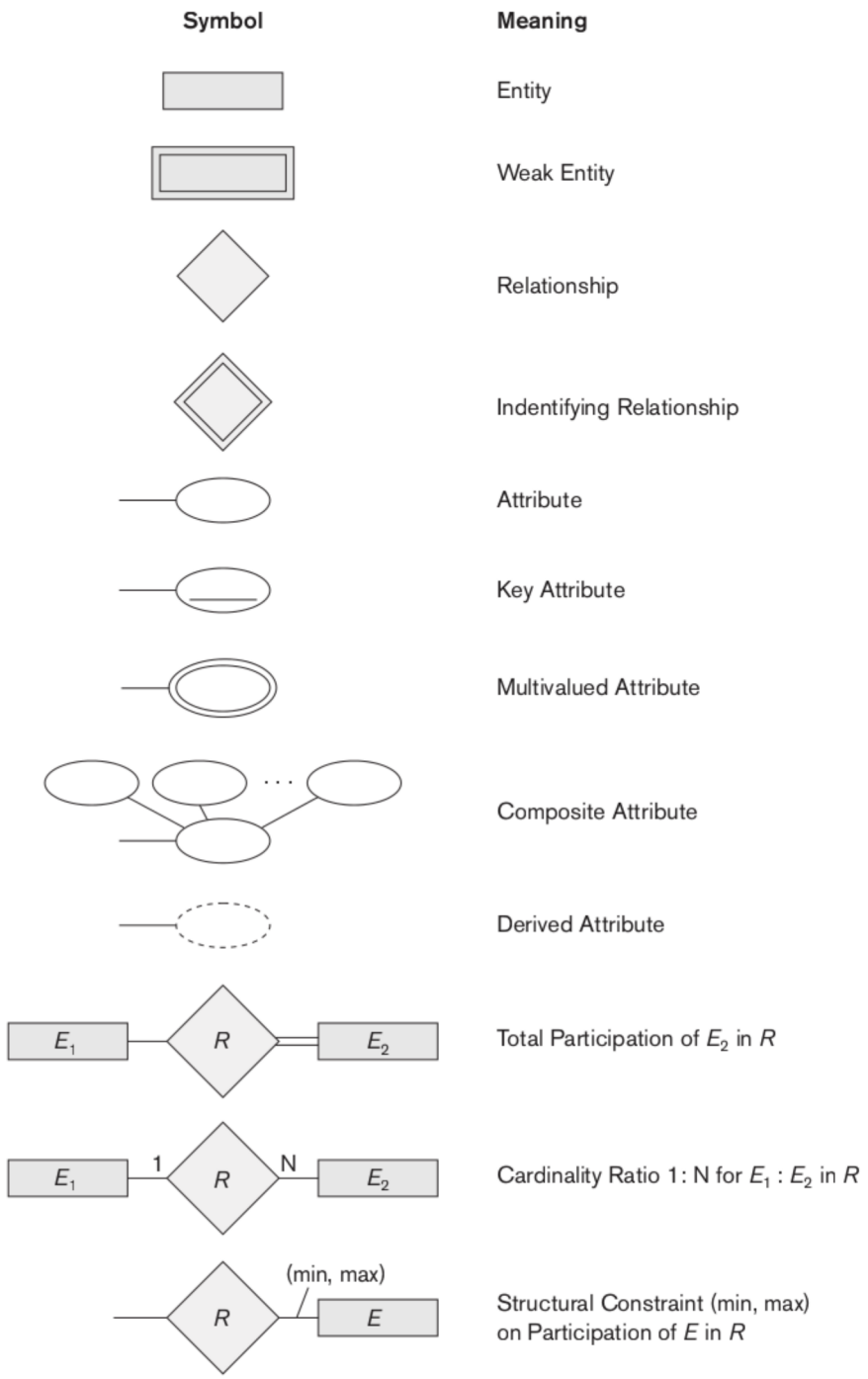
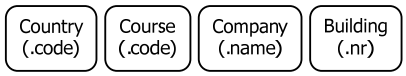
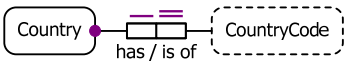
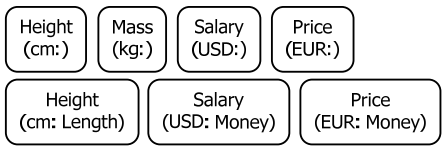
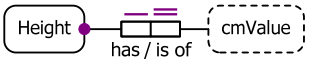
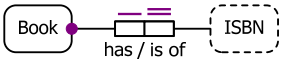
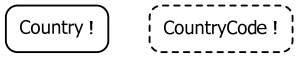
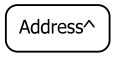
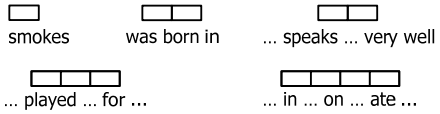
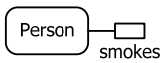
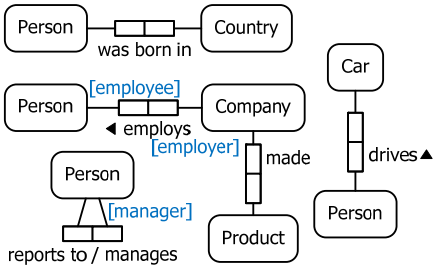
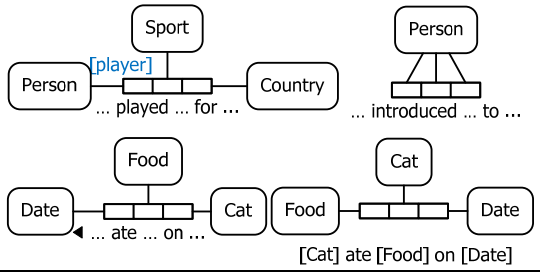
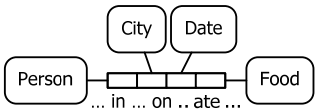
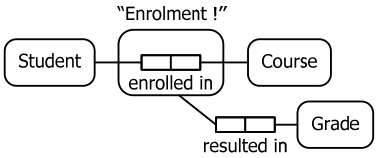
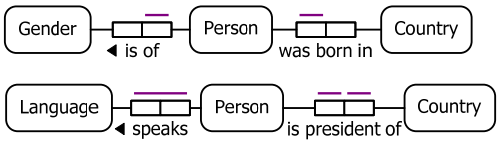
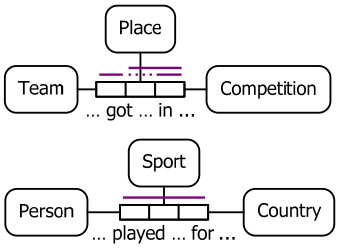
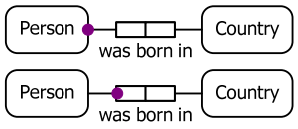
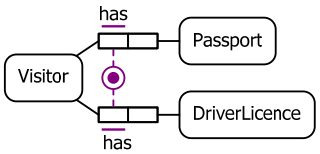
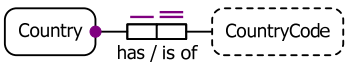


