Tentamen Gegevensbanken (211074) — 1 november 2007

NAAM, VOORLETTERS: _____

STUDENTNUMMER: _____

OPLEIDING: _____

De uitwerkingen moeten op deze opgavenformulieren worden genoteerd in de daarvoor bestemde vakken. Alle overige ruimte en de achterkanten van de bladen kun je zo nodig als kladpapier gebruiken en wordt niet bekeken en niet beoordeeld. Gebruik van ander kladpapier is toegestaan. Het gebruik van boeken, dictaten en dergelijke is niet toegestaan, behoudens één vel van A4 formaat met *eigen* aantekeningen (dubbelzijdig, getypt of geschreven) of kopieen van *delen van het boek*; kopieen van ander materiaal (zoals tentamenuitwerkingen) zijn niet toegestaan.

Normering: per opgave staan de te behalen punten in de kantlijn; in totaal 100 punten. Het eindcijfer is het aantal behaalde punten gedeeld door 10. Onleesbare tekst wordt steeds fout gerekend.

Na afloop moet de *volledige* set opgavenformulieren worden ingeleverd; het kladpapier niet. De tentamenopgaven zijn niet geheim en worden voorzien van modeluitwerkingen op Teletop gepubliceerd. (Die modeluitwerkingen moet je op papier of elektronisch bij je hebben wanneer je je tentamen komt inzien.)

5p. **Opgave 1.** Zorg ervoor dat van het ingevulde tentamenbriefje **BEIDE DOORSLAGEN GOED LEESBAAR** zijn. Vul ook de velden *vaknaam*, *vakcode*, en *datum* in.

Niet invullen

1	2	3	4	5	6	7	8	9	10	11	12

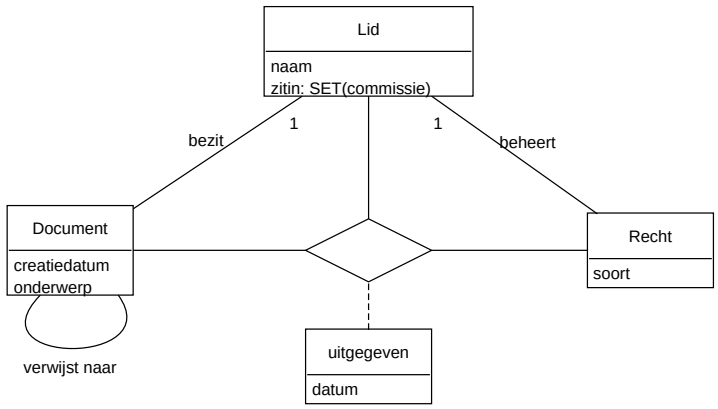
10p. **Opgave 2.** De sportvereniging SSS (Sport Schaadt Studie) wil een uitgebreide website hebben voor informatievoorziening (middels documenten). Er is een rechtenstructuur die aangeeft wie wat mag doen met welk document. Hier is een specificatie van de gewenste situatie:

- Van ieder lid is de naam bekend, en de commissies waarin dat lid zitting heeft. Van ieder document is onder andere de creatiedatum en het onderwerp bekend.
- Ieder document heeft precies één lid als eigenaar.
- Er zijn diverse rechten; voorbeelden daarvan zijn: het recht om te schrijven, te lezen, te plaatsen, te verwijderen, et cetera. Niet alle rechten hoeven uitgedeeld te zijn.
- Per lid, document en recht staat vast of dat recht verleend is aan dat lid betreffende dat document, en tot wanneer.
- Per recht is er een lid dat als beheerder optreedt voor dat recht (dat lid mag die rechten uitdelen, terugnemen, et cetera).
- Documenten kunnen naar elkaar verwijzen. Kennis hierover is belangrijk omdat een verwijzing vanuit een document naar een ander mogelijk gevolgen heeft voor de rechten met betrekking tot die documenten.
- Een lid dat eigenaar is van een document, heeft zeker een recht met betrekking tot dat document.

Geef in het antwoordblok een Entity-Relationship diagram dat bovenstaande situatie *zo precies mogelijk* modelleert en gebruik daarbij *zo geschikt mogelijke* constructies. Zowel de ERD-notatie uit het boek als ook de UML notatie (class diagram) is toegestaan, maar een mengeling van beide niet.

