Del A: Modellering og realisering

Eksamen i IN2090 - Databasar og datamodellering og INF1300 - Introduksjon til databasar

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

Tillatne hjelpemiddel:

- Halpin & Morgan: Information Modelling and Relational Databases. Second Edition.
- 4 handskrivne A4-sider med notat (2 ark viss skrive på begge sider).

Kalkulator er ikkje tillaten.

Oppgåva består av to delar, del A (modellering i ORM og realisering) og del B (SQL). Kvar del tel 50%.

Faglærer vitjar eksamenslokalet omtrent 30 minutt til ein time etter at eksamen starta. Det er viktig at du då har lese igjennom heile oppgåvesettet slik at du kan spørja om noko i teksten er uklart.

Du bør lese igjennom heile del A før du tek til å modellere, og du bør lese igjennom heile del B før du tek til å svare på oppgåvene der.

Ha gjerne med kommentarar i modellen. Om du meiner noko er uklart i oppgåveteksta, kan du spesifisere med kommentarar kva føresetnadar du har lagt til grunn.

Oppgåve 2 (modelleringsoppgåve) skal teiknast for hånd på papir (skisseark) som du får utdelt.
Instruksjon om utfylling av skisseark finn du på pulten din. Hugs å notera kodenummeret og annan informasjon med ein gong; du vil ikkje få tid til å gjera dette etter at eksamen er over. Det vert IKKJE gjeve ekstratid for å fylla ut informasjonsboksane på skisseark (engangskodar, kand.nr. o.l.). OBS: denne oppgåva er merket som type "munnleg" i oppgåveoversynet av tekniske årsaker.

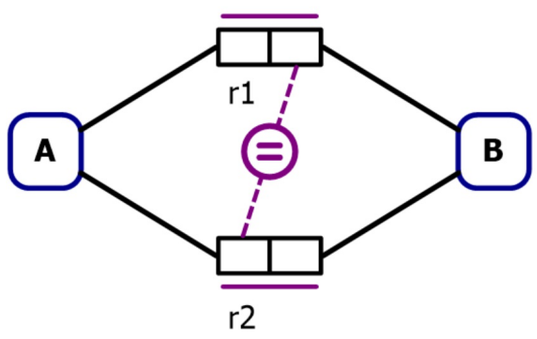
Om du har behov for å teikna noko anna (for andre oppgåver enn modeleringsoppgaven) for å visa sensor korleis du tenkjer, så teiknar du dette etter svaret på oppgåve 2, skriv tydeleg på teikninga kva for oppgåve(r) teikninga tilhøyrar, og leverer alt som svar på oppgåve 2.

Det er berre oppgåve 2 som skal løysast på ark. Teikn pent og tydeleg.

1 Eksterne skranker (5%)

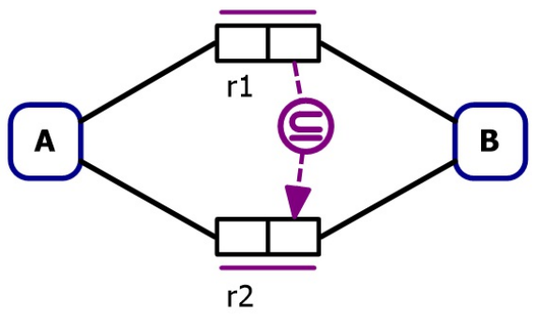
*I denne oppgåva får du **1** poeng for kvart riktige svar. **-1** poeng for feil svar. **0** poeng viss du ikkje svarar.*

I modellane nedanfor (ORM2) skal du anta at alle begrep har ein unik representasjon.



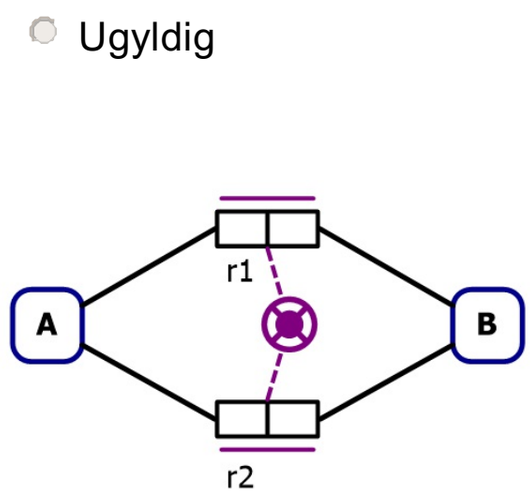
Er plasseringa av den eksterne skranken over gyldig eller ugyldig?

