Del A: Modellering og realisering

Eksamen i IN2090 - Databaser og datamodellering og INF1300 - Introduksjon til databaser

6. desember 2018 14:30 – 18:30 (4 timer)

Tillatte hjelpemidler:

- Halpin & Morgan: Information Modelling and Relational Databases. Second Edition.
- 4 håndskrevne A4-sider med notater (2 ark hvis skrevet på begge sider).

Kalkulator er ikke tillatt.

Oppgaven består av to deler, del A (modellering i ORM og realisering) og del B (SQL). Hver del teller 50%.

Faglærer besøker eksamenslokalet omtrent 30 minutter til én time etter at eksamen startet. Det er viktig at du da har lest igjennom hele oppgavesettet slik at du kan spørre om noe i teksten er uklart.

Du bør lese igjennom hele del A før du begynner å modellere, og du bør lese igjennom hele del B før du begynner å svare på oppgavene der.

Ha gjerne med kommentarer i modellen. Om du mener noe er uklart i oppgaveteksten, kan du spesifisere med kommentarer hvilke antakelser du har lagt til grunn.

Oppgave 2 (modelleringsoppgave) skal tegnes for hånd på papir (skisseark) som du får utdelt.
Instruksjon om utfylling av skisseark finner du på pulten din. Husk å notere kodenummeret og annen informasjon med én gang; du vil ikke få tid til å gjøre dette etter at eksamen er over. Det blir IKKE gitt ekstratid for å fylle ut informasjonsboksene på skisseark (engangskoder, kand.nr. o.l.). OBS: denne oppgaven er merket som type "muntlig" i oppgaveoversikten av tekniske årsaker.

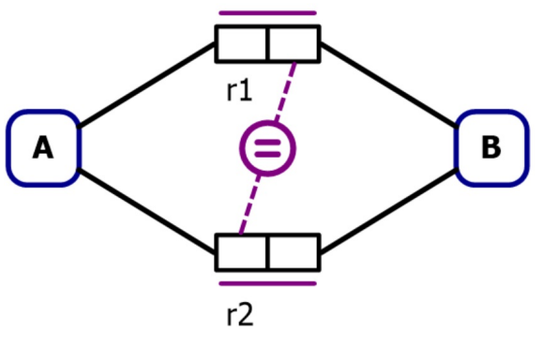
Om du har behov for å tegne noe annet (for andre oppgaver enn modelleringsoppgaven) for å vise sensor hvordan du tenker, så tegner du dette etter svaret på oppgave 2, skriver tydelig på tegningen hvilken/hvilke oppgave(r) tegningen tilhører, og leverer alt som svar på oppgave 2.

Det er kun oppgave 2 som skal løses på ark. Tegn pent og tydelig.

1 Eksterne skranker (5%)

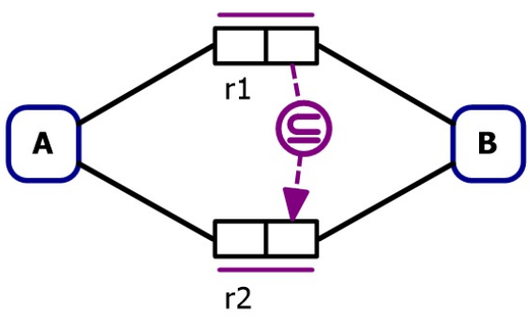
*I denne oppgaven får du **1** poeng for hvert riktige svar. **-1** poeng for feil svar. **0** poeng hvis du ikke svarer.*

I modellene nedenfor (ORM2) skal du anta at alle begreper har en unik representasjon.



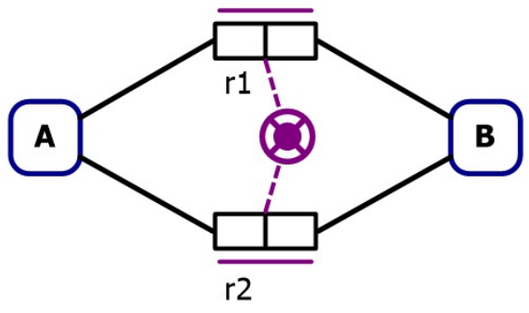
Er plasseringen av den eksterne skranken over gyldig eller ugyldig?

- Gyldig
- Ugyldig



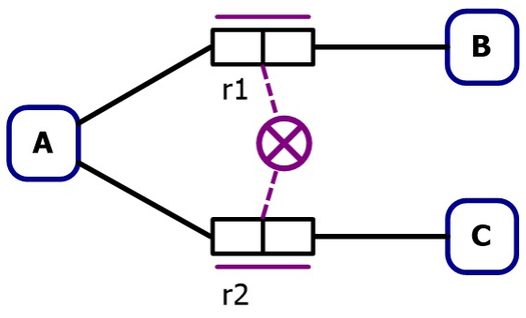
Er plasseringen av den eksterne skranken over gyldig eller ugyldig?