Onleesbare tekst wordt fout gerekend.

(Niet-vermelde multipliciteiten leggen geen beperkingen op: $0 \ldots *$.)



(Zie Toelichting.)

Het zou kunnen zijn (maar het is niet de bedoeling) dat uw diagram onbedoeld of met opzet *minder* eigenschappen modelleert dan de casus aangeeft. Om dat gebrek enigszins te herstellen is er deze vraag: Geef, in precieze bewoordingen of formules, eigenschappen die *wel* uit de casustekst volgen maar *niet* in uw ER-diagram opgenomen zijn; geef er zoveel mogelijk maar hóógstens twee (en dus geen enkele indien zulke eigenschappen niet bestaan):

Onleesbare tekst wordt fout gerekend.

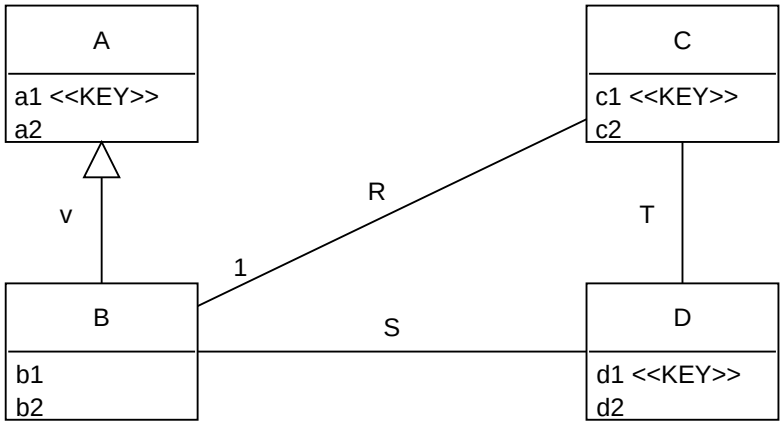
In het eerste ERD is de eigenschap van bullet 7 niet geformaliseerd (Een lid dat eigenaar is van een document, heeft zeker een recht met betrekking tot dat document).

Het zou kunnen zijn (maar het is niet de bedoeling) dat uw diagram ten onrechte of met opzet *méér* eigenschappen modelleert dan de casus aangeeft. Om zo'n "fout" enigszins te herstellen is er deze vraag: Noem en motiveer eigenschappen die *wel* in het diagram staan maar *niet* uit de letterlijke tekst van de casus volgen; geef er zoveel mogelijk maar hóógstens twee (en dus geen enkele indien zulke eigenschappen niet bestaan):

Onleesbare tekst wordt fout gerekend.

Geen

10p. **Opgave 3.** Beschouw het volgende ERD in de notatie van de UML (niet-geschreven multipliciteiten staan voor $0 \,..\, *$):



Het ER-diagram kan op verschillende manieren vertaald worden naar een databaseschema dat geschikt is om informatie die past in het ER-diagram, op te slaan. U dient een manier te kiezen waarbij er *zo weinig mogelijk* relatieschema's in het databaseschema zijn, maar met de beperkingen dat er geen NULLs nodig zijn vanwege de vertaling, er geen redundantie geïntroduceerd wordt door de vertaling, en alle attributen atomaire waarden hebben.

Geef de relatieschema's in SQL syntaxis waarbij de tekst '*create table*' en iedere domeinindicatie weggelaten mag worden; een voorbeeld van de vorm van zo'n schema is:

$$X(x_1, \ldots, \text{ primary key } (x_i, x_j ...), \text{ foreign key } (x_m, x_n ...) \text{ references } Y(y_1, y_2, ...), \ \ldots)$$

Onleesbare tekst wordt fout gerekend.