- Gyldig
- Ugyldig



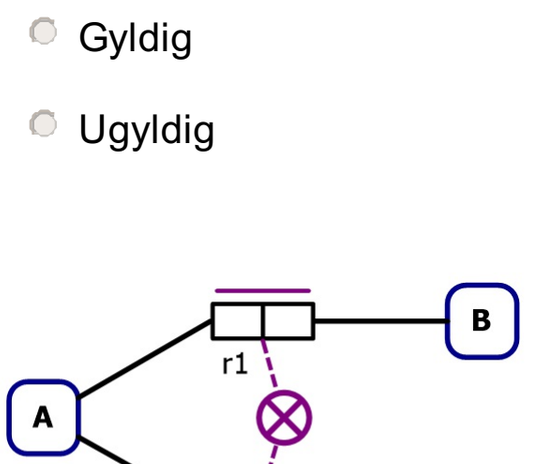
Er plasseringa av den eksterne skranken over gyldig eller ugyldig?

- Gyldig
- Ugyldig



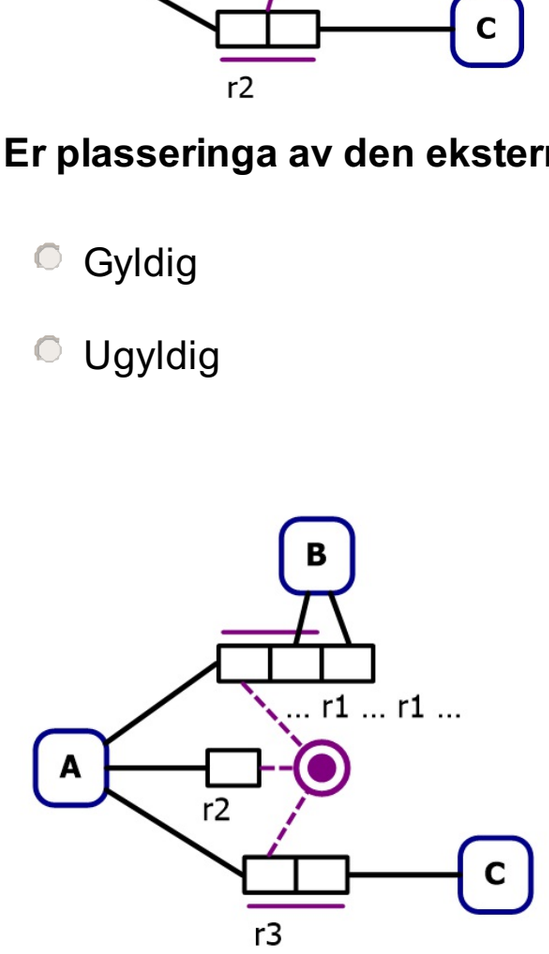
Er plasseringa av den eksterne skranken over gyldig eller ugyldig?

- Gyldig
- Ugyldig



Er plasseringa av den eksterne skranken over gyldig eller ugyldig?

- Gyldig
- Ugyldig



Er plasseringa av den eksterne skranken over gyldig eller ugyldig?

- Gyldig
- Ugyldig

Maks poeng: 5

2 Modellering i ORM (35%)

Denne oppgåva skal du svara med digital handteikning. Bruk eige skisseark (utdelt). Sjå instruksjon for utfylling av skisseark på pult. Oppslagsdokumentet ORM2 Graphical Notation er vedlagt.

I denne oppgåva skal du laga ein modell for eit emneregistreringssystem på eit universitet. Systemet skal òg handtere oblig- og eksamensinnleveringar. Modellen skal innfri krava som er lista opp i oppgåvene nedanfor. For å gjere det meir oversiktleg, kan du dela opp ORM-modellen over fleire sider. Det beste er å bruka ein side for kvar av dei to delane, men det heile kan òg teiknast som ein samanhengande modell.

Del 1: *Lag ORM2-modell som inneheld denne informasjonen om brukarar og emnar:*

a. Kvar brukar er identifisert av eit unikt nummer (t.d. 483226). I tillegg skal alle brukarar ha eit brukarnamn (t.d. har brukar 483226 brukarnamnet «olanor»). Ingen brukarar kan ha same brukarnamn. Vi vil òg registrera fullt namn for brukarane, men dette vil kanskje mangla for nokre brukarar. Fleire brukarar skal kunne ha same namn.
b. Brukarar kan registrerast i fleire emne. Emne er representert av ein emnekode. I tillegg skal alle emna ha eit emnenamn, og fleire emnar kan ha same namn. Det er ingen øvre grense på kor mange brukarar som kan registrerast i eit emne.
c. Nokre brukarar er gruppelærar for emne, men ein kan ikkje vera gruppelærar på emne som ein sjølv er registrert på. Man kan vera gruppelærar for fleire emne, og kvart emne kan ha mange gruppelærarar.
d. Vi vil òg registrera resultat frå tidlegare semester, slik at kvar brukar har eitt resultat per emne for eit gitt semester. Til dømes kan brukar 483226 ha fått karakteren «C» i emnet IN1010 i semesteret vår 2017. Det skal vera mogleg å registrera resultat for same emne i fleire semester, t.d. kan òg brukar 483226 få registrert «B» i emnet IN1010 vår 2018. Det er mogleg for ein brukar å ha same karakter i fleire semester, men ein brukar kan ikkje ha fleire resultat i eit emne i same semester. Semester vert representert med ein semesterkode (t.d. «vår 2018»). Resultat kan vera «bestått» eller «ikkje bestått», eller ein bokstav mellom «A» og «F».
e. Alle brukarar som er gruppelærarar for emne, må ha fått registrert eit resultat i dette emnet.