- Gyldig
- Ugyldig



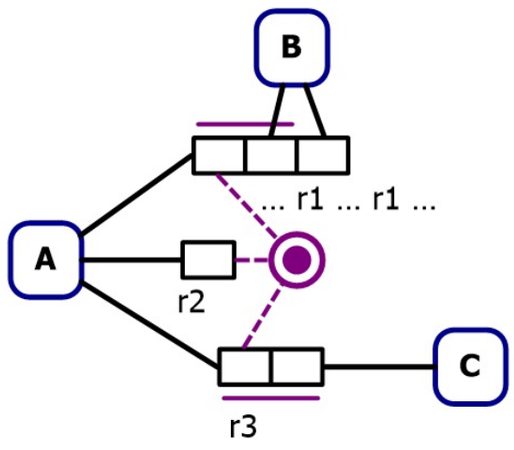
Er plasseringen av den eksterne skranken over gyldig eller ugyldig?

- Gyldig
- Ugyldig



Er plasseringen av den eksterne skranken over gyldig eller ugyldig?

- Gyldig
- Ugyldig



Er plasseringen av den eksterne skranken over gyldig eller ugyldig?

- Gyldig
- Ugyldig

Maks poeng: 5

2 Modellering i ORM (35%)

Denne oppgaven skal du svare med digital håndtegning. Bruk eget skisseark (utdelt). Se instruksjon for utfylling av skisseark på pult. Oppslagsdokumentet ORM2 Graphical Notation er vedlagt.

I denne oppgaven skal du lage en modell for et emneregistreringssystem på et universitet. Systemet skal også håndtere oblig- og eksamensinnleveringer. Modellen skal innfri kravene som er listet opp i oppgavene nedenfor. For å gjøre det mer oversiktlig, kan du dele opp ORM-modellen over flere sider. Det beste er å bruke én side for hver av de to delene, men det hele kan også tegnes som én sammenhengende modell.

Del 1: *Lag ORM2-modell som inneholder denne informasjonen om brukere og emner:*

a. Hver bruker er identifisert av et unikt nummer (f.eks. 483226). I tillegg skal alle brukere ha et brukernavn (f.eks. har bruker 483226 brukernavnet «olanor»). Ingen brukere kan ha samme brukernavn. Vi vil også registrere fullt navn for brukerne, men dette vil kanskje mangle for noen brukere. Flere brukere skal kunne ha samme navn.
b. Brukere kan være registrert i flere emner. Emner er representert av en emnekode. I tillegg skal alle emnene ha et emnenavn, og flere emner kan ha samme navn. Det er ingen øvre grense på hvor mange brukere som kan være registrert i et emne.
c. Noen brukere er gruppelærer for emner, men man kan ikke være gruppelærer på emner som man selv er registrert på. Man kan være gruppelærer for flere emner, og hvert emne kan ha mange gruppelærere.
d. Vi vil også registrere resultater fra tidligere semestre, slik at hver bruker har ett resultat per emne for et gitt semester. For eksempel kan bruker 483226 ha fått karakteren «C» i emnet «IN1010» i semesteret «vår 2017». Det skal være mulig å registrere resultat for samme emne i flere semestre, f.eks. kan også bruker 483226 få registrert «B» i emnet IN1010 vår 2018. Det er mulig for en bruker å ha samme karakter i flere semestre, men en bruker kan ikke ha flere resultat i et emne i samme semester. Semestre representeres med en semesterkode (f.eks. «vår 2018»). Resultat kan være «bestått» eller «ikke bestått», eller en bokstav mellom «A» og «F».
e. Alle brukere som er gruppelærere for emner, må ha fått registrert et resultat i dette emnet.

Del 2: *Lag ORM2-modell som inneholder utvidet informasjon om oppgaver, innleveringer og emner. I denne deloppgaven skal vi ikke ta hensyn til semestre eller resultater.*

a. I systemet vårt ønsker vi å kunne legge inn obliger. Alle obliger gjelder ett emne, og alle obligene har et oblignummer. Samme nummer kan ikke brukes flere ganger i et emne, men kan brukes på tvers av emnene (f.eks. kan IN1010 bare ha én oblig 1, men IN2090 kan også ha oblig 1).
b. Brukere kan levere innleveringer. Alle innleveringene hører til én eller flere brukere (man kan altså levere inn som gruppe). Innleveringene hører til én oblig (fra oppgave a), eller så er innleveringen en eksamensinnlevering, og da gjelder den et emne (fra del 1). Alle innleveringer må enten gjelde en oblig, eller være eksamensinnlevering i et emne, men de kan ikke være begge. Innleveringene må også ha en hensiktsmessig unik representasjon i modellen.
c. Enkelte emner kan ha et visst antall studiepoeng overlapp med andre emner. Hvert emne kan overlappe med flere emner, men et emne skal ikke kunne overlappe med seg selv. Overlappingen skal gå begge veier, dvs. at om IN2090 har 10 studiepoeng overlapp med INF1300, har også INF1300 10 studiepoeng overlapp med IN2090.