$A(a1, a2, \text{ primary key } (a1))$
$B(a1, b1, b2, \text{ primary key } (a1), \text{ foreign key } (a1) \text{ references } A(a1))$
$C(c1, c2, a1, \text{ primary key } c1, \text{ foreign key } (a1) \text{ references } B(a1))$
$D(d1, d2, \text{primary key } d1)$
$S(a1, d1, \text{ primary key } (a1, d1),$
foreign key $(a1)$ references $B(a1)$,
foreign key $(d1)$ references $D(d1))$
$T(c1, d1, \text{ primary key } (c1, d1),$
foreign key $(c1)$ references $C(c1)$,
foreign key $(d1)$ references $D(d1))$ *(Zie Toelichting.)*

Zijn er eigenschappen die wel in het ERD staan maar niet in uw databaseschema, of omgekeerd? Zo ja, geef die eigenschappen met een maximum van drie:

Onleesbare tekst wordt fout gerekend.

Nee

Aan het ERD wordt nu de volgende eigenschap ("integrity constraint") toegevoegd:

Voor iedere $(b, d) \in S$ is er een c zodanig dat $(b, c) \in R$ en $(c, d) \in T$.

Geef aan hoe u deze eigenschap in het database schema verwerkt op zodanige manier dat in elke databasevulling die eigenschap zal gelden:

Onleesbare tekst wordt fout gerekend.

Neem de volgende check in het schema van S op:

exists (select $*$ from C c, T t where $a1 = c.a1$ and $c.c1 = t.c1$ and $t.d1 = d1$)

Of –minder efficiënt– voeg een assertion toe aan het database schema met de volgende check:

not exists (

select $*$ from S s where not exists (

select $*$ from C c, T t where $a1 = c.a1$ and $c.c1 = t.c1$ and $t.d1 = d1$))

Of –met enige redundantie– voeg een attribuut $c1$ toe aan S en ook de volgende checks:

check ((a1,c1) in (select $a1$, $c1$ from C))

check ((c1,d1) in (select $c1$, $d1$ from T))

De laatste check hierboven kan als een foreign key constraint geformuleerd worden:

foreign key $(c1, d1)$ references $T(c1, d1)$

10p. **Opgave 4.** Beschouw een muziekliefhebber die allerlei gegevens bijhoudt, zoals: liederen, schrijvers, muziekwinkels, opslagmedia, muziekcategorieen, etc. Deze gegevens tezamen vormen een relatie $\mathcal{R}$ die de entiteiten op de volgende manier aan elkaar relateert. Een tuple (L, S, W, P, C, U, M) zit op zeker tijdstip in $\mathcal{R}$ precies wanneer op dat tijdstip het volgende geldt:

1. L is een Lied.
2. S is de Schrijver van lied L.
3. W is een Winkel waar L verkocht wordt.
4. P is de Prijs die winkel W vraagt voor lied L op medium M.
5. C is een Categorie waarin lied L is ingedeeld.
6. U is de Uitgever van lied L.
7. M is een opslag$M$edium waarop lied L te koop is in winkel W.

Voorts gelden er de volgende eigenschappen op ieder tijdstip:

a. Een lied heeft precies één schrijver.
b. Een schrijver werkt voor hooguit één uitgever (dat wil zeggen, van die schrijver worden alle liederen door hooguit één uitgever uitgegeven).
c. Een uitgever is gespecialiseerd in precies één categorie (dat wil zeggen, die uitgever geeft geen liederen uit die niet tot zijn specialisatiecategorie behoren).

Geef voor ieder van de functionele afhankelijkheden hieronder, met een letter W of O aan of die altijd Waar of Onwaar is in relatie $\mathcal{R}$, en motiveer kort uw keuze (de motivatie telt mee in de beoordeling):

Onleesbare tekst wordt fout gerekend.

FD	W/O	Motivatie voor uw keuze
$W, S \to U$	W	(b)
$C \to U$	O	twee uitgevers kunnen in dezelfde categorie gespecialiseerd zijn
$P, M, W \to L$	O	twee liederen kunnen te koop zijn bij dezelfde P, M en W
$P, M, L \to W$	O	twee winkels kunnen hetzelfde lied verkopen met dezelfde P en M
$L, W \to M$	O	een L kan in W op verscheidene Media te koop zijn
$S \to C$	W	wegens $S \to U$ (b) en $U \to C$ (c)
$L \to C$	W	wegens $L \to S$ (2,a) en $S \to U$ (b) en $U \to C$ (c) ("wegens 5" is fout!)