Del 2: *Lag ORM2-modell som inneheld utvida informasjon om oppgåver, innleveringar og emnar. I denne deloppgåven skal vi ikkje ta omsyn til semester eller resultat.*

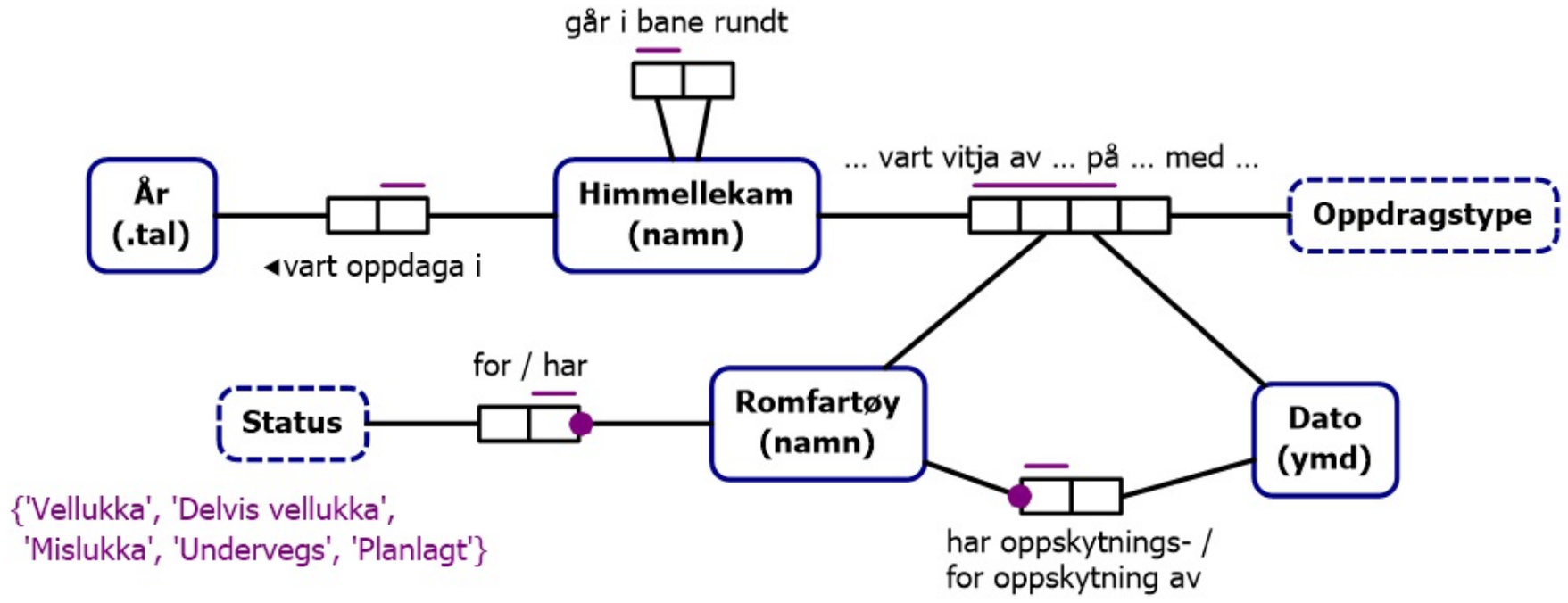
a. I systemet vårt ønskjer vi å kunne leggje inn obligar. Alle obligar gjeld eitt emne, og alle obligane har eit oblignummer. Same nummer kan ikkje brukast fleire gonger i eit emne, men kan brukast på tvers av emna (t.d. kan IN1010 berre ha ein oblig 1, men IN2090 kan òg ha oblig 1).
b. Brukarar kan levere innleveringar. Alle innleveringane høyrer til ein eller fleire brukarar (ein kan altså levere inn som gruppe). Innleveringane høyrer til ein oblig (frå oppgåve a), eller så er innleveringa ein eksamensinnlevering, og då gjeld han eit emne (frå del 1). Alle innleveringar må anten gjelde ein oblig, eller vera eksamensinnlevering i eit emne, men dei kan ikkje vera begge. Innleveringane må òg ha en føremålstenleg unik representasjon i modellen.
c. Enkelte emne kan ha ei viss mengde studiepoeng overlapp med andre emne. Kvart emne kan overlappe med fleire emne, men eit emne skal ikkje kunne overlappe med seg sjølv. Overlappinga skal gå begge vegar, dvs. at om IN2090 har 10 studiepoeng overlapp med INF1300, har òg INF1300 10 studiepoeng overlapp med IN2090.