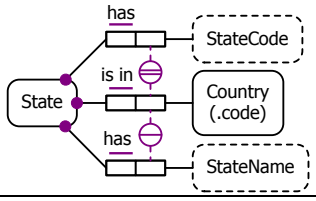
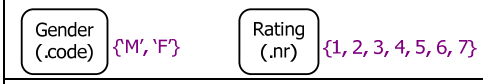
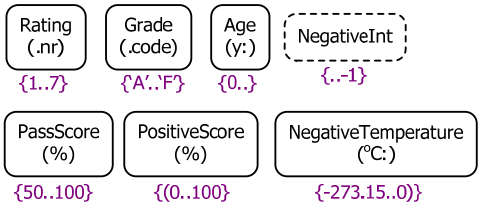
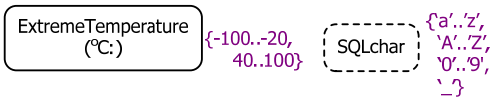
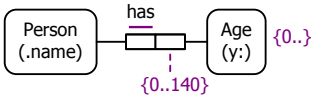
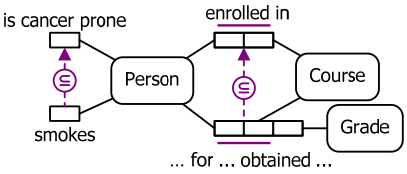
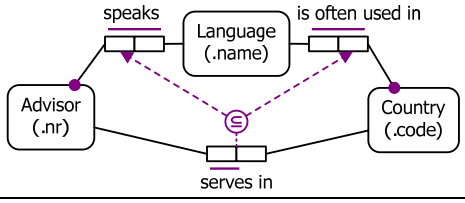
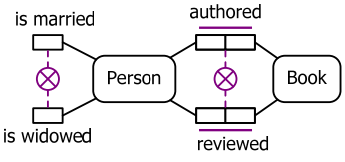
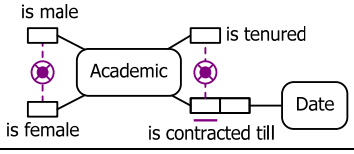
Figure 3.14
Summary of the notation for ER diagrams.