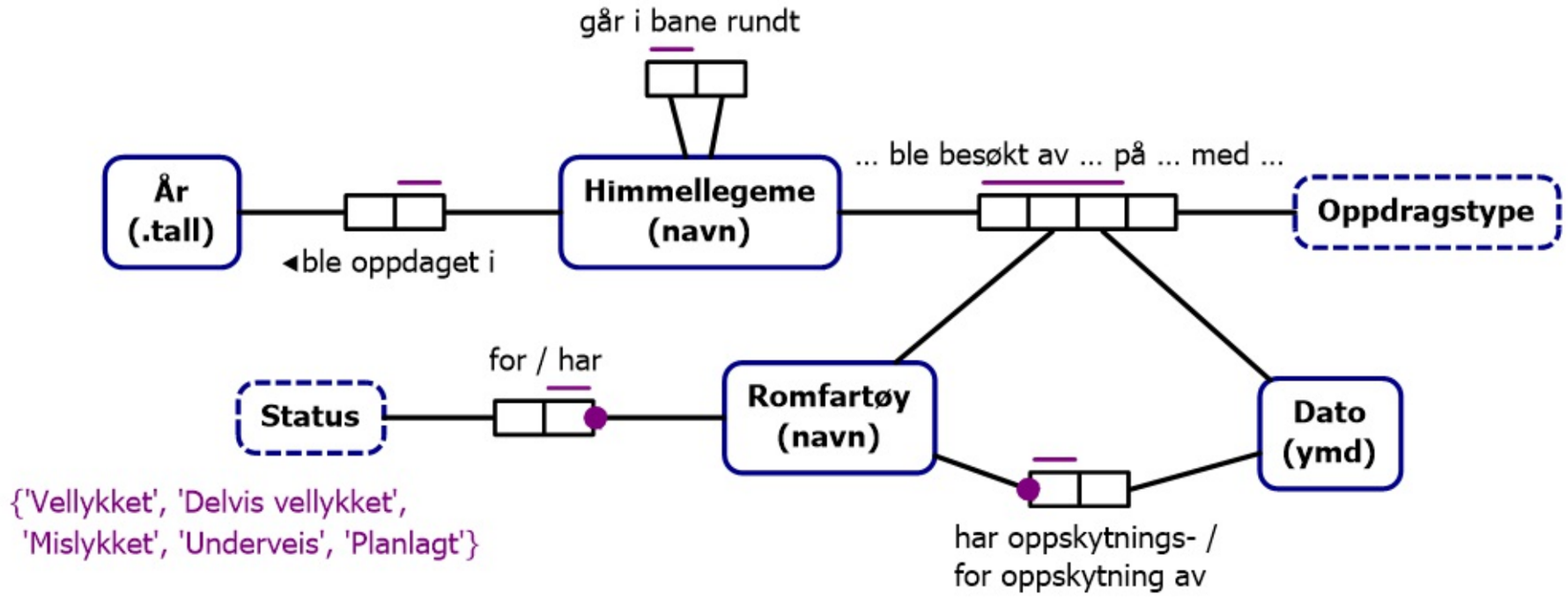
Maks poeng: 35

3 Realisering (10%)

Realiser ORM2-modellen nedenfor til et relasjonsdatabaseskjema.
Relasjonsdatabaseskjemaet skal være korrekt (tilsvare ORM2-modellen), effektivt (unngå redundans og begrens antallet tabeller), og tydelig (lett å forstå). Bruk realiseringsalgoritmen for å danne et slikt relasjonsdatabaseskjema.

For hver relasjon, spesifiser relasjonens navn og navnet til hvert attributt. Du skal ikke spesifisere datatyper/domener for attributtene, og ikke benytte SQL i denne oppgaven. Marker primærnøkler med understreking. Marker andre kandidatnøkler med **fete typer**. Marker fremmednøkler på denne formen:

FraRelasjon(fraAttributt) → TilRelasjon(TilAttributt)



Skriv ditt svar her...

Maks poeng: 10

Del B: SQL og relasjonsteori (50%)

I oppgave 4-11 skal du jobbe med tabellene beskrevet nedenfor. Beskrivelsen er gjengitt i alle oppgavene. Primærnøkler er understreket.

products(id, name, description)
customers(id, pnr, fdate, name, address, country, time_registered)
orders(id, customer_id, shipping_addr, time_entered, status)
order_items(order_id, product_id, quantity, unit_price)

I tabellen *products* er id en int, og de andre attributtene strenger, henholdsvis for å lagre navn og beskrivelse.