Maks poeng: 35

3 Realisering (10%)

Realiser ORM2-modellen nedanfor til eit relasjonsdatabaseskjema.
Relasjonsdatabaseskjemaet skal vera korrekt (svarar til ORM2-modellen), effektivt (unngå redundans og avgrens mengda tabellar), og tydeleg (lett å forstå). Bruk realiseringsalgoritmen for å danne eit slikt relasjonsdatabaseskjema.

For kvar relasjon, spesifiser relasjonens namn og namnet til kvart attributt. Du skal ikkje spesifisere datatype/domene for attributta, og ikkje bruke SQL i denne oppgåva. Marker primærnøklar med <u>understreking</u>. Marker andre kandidatnøklar med **feite typar**. Marker framandnøklar på denne forma:
FråRelasjon(fråAttributt) → TilRelasjon(TilAttributt)



Skriv svaret ditt her...

Maks poeng: 10

Del B: SQL og relasjonsteori (50%)

I oppgåve 4-11 skal du jobbe med tabellane beskrive nedanfor. Beskrivinga er attgjeven i alle oppgåvene. Primærnøklar er understreka.

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 ein int, og dei andre attributta strenger, høvesvis for å lagre namn og skildring.

I tabellen *customers* vert informasjon om kundar lagra. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, medan time_registered har typen timestamp. time_registered er eit tidsstempel for når ein kunde først vart registrert i databasen. Alle kundar har ein unik kombinasjon av pnr og fdate.

I tabellen *orders* vert data om enkeltordrar lagra. id er ein int, customer_id er framandnøkkel til customers, shipping_addr ein streng, time_entered ein timestamp som seier når ein ordre kom inn, og status er ein streng for å lagre statusen til ordren (til dømes «shipped» eller «delayed»).

I tabellen *order_items* vert data om enkeltpostar på ein ordre lagra. order_id er framandnøkkel til orders, product_id er ein framandnøkkel til products, medan quantity og unit_price er av typen int, og seier kor mange av dette produktet vart bestilt og prisen for ei eining av produktet i denne ordren.

4 SQL, create (5%)

I denne delen av eksamen skal vi jobbe med følgjande skjema for ein liten ordredatabase.
Primærnøklar er understreka.

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 ein int, og dei andre attributta strenger, høvesvis for å lagre namn og skildring.

I tabellen *customers* vert informasjon om kundar lagra. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, medan time_registered har typen timestamp. time_registered er eit tidsstempel for når ein kunde først vart registrert i databasen. Alle kundar har ein unik kombinasjon av pnr og fdate.

I tabellen *orders* vert data om enkeltordrar lagra. id er ein int, customer_id er framandnøkkel til customers, shipping_addr ein streng, time_entered ein timestamp som seier når ein ordre kom inn, og status er ein streng for å lagre statusen til ordren (til dømes «shipped» eller «delayed»).

I tabellen *order_items* vert data om enkeltpostar på ein ordre lagra. order_id er framandnøkkel til orders, product_id er ein framandnøkkel til products, medan quantity og unit_price er av typen int, og seier kor mange av dette produktet vart bestilt og prisen for ei eining av produktet i denne ordren.

Oppgåve: Skriv ein CREATE-spørring for tabellen *orders* med passande datatypar for attributta, slik at tabellen vert som beskriven, inkludert primær- og framandnøklar.

Skriv svaret ditt her...

Maks poeng: 5

5 SQL, view (5%)

I denne delen av eksamen skal vi jobbe med følgjande skjema for ein liten ordredatabase.
Primærnøklar er understreka.

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 ein int, og dei andre attributta strenger, høvesvis for å lagre namn og skildring.

I tabellen *customers* vert informasjon om kundar lagra. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, medan time_registered har typen timestamp. time_registered er eit tidsstempel for når ein kunde først vart registrert i databasen. Alle kundar har ein unik kombinasjon av pnr og fdate.

I tabellen *orders* vert data om enkeltordrar lagra. id er ein int, customer_id er framandnøkkel til customers, shipping_addr ein streng, time_entered ein timestamp som seier når ein ordre kom inn, og status er ein streng for å lagre statusen til ordren (til dømes «shipped» eller «delayed»).

I tabellen *order_items* vert data om enkeltpostar på ein ordre lagra. order_id er framandnøkkel til orders, product_id er ein framandnøkkel til products, medan quantity og unit_price er av typen int, og seier kor mange av dette produktet vart bestilt og prisen for ei eining av produktet i denne ordren.