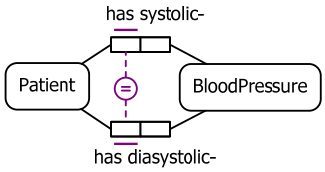
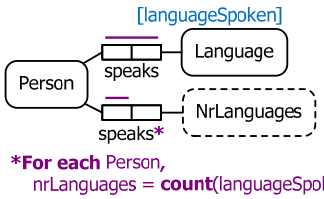
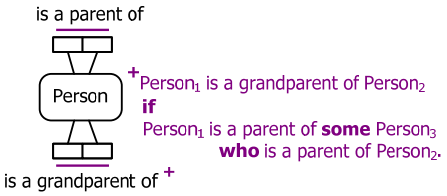
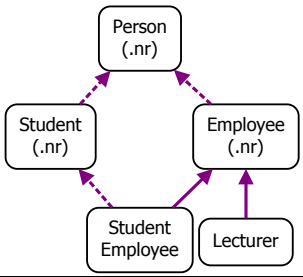
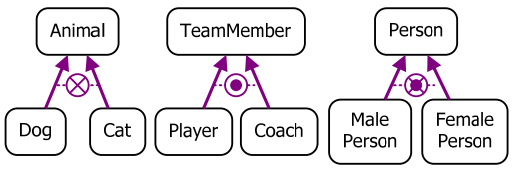
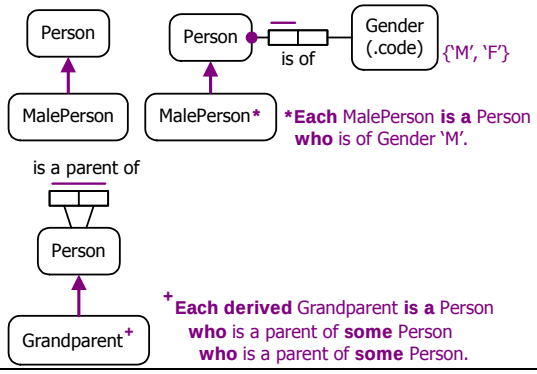
ORM 2 Graphical Notation

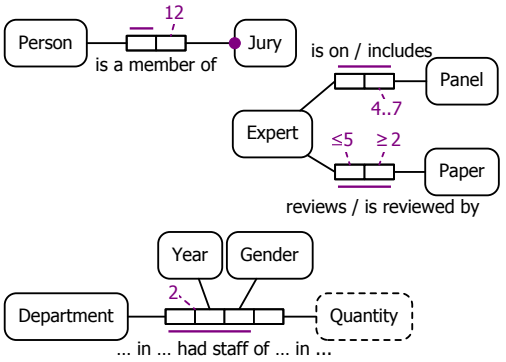
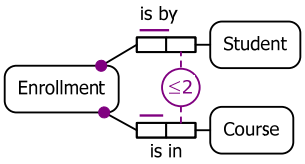
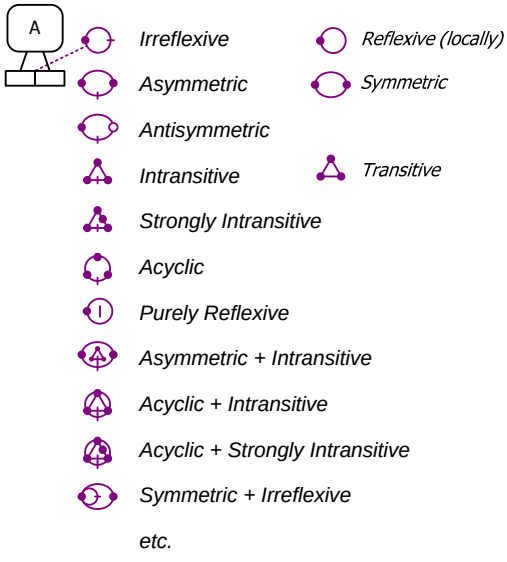
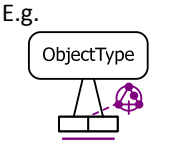
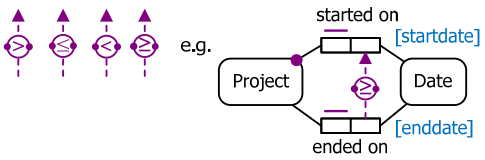
Terry Halpin