I tabellen *customers* lagres informasjon om kunder. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, mens time_registered har typen timestamp. time_registered er et tidsstempel for når en kunde først ble registrert i databasen. Alle kunder har en unik kombinasjon av pnr og fdate.

I tabellen *orders* lagres data om enkeltordrer. id er en int, customer_id er fremmednøkkel til customers, shipping_addr en streng, time_entered en timestamp som sier når en ordre kom inn, og status er en streng for å lagre ordrens status (for eksempel "shipped" eller "delayed").

I tabellen *order_items* lagres data om enkeltposter på en ordre. order_id er fremmednøkkel til orders, product_id er en fremmednøkkel til products, mens quantity og unit_price er av typen int, og sier hvor mye av dette produktet ble bestilt og til hvilken enhetspris i denne ordren.

4 SQL, create (5%)

I denne delen av eksamen skal vi jobbe med følgende skjema for en liten ordredatabase. Primærnøkler er understreket.

products(id, name, description)
customers(id, pnr, fdate, name, address, country, time_registered)
orders(id, customer_id, shipping_addr, time_entered, status)
order_items(order_id, product_id, quantity, unit_price)

I tabellen *products* er id en int, og de andre attributtene strenger, henholdsvis for å lagre navn og beskrivelse.

I tabellen *customers* lagres informasjon om kunder. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, mens time_registered har

typen timestamp. time_registered er et tidsstempel for når en kunde først ble registrert i databasen. Alle kunder har en unik kombinasjon av pnr og fdate.

I tabellen *orders* lagres data om enkeltordrer. id er en int, customer_id er fremmednøkkel til customers, shipping_addr en streng, time_entered en timestamp som sier når en ordre kom inn, og status er en streng for å lagre ordrens status (for eksempel "shipped" eller "delayed").

I tabellen *order_items* lagres data om enkeltposter på en ordre. order_id er fremmednøkkel til orders, product_id er en fremmednøkkel til products, mens quantity og unit_price er av typen int, og sier hvor mye av dette produktet ble bestilt og til hvilken enhetspris i denne ordren.

Oppgave: Skriv en CREATE-spørring for tabellen *orders* med passende datatyper for attributtene, slik at tabellen blir som beskrevet, inkludert primær- og fremmednøkler.

Skriv ditt svar her...

Maks poeng: 5

5 SQL, view (5%)

I denne delen av eksamen skal vi jobbe med følgende skjema for en liten ordredatabase. Primærnøkler er understreket.

products(id, name, description)
customers(id, pnr, fdate, name, address, country, time_registered)
orders(id, customer_id, shipping_addr, time_entered, status)
order_items(order_id, product_id, quantity, unit_price)

I tabellen *products* er id en int, og de andre attributtene strenger, henholdsvis for å lagre navn og beskrivelse.

I tabellen *customers* lagres informasjon om kunder. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, mens time_registered har typen timestamp. time_registered er et tidsstempel for når en kunde først ble registrert i databasen. Alle kunder har en unik kombinasjon av pnr og fdate.

I tabellen *orders* lagres data om enkeltordrer. id er en int, customer_id er fremmednøkkel til customers, shipping_addr en streng, time_entered en timestamp som sier når en ordre kom inn, og status er en streng for å lagre ordrens status (for eksempel "shipped" eller "delayed").

I tabellen *order_items* lagres data om enkeltposter på en ordre. order_id er fremmednøkkel til orders, product_id er en fremmednøkkel til products, mens quantity og unit_price er av typen int, og sier hvor mye av dette produktet ble bestilt og til hvilken enhetspris i denne ordren.

Oppgave: Definer et view delayed_orders(id, customer_id, time_entered) som inneholder alle ordrer som har status 'delayed'.

Skriv ditt svar her...

Maks poeng: 5

6 SQL, kunder og ordrer (5%)

I denne delen av eksamen skal vi jobbe med følgende skjema for en liten ordredatabase. Primærnøkler er understreket.

products(id, name, description)
customers(id, pnr, fdate, name, address, country, time_registered)
orders(id, customer_id, shipping_addr, time_entered, status)
order_items(order_id, product_id, quantity, unit_price)

I tabellen *products* er id en int, og de andre attributtene strenger, henholdsvis for å lagre navn og beskrivelse.

I tabellen *customers* lagres informasjon om kunder. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, mens time_registered har typen timestamp. time_registered er et tidsstempel for når en kunde først ble registrert i databasen. Alle kunder har en unik kombinasjon av pnr og fdate.