Oppgåve: Definer eit view delayed_orders(id, customer_id, time_entered) som inneheld alle ordrar som har status 'delayed'.

Skriv svaret ditt her...

Maks poeng: 5

6 SQL, kunder og ordrer (5%)

I denne delen av eksamen skal vi jobbe med følgjande skjema for ein liten ordredatabase.
Primærnøklar er understreka.

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 ein int, og dei andre attributta strenger, høvesvis for å lagre namn og skildring.

I tabellen *customers* vert informasjon om kundar lagra. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, medan time_registered har typen timestamp. time_registered er eit tidsstempel for når ein kunde først vart registrert i databasen. Alle kundar har ein unik kombinasjon av pnr og fdate.

I tabellen *orders* vert data om enkeltordrar lagra. id er ein int, customer_id er framandnøkkel til customers, shipping_addr ein streng, time_entered ein timestamp som seier når ein ordre kom inn, og status er ein streng for å lagre statusen til ordren (til dømes «shipped» eller «delayed»).

I tabellen *order_items* vert data om enkeltpostar på ein ordre lagra. order_id er framandnøkkel til orders, product_id er ein framandnøkkel til products, medan quantity og unit_price er av typen int, og seier kor mange av dette produktet vart bestilt og prisen for ei eining av produktet i denne ordren.

Oppgåve: Skriv ei spørring som returnerer alle kundar der namnet byrjar med «Ola», og ordrane deira viss dei har nokon. Ta òg med kundar som ikkje har kjøpt noko. Resultatet skal innehalde namnet og adressa til kunden, og id og status til ordrane, sortert etter registreringstidspunkt slik at dei siste registrerte kundane kjem først.

Skriv svaret ditt her...

Maks poeng: 5

7 SQL, aggregering (5%)

I denne delen av eksamen skal vi jobbe med følgjande skjema for ein liten ordredatabase. Primærnøklar er understreka.

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 ein int, og dei andre attributta strenger, høvesvis for å lagre namn og skildring.

I tabellen *customers* vert informasjon om kundar lagra. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, medan time_registered har typen timestamp. time_registered er eit tidsstempel for når ein kunde først vart registrert i databasen. Alle kundar har ein unik kombinasjon av pnr og fdate.

I tabellen *orders* vert data om enkeltordrar lagra. id er ein int, customer_id er framandnøkkel til customers, shipping_addr ein streng, time_entered ein timestamp som seier når ein ordre kom inn, og status er ein streng for å lagre statusen til ordren (til dømes «shipped» eller «delayed»).

I tabellen *order_items* vert data om enkeltpostar på ein ordre lagra. order_id er framandnøkkel til orders, product_id er ein framandnøkkel til products, medan quantity og unit_price er av typen int, og seier kor mange av dette produktet vart bestilt og prisen for ei eining av produktet i denne ordren.

Oppgåve: Skriv ein spørring som finn kor mange ordrar kvar kunde har for kvar ulike status. Skriv ut id og namn til kunden, antal ordre og statusen det gjeld.

Skriv svaret ditt her...

Maks poeng: 5

8 SQL, kunder med flere ordrer (5%)

I denne delen av eksamen skal vi jobbe med følgjande skjema for ein liten ordredatabase. Primærnøklar er understreka.

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 ein int, og dei andre attributta strenger, høvesvis for å lagre namn og skildring.

I tabellen *customers* vert informasjon om kundar lagra. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, medan time_registered har typen timestamp. time_registered er eit tidsstempel for når ein kunde først vart registrert i databasen. Alle kundar har ein unik kombinasjon av pnr og fdate.

I tabellen *orders* vert data om enkeltordrar lagra. id er ein int, customer_id er framandnøkkel til customers, shipping_addr ein streng, time_entered ein timestamp som seier når ein ordre kom inn, og status er ein streng for å lagre statusen til ordren (til dømes «shipped» eller «delayed»).

I tabellen *order_items* vert data om enkeltpostar på ein ordre lagra. order_id er framandnøkkel til orders, product_id er ein framandnøkkel til products, medan quantity og unit_price er av typen int, og seier kor mange av dette produktet vart bestilt og prisen for ei eining av produktet i denne ordren.

Oppgåve: Skriv ei spørring som finn alle kundar som har minst to ordrar med ulik shipping_address. Skriv ut kunde-id og kundenamn.

Skriv svaret ditt her...

Maks poeng: 5

9 SQL, produkter bestilt oftest (10%)

I denne delen av eksamen skal vi jobbe med følgjande skjema for ein liten ordredatabase. Primærnøklar er understreka.

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 ein int, og dei andre attributta strenger, høvesvis for å lagre namn og skildring.

I tabellen *customers* vert informasjon om kundar lagra. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, medan time_registered har typen timestamp. time_registered er eit tidsstempel for når ein kunde først vart registrert i databasen. Alle kundar har ein unik kombinasjon av pnr og fdate.