Geef, zonder motivatie, zoveel mogelijk maar hooguit twee, altijd ware multivalued dependencies in $\mathcal{R}$ die niet triviaal of onvermijdelijk zijn en zelf geen functionele afhankelijkheid zijn:

Onleesbare tekst wordt fout gerekend.

$LSWPCUM = LSCU \bowtie LWPM$ (L is key in lhs) (maar dit is equivalent met $L \to SCU$).
$LSWPCUM = SCU \bowtie LSWPM$ (S is key in lhs) (maar dit is equivalent met $S \to CU$).

10p. **Opgave 5.** Beschouw het relatieschema $\mathbf{R} = (\bar{R}, \mathcal{F})$, waarbij de attribuutverzameling $\bar{R}$ en de verzameling $\mathcal{F}$ van functionele afhankelijkheden als volgt luiden:

$$\begin{aligned} \bar{R} &= ABCDEF \\ \mathcal{F} &= \{AB \to D, \quad BCD \to F, \quad EF \to C, \quad E \to B\} \end{aligned}$$

(1) Geef in het antwoordblok in iedere regel een zo groot mogelijk rechterlid Y zó dat de functionele afhankelijkheid $X \to Y$ volgt uit de hierboven gegeven verzameling $\mathcal{F}$ (met andere woorden: Y is de closure $X^+_{\mathcal{F}}$). U hoeft de leden van X niet op te nemen in Y.
(2) *Omcirkel* in het antwoordblok iedere *sleutel* van **R**.
(3) *Onderstreep* in het antwoordblok iedere *supersleutel* van **R**.
(4) Omcirkel in het antwoordblok de *nummers* van de functionele afhankelijkheden die een schending vormen van de BCNF-eis.

Onleesbare tekst wordt fout gerekend.

	X	$\rightarrow$	Y
10	AD	$\rightarrow$	
11	AE	$\rightarrow$	BD
12	AF	$\rightarrow$	
13	BC	$\rightarrow$	
14	BD	$\rightarrow$	
15	BE	$\rightarrow$	
16	BF	$\rightarrow$	
17	CD	$\rightarrow$	
18	CE	$\rightarrow$	B
19	CF	$\rightarrow$	
20	DE	$\rightarrow$	B
21	DF	$\rightarrow$	
22	EF	$\rightarrow$	BC
23	ABC	$\rightarrow$	DF
24	ABD	$\rightarrow$	
25	ABE	$\rightarrow$	D
26	ABF	$\rightarrow$	D
27	ACD	$\rightarrow$	
28	ACE	$\rightarrow$	BDF
29	ACF	$\rightarrow$	
30	ADE	$\rightarrow$	B
31	ADF	$\rightarrow$	
32	AEF	$\rightarrow$	BCD

(Zie Toelichting.)

10p. **Opgave 6.** Beschouw het relatieschema $\mathbf{R} = (ABCDEFGH, \mathcal{F})$, waarbij:

$$\mathcal{F} \;=\; \{AB{\rightarrow}CD, \quad BC{\rightarrow}E, \quad GH{\rightarrow}F, \quad F{\rightarrow}E\}$$

Construeer in het antwoordblok een lossless decompositie van $\mathbf{R}$ tot schema's die ieder in BCNF staan. *Geef ook aan of de functionele afhankelijkheden behouden blijven onder de decompositie.* Geef bij iedere stap een verklaring zodat het voor de corrector heel duidelijk is hoe u te werk gaat.

Let op: $AB \rightarrow E$ volgt uit $\mathcal{F}$, en zal dus (bij een correcte redenering) een FD worden van een component met attributen ABE... (ook al zit C daar niet bij). Net zo: $GH \rightarrow E$ volgt uit $\mathcal{F}$ en zal dus een FD worden van een component met attributen GHE... (ook al zit F daar niet bij).
We passen het BCNF-algoritme toe.

- We bekijken $\mathbf{R} = (ABCDEFGH, \mathcal{F})$.