I tabellen *orders* lagres data om enkeltordrer. id er en int, customer_id er fremmednøkkel til customers, shipping_addr en streng, time_entered en timestamp som sier når en ordre kom inn, og status er en streng for å lagre ordrens status (for eksempel "shipped" eller "delayed").

I tabellen *order_items* lagres data om enkeltposter på en ordre. order_id er fremmednøkkel til orders, product_id er en fremmednøkkel til products, mens quantity og unit_price er av typen int, og sier hvor mye av dette produktet ble bestilt og til hvilken enhetspris i denne ordren.

Oppgave: Skriv en spørring som returnerer alle kunder der navnet begynner med «Ola», samt deres ordrer hvis de har noen. Ta også med kunder som ikke har kjøpt noe. Resultatet skal inneholde kundens navn og adresse, samt ordrenes id og status, sortert etter registreringstidspunkt slik at de siste registrerte kundene kommer først.

Skriv ditt svar her...

Maks poeng: 5

7 SQL, aggregering (5%)

I denne delen av eksamen skal vi jobbe med følgende skjema for en liten ordredatabase. Primærnøkler er understreket.

products(id, name, description)
customers(id, pnr, fdate, name, address, country, time_registered)
orders(id, customer_id, shipping_addr, time_entered, status)
order_items(order_id, product_id, quantity, unit_price)

I tabellen *products* er id en int, og de andre attributtene strenger, henholdsvis for å lagre navn og beskrivelse.

I tabellen *customers* lagres informasjon om kunder. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, mens time_registered har typen timestamp. time_registered er et tidsstempel for når en kunde først ble registrert i databasen. Alle kunder har en unik kombinasjon av pnr og fdate.

I tabellen *orders* lagres data om enkeltordrer. id er en int, customer_id er fremmednøkkel til customers, shipping_addr en streng, time_entered en timestamp som sier når en ordre kom inn, og status er en streng for å lagre ordrens status (for eksempel "shipped" eller "delayed").

I tabellen *order_items* lagres data om enkeltposter på en ordre. order_id er fremmednøkkel til orders, product_id er en fremmednøkkel til products, mens quantity og unit_price er av typen int, og sier hvor mye av dette produktet ble bestilt og til hvilken enhetspris i denne ordren.

Oppgave: Skriv en spørring som finner antall ordrer per kunde for hver forskjellig status. Skriv ut kundens id og navn, samt antall og status.

Skriv ditt svar her...

Maks poeng: 5

8 SQL, kunder med flere ordrer (5%)

I denne delen av eksamen skal vi jobbe med følgende skjema for en liten ordredatabase. Primærnøkler er understreket.

products(id, name, description)
customers(id, pnr, fdate, name, address, country, time_registered)
orders(id, customer_id, shipping_addr, time_entered, status)
order_items(order_id, product_id, quantity, unit_price)

I tabellen *products* er id en int, og de andre attributtene strenger, henholdsvis for å lagre navn og beskrivelse.

I tabellen *customers* lagres informasjon om kunder. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, mens time_registered har typen timestamp. time_registered er et tidsstempel for når en kunde først ble registrert i databasen. Alle kunder har en unik kombinasjon av pnr og fdate.

I tabellen *orders* lagres data om enkeltordrer. id er en int, customer_id er fremmednøkkel til customers, shipping_addr en streng, time_entered en timestamp som sier når en ordre kom inn, og status er en streng for å lagre ordrens status (for eksempel "shipped" eller "delayed").

I tabellen *order_items* lagres data om enkeltposter på en ordre. order_id er fremmednøkkel til orders, product_id er en fremmednøkkel til products, mens quantity og unit_price er av typen int, og sier hvor mye av dette produktet ble bestilt og til hvilken enhetspris i denne ordren.

Oppgave: Skriv en spørring som finner alle kunder som har minst to ordrer med forskjellig shipping_address. Skriv ut kundenes id og navn.

Skriv ditt svar her...

Maks poeng: 5

9 SQL, produkter bestilt oftest (10%)

I denne delen av eksamen skal vi jobbe med følgende skjema for en liten ordredatabase. Primærnøkler er understreket.

products(id, name, description)
customers(id, pnr, fdate, name, address, country, time_registered)
orders(id, customer_id, shipping_addr, time_entered, status)
order_items(order_id, product_id, quantity, unit_price)

I tabellen *products* er id en int, og de andre attributtene strenger, henholdsvis for å lagre navn og beskrivelse.

I tabellen *customers* lagres informasjon om kunder. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, mens time_registered har typen timestamp. time_registered er et tidsstempel for når en kunde først ble registrert i databasen. Alle kunder har en unik kombinasjon av pnr og fdate.

I tabellen *orders* lagres data om enkeltordrer. id er en int, customer_id er fremmednøkkel til customers, shipping_addr en streng, time_entered en timestamp som sier når en ordre kom inn, og status er en streng for å lagre ordrens status (for eksempel "shipped" eller "delayed").