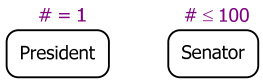
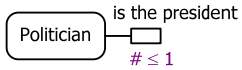
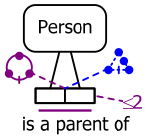
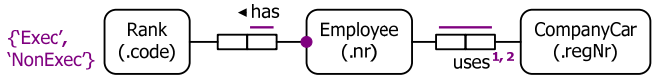
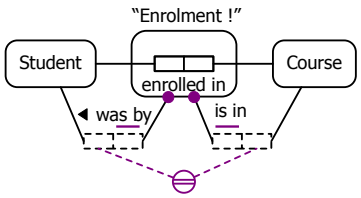
Construct	*Examples*	*Description/Notes*
Entity Type		Named soft rectangle, named hard rectangle, or named ellipse. The soft rectangle shape is the default.
Value Type		Named, dashed, soft rectangle (or hard rectangle or ellipse).
Entity type with popular reference mode		Abbreviation for injective reference 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		Role names may be added in square brackets. Arrow-tips are used to reverse the default left-right or top-down reading order. Reading orders other than forward and reverse are shown using named placeholders.
Quaternary fact type		The above notes for the ternary case apply here also. Fact types of higher arity (number of roles) are also permitted.
Objectification (a.k.a. nesting)		The enrolment fact type is objectified as an entity type whose instances can play roles. In this example, the objectification type is independent, so we can know about an enrolment before the grade is obtained.
Internal uniqueness constraint (UC) on unaries		These are equivalent (by default, predicates are assumed to be populated with sets, so no whole fact may be duplicated).
Internal UC on binaries		The examples show the 4 possible patterns: 1:*n* (one-to-many); *n:*1 (many-to-one); *m:n* (many-to-many); 1:1 (one-to-one0
Internal UC on ternaries. For *n*-aries (n > 1) each UC must span at least *n*-1 roles		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). The second example has a spanning UC (many-to-many-to-many).
Simple mandatory role constraint		The example constraint means that each person was born in some country. The mandatory role dot may be placed at either end of the role connector.
Inclusive-or constraint (disjunctive mandatory role)		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).
Preferred internal UC		A double bar on a UC indicates it underlies the preferred reference scheme.

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		This constraint means that a patient's systolic BP is recorded if and only if his/her diastolic BP is recorded. An equality constraint may apply between 2 or more compatible role sequences, possibly involving joins.
Derived fact type, and derivation rule		A fact type is either asserted, derived, or semiderived. A derived fact type is marked with an asterisk "*". A derivation rule is supplied. A double asterisk "**" indicates derived and stored (eager evaluation).
Semiderived fact type, and derivation rule		A fact type is semiderived if some of its instances may be derived, and some of its instances may be simply asserted. It is marked by "+" (half an asterisk). "++"indicates semiderived and stored (eager evaluation for derived instances).
Subtyping		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.
Subtyping constraints		A circled "X" 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).
Subtype derivation status		A subtype may be • asserted, • derived (denoted by "*"), • or semiderived (denoted by "+"). If the subtype is asserted, it has no mark appended and has no derivation rule. If the subtype derived or semiderived, a derivation rule is supplied.

Construct	*Examples*	*Description/Notes*
Internal frequency constraint		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).
External frequency constraint		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)
Ring constraints	 E.g. 	A ring predicate *R* is locally reflexive if and only if, for all *x* and *y*, *xRy* implies *xRx*. E.g. "knows" is locally but not globally reflexive. Reflexive, symmetric and transitive properties may also be enforced using semiderivation rather than by constraining asserted fact types. 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 *A* can be both a direct subtype of another type *B* and an indirect subtype of *B,* where indirect subtyping means there is a chain of two or more subtyping relationships that lead from *A* to *B*). 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.
Value-comparison constraints		The example constraint verbalizes as: **For each** Project, **existing** enddate >= startdate.

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	*Uniqueness* *Mandatory* *Subset, Equality, Exclusion* *Frequency* ≤2 *Irreflexive* *Acyclic* *Asymmetric* *Asym-Intrans* *Intransitive* *Acyclic-Intrans* *Antisymmetric* *Symmetric* *Strongly Intransitive* *etc.* e.g. 	Unlike alethic constraints, deontic constraint shapes are colored blue rather than violet. Most include “o” for “obligatory”. Deontic ring constraints instead use dashed lines. 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.
Textual constraints	 [1] **Each** Employee **who** has Rank 'NonExec' uses **at most one** CompanyCar. [2] **Each** Employee **who** has Rank 'Exec' uses **some** CompanyCar.	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