Van de gegeven $\mathcal{F}$ zijn alle leden een schending van de BCNF-eis voor $\mathbf{R}$; bijvoorbeeld voor $AB \rightarrow CD$: de FD is niet-triviaal en het linkerlid is geen sleutel: $AB^+_{\mathcal{F}} = ABCDE \neq ABCDEFGH$. We kiezen (zomaar) de eerste, $AB \rightarrow CD$, ter eliminatie. Dus splitsen we $\bar{R}$ in $\bar{R}_1 = ABCD$ en $\bar{R}_2 = AB \not\subset\!\!\!\not\supset EFGH = ABEFGH$. Dit levert schema's $\mathbf{R}_i = (\bar{R}_i, \mathcal{F}_i)$, waarbij $\mathcal{F}_i$ (een basis voor) de inperking is van $\mathcal{F}^+$ tot $\bar{R}_i$. Dus, $\mathbf{R}_1 = (ABCD, \{AB \rightarrow CD\})$ en $\mathbf{R}_2 = (ABEFGH, \{AB \rightarrow E,\ GH \rightarrow F,\ F \rightarrow E\})$. Let op: de $AB \rightarrow E$ zit weliswaar niet in $\mathcal{F}$, maar toch in $\mathcal{F}_2$ zoals aan het begin is uitgelegd. De afhankelijkheid $BC \rightarrow E$ is verloren gegaan.

- We bekijken nu $\mathbf{R}_1 = (ABCD, \{AB \rightarrow CD\})$.

In $\mathbf{R}_1$ is $AB \rightarrow CD$ geen schending van de BCNF-conditie, en dus staat $\mathbf{R}_1$ in BCNF.

- We bekijken nu $\mathbf{R}_2 = (ABEFGH, \{AB \rightarrow E,\ GH \rightarrow F,\ F \rightarrow E\})$.

In $\mathbf{R}_2$ zijn $AB \rightarrow E$ en $GH \rightarrow F$ en $F \rightarrow E$ alledrie een schending van de BCNF-conditie; voor $GH \rightarrow F$ is de reden dat die niet-triviaal is en GH geen sleutel is in $\mathbf{R}_2$ (want $GH^+_{\mathcal{F}_2} = EFGH \neq ABEFGH$). We kiezen (zomaar) $GH \rightarrow F$ ter eliminatie. Dus splitsen we $\bar{R}_2$ in $\bar{R}_{2a} = GHF$ en $\bar{R}_{2b} = ABE \not\subset\!\!\!\not\supset GH = ABEGH$. Dit levert schema's $\mathbf{R}_{2j} = (\bar{R}_{2j}, \mathcal{F}_{2j})$, waarbij $\mathcal{F}_{2j}$ (een basis voor) de inperking is van $\mathcal{F}^+$ (of $\mathcal{F}_2^+$) tot $\bar{R}_{2j}$. Dus $\mathbf{R}_{2a} = (GHF, \{GH \rightarrow F\})$ en $\mathbf{R}_{2b} = (ABEGH, \{AB \rightarrow E, GH \rightarrow E\})$. Let op: de $GH \rightarrow E$ zit weliswaar niet in $\mathcal{F}_2$, maar toch in $\mathcal{F}_{2b}$ zoals aan het begin is uitgelegd.

- We bekijken nu $\mathbf{R}_{2a} = (FGH, \{GH \rightarrow F\})$. Deze staat in BCNF.

- We bekijken nu $\mathbf{R}_{2b} = (ABEGH, \{AB \rightarrow E,\ GH \rightarrow E\})$. Zowel $AB \rightarrow E$ als ook $GH \rightarrow E$ vormen een schending van de BCNF-conditie in $\mathbf{R}_{2b}$; bijvoorbeeld voor $AB \rightarrow E$: die FD is niet triviaal en het linkerlid is geen sleutel (want $AB^+_{\mathcal{F}_{2b}} = ABE \neq ABEGH$). We kiezen (zomaar) $AB \rightarrow E$ ter eliminatie. Dus splitsen we $\bar{R}_{2b}$ in $\bar{R}_{2b1} = ABE$ en $\bar{R}_{2b2} = AB \not\subset\!\!\!\not\supset GH = ABGH$. Dit levert schema's $\mathbf{R}_{2bx} = (\bar{R}_{2bx}, \mathcal{F}_{2bx})$, waarbij $\mathcal{F}_{2bx}$ (een basis voor) de inperking is van $\mathcal{F}^+$ (of $\mathcal{F}_{2b}^+$) tot $\bar{R}_{2bx}$. Dus $\mathbf{R}_{2b1} = (ABE, \{AB \rightarrow E\})$ en $\mathbf{R}_{2b2} = (ABGH, \{\,\})$. Beide staan in BCNF.