I tabellen *order_items* lagres data om enkeltposter på en ordre. order_id er fremmednøkkel til orders, product_id er en fremmednøkkel til products, mens quantity og unit_price er av typen int, og sier hvor mye av dette produktet ble bestilt og til hvilken enhetspris i denne ordren.

Oppgave: Skriv en spørring som returnerer alle kunder som har bestilt de produktene som har vært bestilt oftest (altså de produktene som finnes på flest forskjellige ordrer). Skriv ut kundenes id.

Husk å ta hensyn til at flere produkter kan ha vært bestilt oftest. F.eks. om produkt A og produkt B

begge har vært bestilt 500 ganger og dermed forekommer på flest ordrer, skal alle kunder som har bestilt enten produkt A eller produkt B skrives ut.

Skriv ditt svar her...

Maks poeng: 10

10 SQL, total kostnad (10%)

I denne delen av eksamen skal vi jobbe med følgende skjema for en liten ordredatabase. Primærnøkler er understreket.

products(id, name, description)
customers(id, pnr, fdate, name, address, country, time_registered)
orders(id, customer_id, shipping_addr, time_entered, status)
order_items(order_id, product_id, quantity, unit_price)

I tabellen *products* er id en int, og de andre attributtene strenger, henholdsvis for å lagre navn og beskrivelse.

I tabellen *customers* lagres informasjon om kunder. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, mens time_registered har typen timestamp. time_registered er et tidsstempel for når en kunde først ble registrert i databasen. Alle kunder har en unik kombinasjon av pnr og fdate.

I tabellen *orders* lagres data om enkeltordrer. id er en int, customer_id er fremmednøkkel til customers, shipping_addr en streng, time_entered en timestamp som sier når en ordre kom inn, og status er en streng for å lagre ordrens status (for eksempel "shipped" eller "delayed").

I tabellen *order_items* lagres data om enkeltposter på en ordre. order_id er fremmednøkkel til orders, product_id er en fremmednøkkel til products, mens quantity og unit_price er av typen int, og sier hvor mye av dette produktet ble bestilt og til hvilken enhetspris i denne ordren.

Oppgave: Skriv en spørring som finner antall enheter og total kostnad for alle produkter solgt til kunder i USA, hvor minst 10 enheter har blitt solgt. Skriv ut produktenes id og navn, samt antall enheter og total kostnad.

Et eksempel finnes under tekstboksen.

Skriv ditt svar her...

Eksempel: om vi har følgende data i *order_lines* for kunder i USA:

order_id	product_id	quantity	unit_price
1	1	9	100
1	2	15	200
5	3	5	25
12	1	1	200

Så kan resultatet se slik ut (kolonnenavnene velger du selv):

product_id	name	antall	kostnad
1	*navn her*	10	1100
2	*navn her*	15	3000

Maks poeng: 10

11 Relasjonsteori (5%)

I denne oppgaven ser vi kun på tabellen *customers*. Beskrivelsen av tabellen er lik som i de andre oppgavene. Primærnøkkelen er understreket.

customers(id, pnr, fdate, name, address, country, time_registered)

I tabellen *customers* lagres informasjon om kunder. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, mens time_registered har typen timestamp. time_registered er et tidsstempel for når en kunde først ble registrert i databasen. Alle kunder har en unik kombinasjon av pnr og fdate.

Oppgave:

a. Hvilke attributter i tabellen *customers* er kandidatnøkler?
b. Hvilken normalform er tabellen *customers* på? Begrunn svaret ditt.

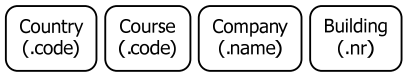
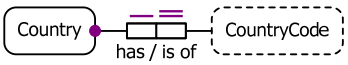
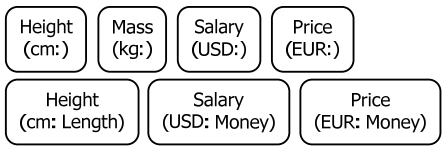
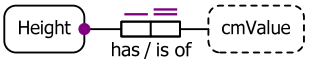
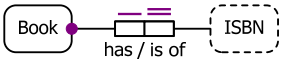
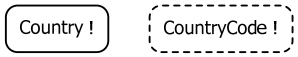
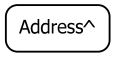
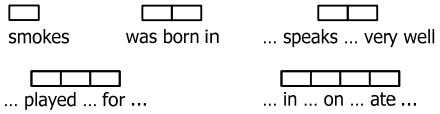
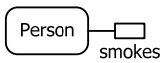
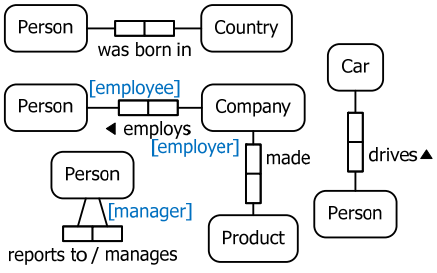
Skriv ditt svar her...