I tabellen *orders* vert data om enkeltordrar lagra. id er ein int, customer_id er framandnøkkel til customers, shipping_addr ein streng, time_entered ein timestamp som seier når ein ordre kom inn, og status er ein streng for å lagre statusen til ordren (til dømes «shipped» eller «delayed»).

I tabellen *order_items* vert data om enkeltpostar på ein ordre lagra. order_id er framandnøkkel til orders, product_id er ein framandnøkkel til products, medan quantity og unit_price er av typen int, og seier kor mange av dette produktet vart bestilt og prisen for ei eining av produktet i denne ordren.

Oppgåve: Skriv ein spørring som returnerer alle kundar som har bestilt dei produkta som har vore bestilt oftast (altså dei produkta som finst på flest ulike ordrar). Skriv ut id til kundane.

Hugs å ta omsyn til at fleire produkt kan ha vore bestilt oftast. T.d. om produkt A og produkt B begge har vore bestilt 500 gonger og dermed finst på flest ordrar, skal alle kundar som har bestilt anten produkt A eller produkt B skrivast ut.

Skriv svaret ditt her...

Maks poeng: 10

10 SQL, total kostnad (10%)

I denne delen av eksamen skal vi jobbe med følgjande skjema for ein liten ordredatabase. Primærnøklar er understreka.

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 ein int, og dei andre attributta strenger, høvesvis for å lagre namn og skildring.

I tabellen *customers* vert informasjon om kundar lagra. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, medan time_registered har typen timestamp. time_registered er eit tidsstempel for når ein kunde først vart registrert i databasen. Alle kundar har ein unik kombinasjon av pnr og fdate.

I tabellen *orders* vert data om enkeltordrar lagra. id er ein int, customer_id er framandnøkkel til customers, shipping_addr ein streng, time_entered ein timestamp som seier når ein ordre kom inn, og status er ein streng for å lagre statusen til ordren (til dømes «shipped» eller «delayed»).

I tabellen *order_items* vert data om enkeltpostar på ein ordre lagra. order_id er framandnøkkel til orders, product_id er ein framandnøkkel til products, medan quantity og unit_price er av typen int, og seier kor mange av dette produktet vart bestilt og prisen for ei eining av produktet i denne ordren.

Oppgåve: Skriv ei spørring som finn antal einingar og total kostnad for alle produkt selt til kundar i USA, kor minst 10 einingar har vorte seld. Skriv ut id og namn til produkta, samt antal einingar og total kostnad.

Eit eksempel finst under tekstboksen.

Skriv svaret ditt her...

Eksempel: om vi har følgjande data i *order_lines* for kundar 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 sjå slik ut (kolonnenamna vel du sjølv):

product_id	name	antal	kostnad
1	*namn her*	10	1100
2	*namn her*	15	3000

Maks poeng: 10

11 Relasjonsteori (5%)

I denne oppgåva ser vi berre på tabellen *customers*. Beskrivinga av tabellen er lik som i dei andre oppgåvene. Primærnøkkelen er understreka.

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

I tabellen *customers* vert informasjon om kundar lagra. id og pnr (personnummer) er av typen int, fdate (fødselsdato) har datatypen date, name, address, og country er strenger, medan time_registered har typen timestamp. time_registered er eit tidsstempel for når ein kunde først vart registrert i databasen. Alle kundar har ein unik kombinasjon av pnr og fdate.

Oppgåve:

a. Kva for attributt i tabellen *customers* er kandidatnøklar?
b. Kva normalform er tabellen *customers* på? Grunngje svaret ditt.

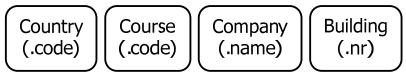
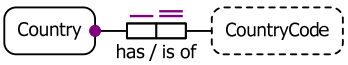
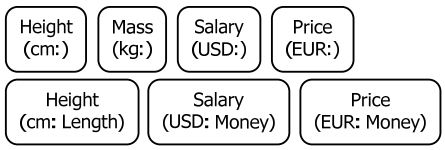
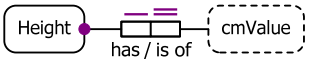
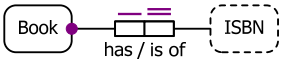
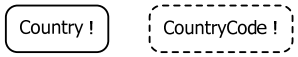
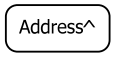
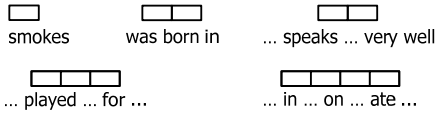
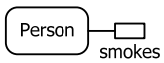
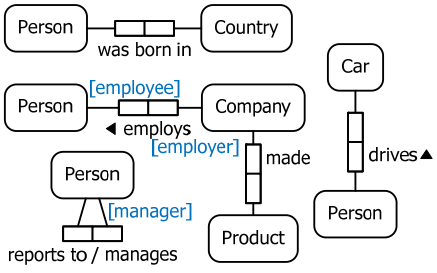
Skriv svaret ditt her...