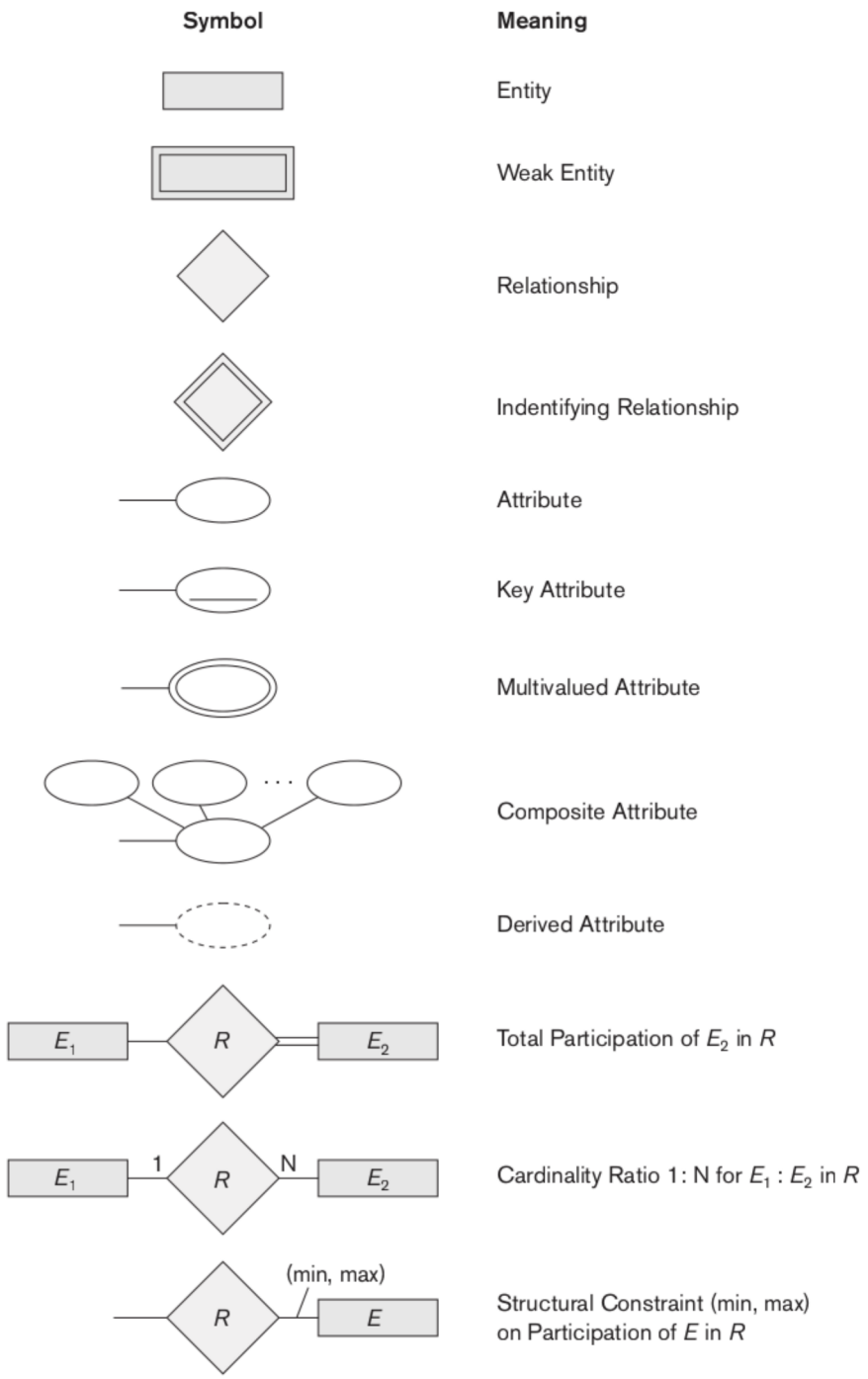
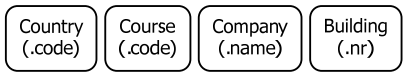
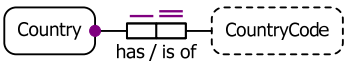
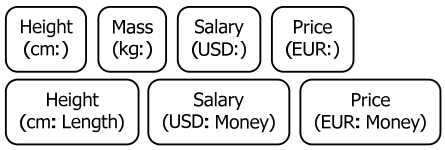
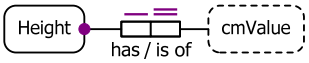
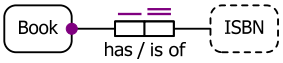
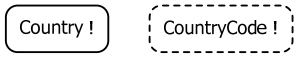
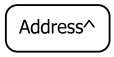
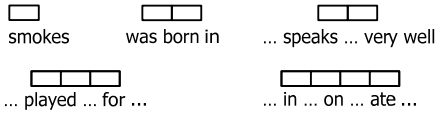
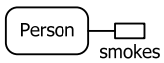
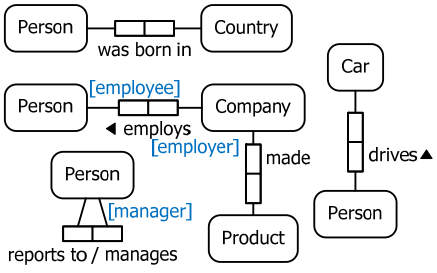
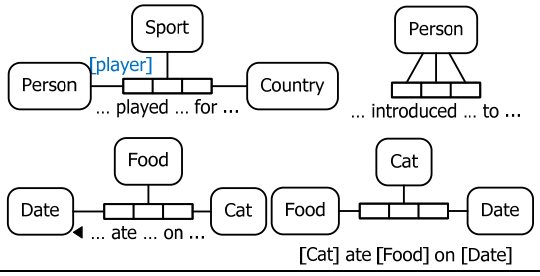
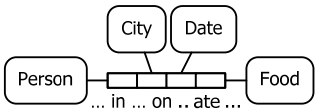
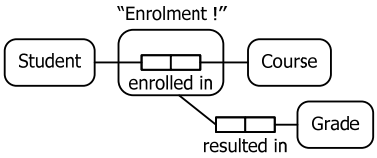
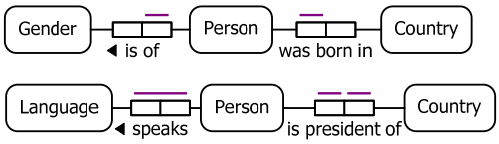
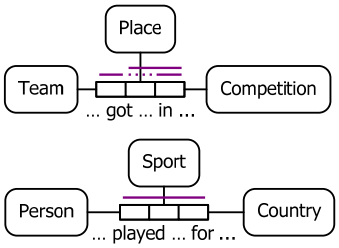
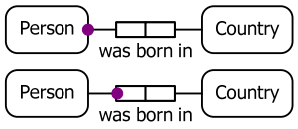
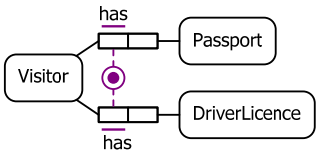
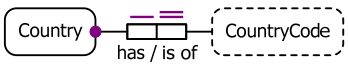