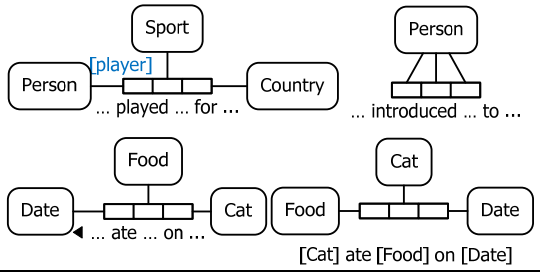
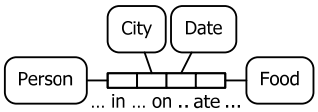
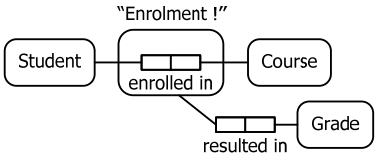
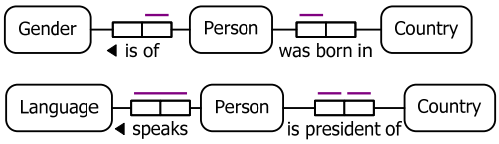
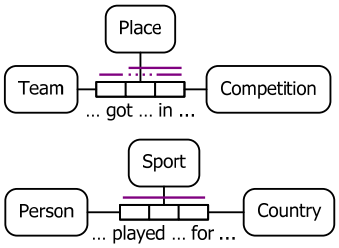
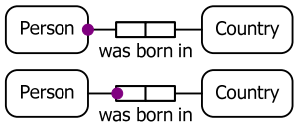
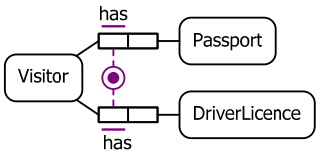
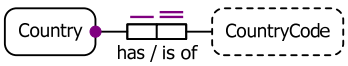
Maks poeng: 5

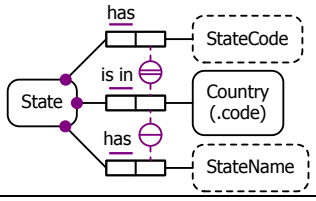
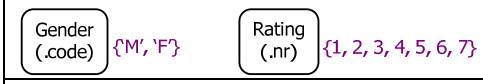
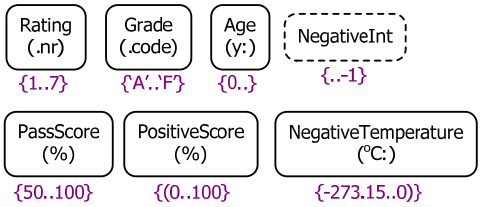
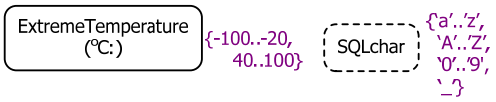
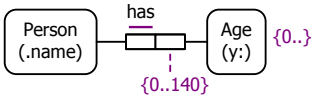
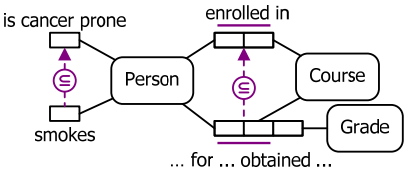
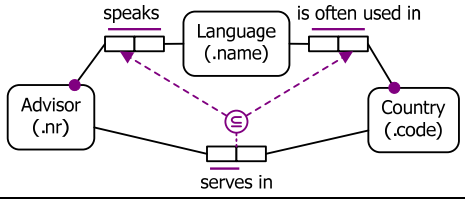
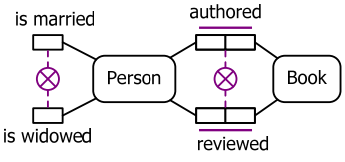
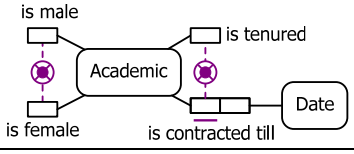
Question 2
Attached