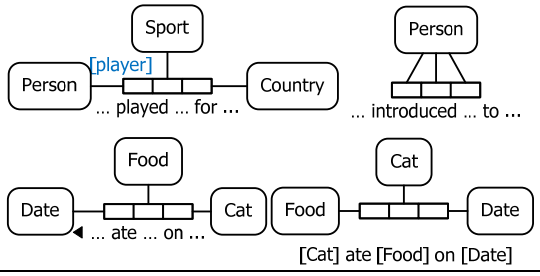
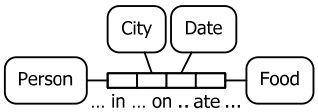
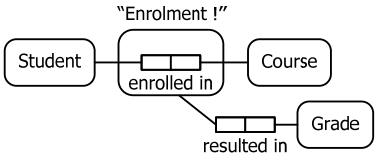
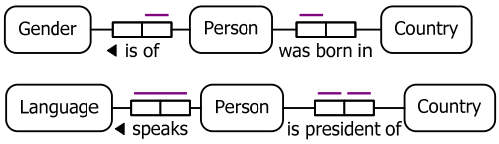
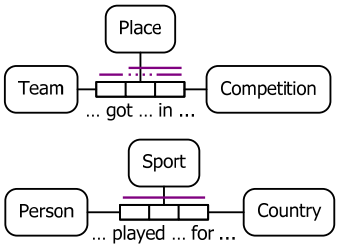
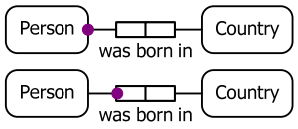
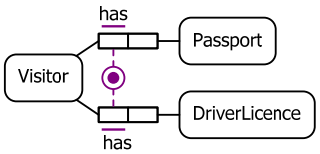
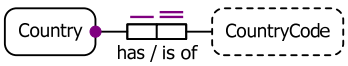
Maks poeng: 5

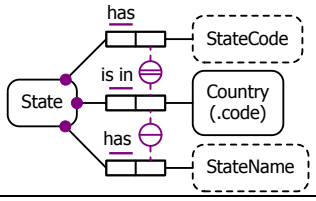
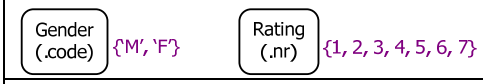
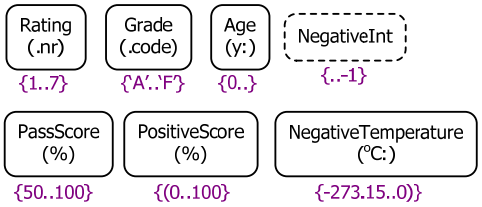
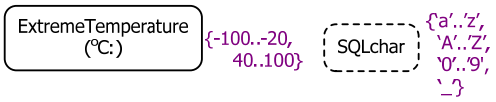
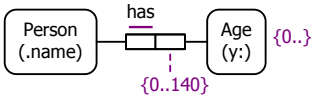
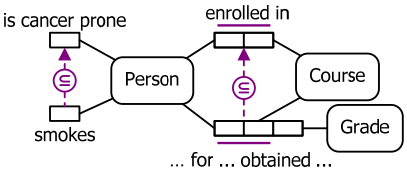
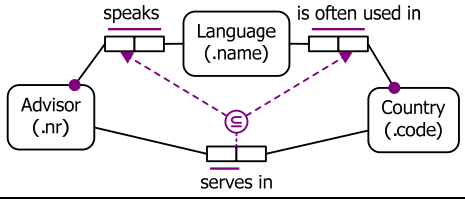
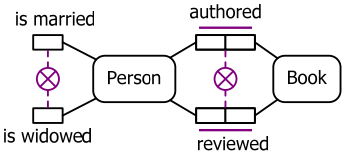
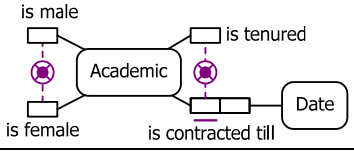
Question 2
Attached