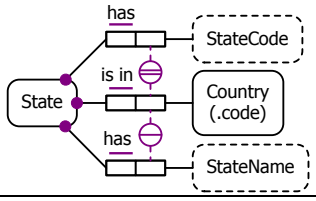
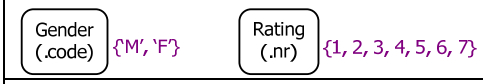
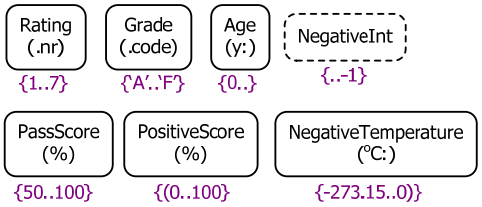
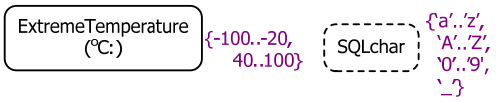
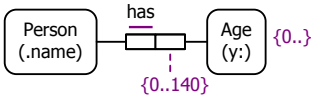
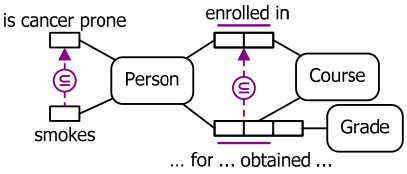
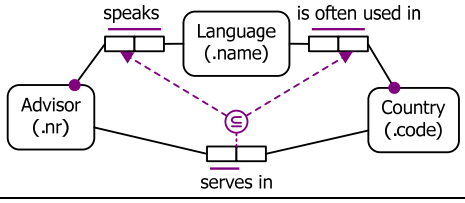
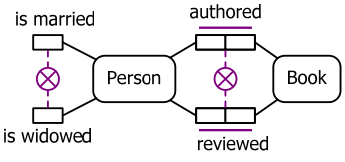
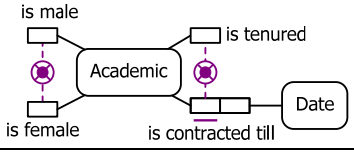
Figure 3.14
Summary of the notation for ER diagrams.