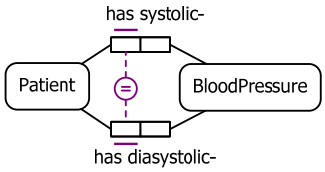
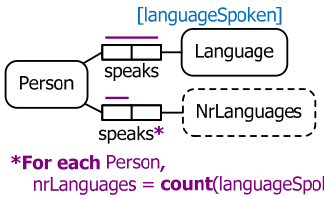
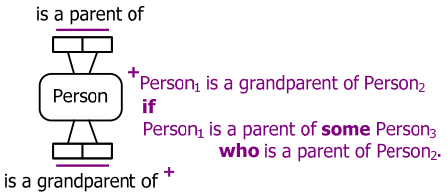
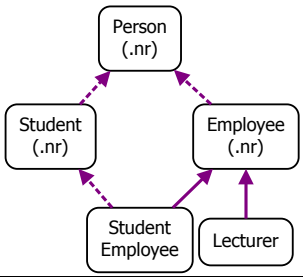
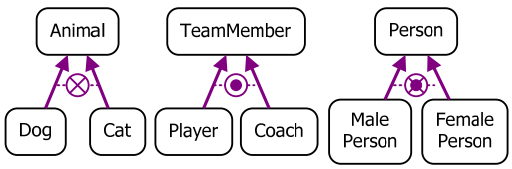
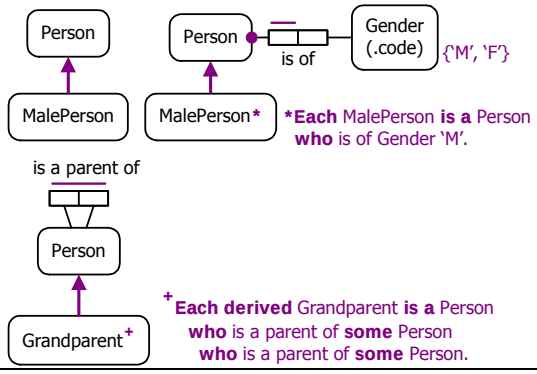
ORM 2 Graphical Notation

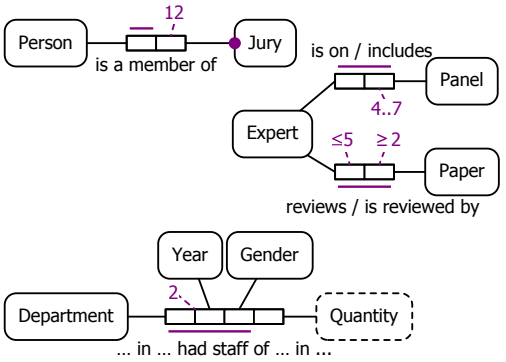
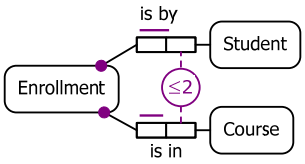
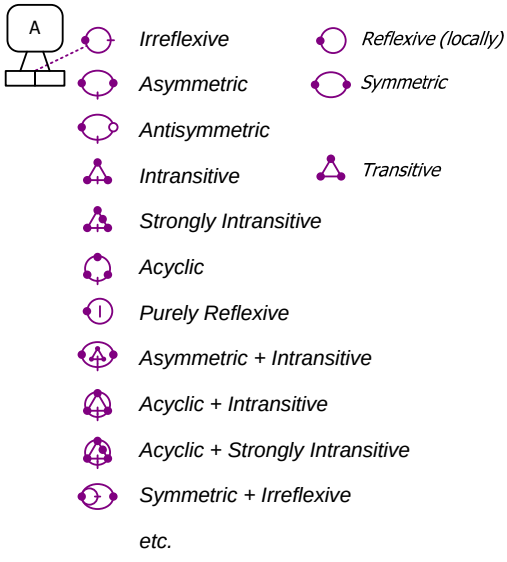
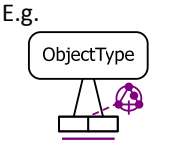
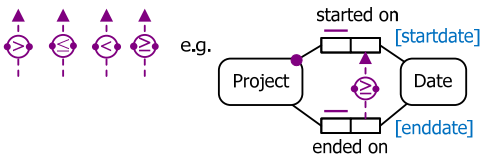
Terry Halpin

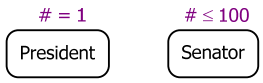
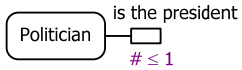
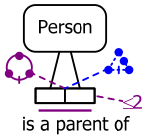
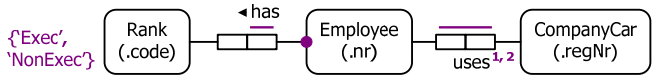
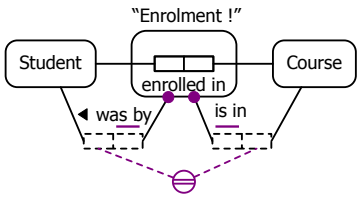
Construct	*Examples*	*Description/Notes*
Entity Type		Named soft rectangle, named hard rectangle, or named ellipse. The soft rectangle shape is the default.
Value Type		Named, dashed, soft rectangle (or hard rectangle or ellipse).
Entity type with popular reference mode		Abbreviation for injective reference 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.