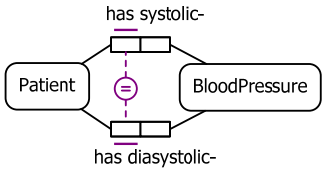
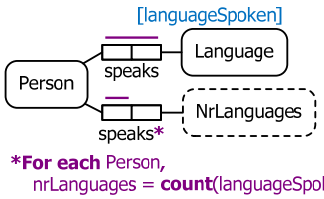
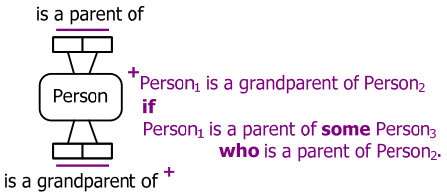
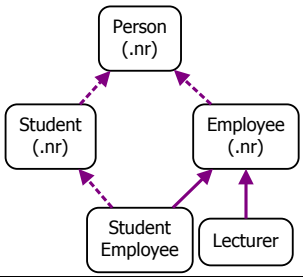
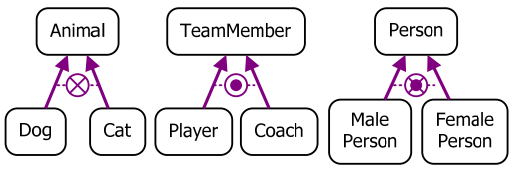
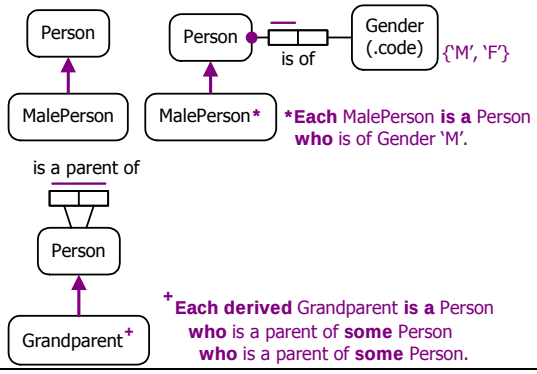
ORM 2 Graphical Notation

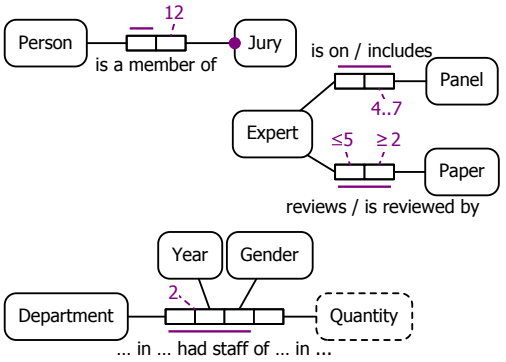
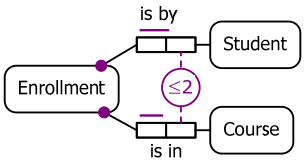
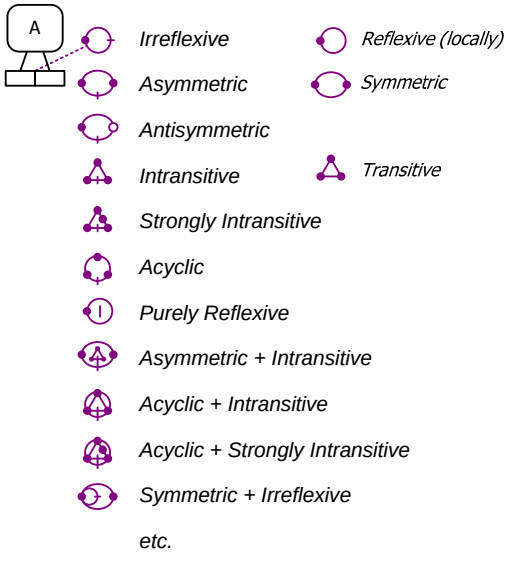
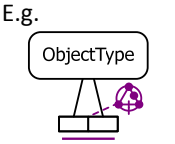
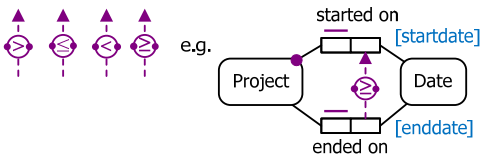
Terry Halpin

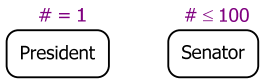
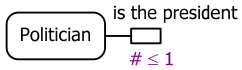
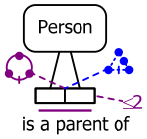
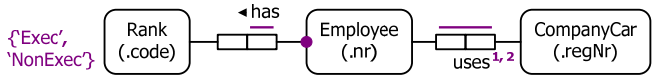
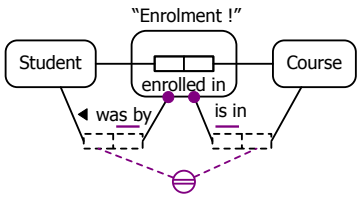
Construct	*Examples*	*Description/Notes*
Entity Type		Named soft rectangle, named hard rectangle, or named ellipse. The soft rectangle shape is the default.
Value Type		Named, dashed, soft rectangle (or hard rectangle or ellipse).
Entity type with popular reference mode		Abbreviation for injective reference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