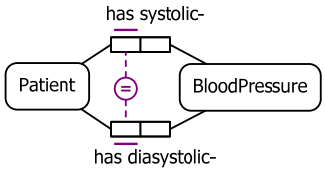
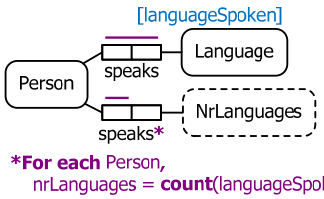
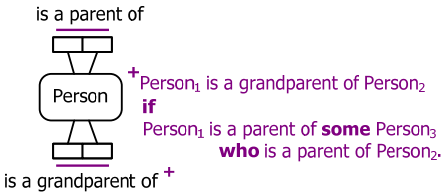
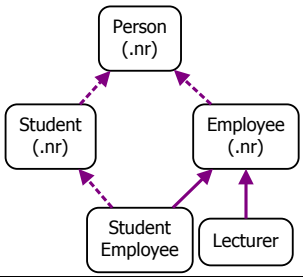
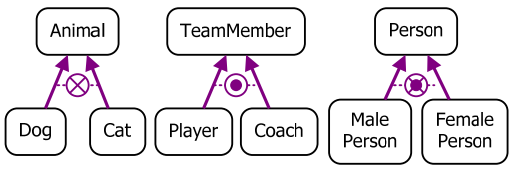
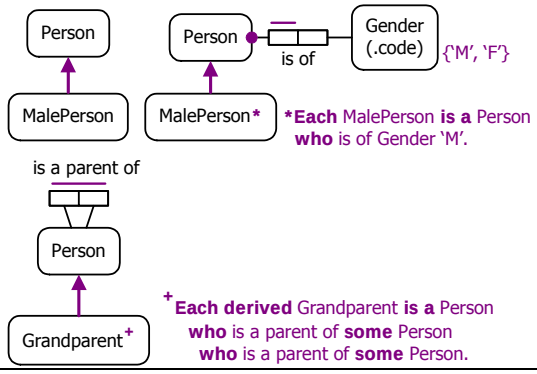
ORM 2 Graphical Notation

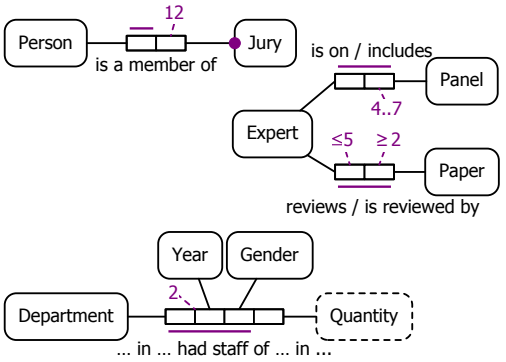
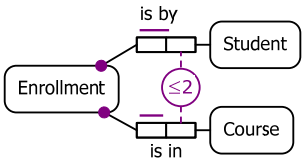
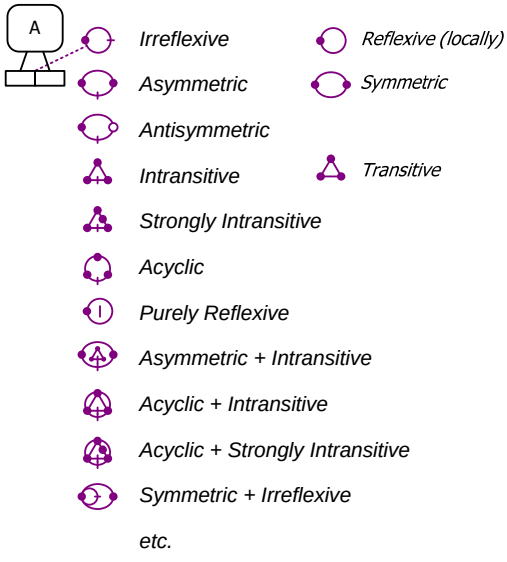
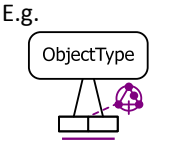
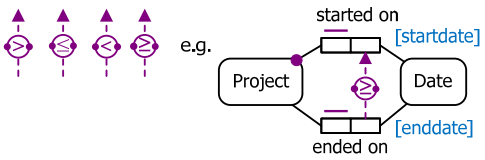
Terry Halpin

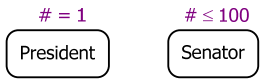
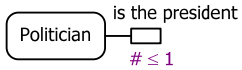
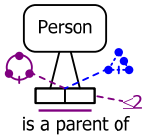
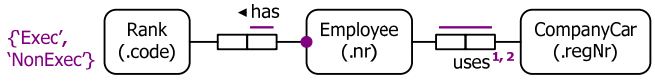
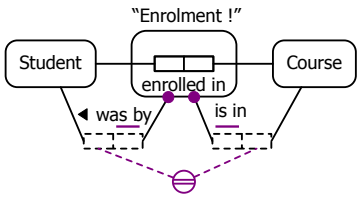
Construct	*Examples*	*Description/Notes*
Entity Type		Named soft rectangle, named hard rectangle, or named ellipse. The soft rectangle shape is the default.
Value Type		Named, dashed, soft rectangle (or hard rectangle or ellipse).
Entity type with popular reference mode		Abbreviation for injective reference 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		Role names may be added in square brackets. Arrow-tips are used to reverse the default left-right or top-down reading order. Reading orders other than forward and reverse are shown using named placeholders.
Quaternary fact type		The above notes for the ternary case apply here also. Fact types of higher arity (number of roles) are also permitted.
Objectification (a.k.a. nesting)		The enrolment fact type is objectified as an entity type whose instances can play roles. In this example, the objectification type is independent, so we can know about an enrolment before the grade is obtained.
Internal uniqueness constraint (UC) on unaries		These are equivalent (by default, predicates are assumed to be populated with sets, so no whole fact may be duplicated).
Internal UC on binaries		The examples show the 4 possible patterns: 1:*n* (one-to-many); *n:*1 (many-to-one); *m:n* (many-to-many); 1:1 (one-to-one0
Internal UC on ternaries. For *n*-aries (n > 1) each UC must span at least *n*-1 roles		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). The second example has a spanning UC (many-to-many-to-many).
Simple mandatory role constraint		The example constraint means that each person was born in some country. The mandatory role dot may be placed at either end of the role connector.
Inclusive-or constraint (disjunctive mandatory role)		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).
Preferred internal UC		A double bar on a UC indicates it underlies the preferred reference scheme.

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		This constraint means that a patient's systolic BP is recorded if and only if his/her diastolic BP is recorded. An equality constraint may apply between 2 or more compatible role sequences, possibly involving joins.
Derived fact type, and derivation rule		A fact type is either asserted, derived, or semiderived. A derived fact type is marked with an asterisk "*". A derivation rule is supplied. A double asterisk "**" indicates derived and stored (eager evaluation).
Semiderived fact type, and derivation rule		A fact type is semiderived if some of its instances may be derived, and some of its instances may be simply asserted. It is marked by "+" (half an asterisk). "++"indicates semiderived and stored (eager evaluation for derived instances).
Subtyping		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.
Subtyping constraints		A circled "X" 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).
Subtype derivation status		A subtype may be • asserted, • derived (denoted by "*"), • or semiderived (denoted by "+"). If the subtype is asserted, it has no mark appended and has no derivation rule. If the subtype derived or semiderived, a derivation rule is supplied.

Construct	*Examples*	*Description/Notes*
Internal frequency constraint		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).
External frequency constraint		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)
Ring constraints	 E.g. 	A ring predicate *R* is locally reflexive if and only if, for all *x* and *y*, *xRy* implies *xRx*. E.g. "knows" is locally but not globally reflexive. Reflexive, symmetric and transitive properties may also be enforced using semiderivation rather than by constraining asserted fact types. 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 *A* can be both a direct subtype of another type *B* and an indirect subtype of *B,* where indirect subtyping means there is a chain of two or more subtyping relationships that lead from *A* to *B*). 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.
Value-comparison constraints		The example constraint verbalizes as: **For each** Project, **existing** enddate >= startdate.

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	*Uniqueness* *Mandatory* *Subset, Equality, Exclusion* *Frequency* ≤2 *Irreflexive* *Acyclic* *Asymmetric* *Asym-Intrans* *Intransitive* *Acyclic-Intrans* *Antisymmetric* *Symmetric* *Strongly Intransitive* *etc.* e.g. 	Unlike alethic constraints, deontic constraint shapes are colored blue rather than violet. Most include “o” for “obligatory”. Deontic ring constraints instead use dashed lines. 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.
Textual constraints	 [1] **Each** Employee **who** has Rank 'NonExec' uses **at most one** CompanyCar. [2] **Each** Employee **who** has Rank 'Exec' uses **some** CompanyCar.	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.