- Dus $\{\mathbf{R}_1, \mathbf{R}_{2a}, \mathbf{R}_{2b1}, \mathbf{R}_{2b2}\}$ is een decompositie van $\mathbf{R}$ waarvan alle componenten in BCNF staan. De functionele afhankelijkheden zijn niet allemaal behouden; met name is $BC \rightarrow E$ verloren gegaan.

- NB. Omdat dit een lossless decompositie is (een eigenschap van het toegepaste BCNF-algoritme), schrijven we ook wel:

  "$ABCDEFGH = ABCD \bowtie FGH \bowtie ABE \bowtie ABGH$ geldt in $\mathbf{R}$."

Wanneer je met een andere schending begint kun je mogelijk een andere decompositie bereiken.

In de volgende opgavenserie wordt het volgende databaseschema gebruikt:

Class (name, type, country, guns, bore, displacement)
Ship (name, classname, launched)
Battle (name, date)
Outcome (shipname, battlename, result)

De attributen die tot de sleutel behoren zijn onderstreept.
In *Ship* is *classname* een foreign key verwijzend naar *Class (name)*.
In *Outcome* is *shipname* een foreign key verwijzend naar *Ship (name)*.
In *Outcome* is *battlename* een foreign key verwijzend naar *Battle (name)*.

Schepen die volgens eenzelfde ontwerp worden gebouwd vormen samen een klasse (*class*). Klassen komen in twee typen (*type*): *bb* (voor *battleship*) en *bc* (voor *battlecruiser*). De overige attributen van een klasse zijn: het land (*country*), het aantal kanonnen (*guns*), de diameter in centimeters van de kanonsloop (*bore*), en de waterverplaatsing (*displacement*, gemeten in tonnen). Van een schip is, naast de naam (*name*) en de klassenaam (*classname*), ook nog bekend wanneer het te water is gelaten (*launched*). Van een zeeslag (*battle*) is de naam (*name*) en datum (*date*) bekend. Relatie *Outcome* geeft aan hoe schepen de zeeslagen hebben doorstaan: gezonken, beschadigd of okay (*result* = *sunk*, *damaged*, en *ok*, respectievelijk).

Wanneer we spreken van het *type* van een schip, dan bedoelen we het *type* van de klasse van dat schip; net zo voor de attributen *country*, *guns*, *bore*, *displacement*. Dus alle schepen van een klasse komen uit één land: het land dat in de klasse genoemd staat.

U mag identifiers tot hun eerste letter afkorten. Het schema luidt dan:

C (n, t, c, g, b, d)
S (n, c, l)
B (n, d)
O (s, b, r)

10p. **Opgave 7.** Beschouw de volgende zoekopdracht:

Geef de landen waarvan een schip voor het jaar 1600 te water is gelaten.

Veronderstel dat datums (zoals het *launched* attribuut van een schip) vergeleken kunnen worden met $<$ ('kleiner dan'), etc. Geef voor deze vraag een *afleiding* in Verzamelingsnotatie naar een vorm die dicht aansluit bij SQL met zo weinig mogelijk subqueries en geen group by clausule. Het begin is al gegeven. Kort tabel- en attribuutnamen af tot hun eerste letter.

Onleesbare tekst wordt fout gerekend.

"(geef) ieder land waarvan een schip voor 1600 te water ging"
$= \{cntry{:}txt \mid$ "$cntry$ is een land" $\wedge$ "een schip uit $cntry$ ging voor 1600 te water" $\bullet\ cntry\}$
$= \{cntry{:}txt \mid (\exists\, c{:}C \bullet c.c{=}cntry) \wedge (\exists\, s{:}S \mid$ "s is uit $cntry$"$\bullet$"s voor 1600 te water"$)\ \bullet\ cntry\}$
$= \{cntry{:}txt;\ c{:}C \mid c.c{=}cntry \quad \wedge (\exists\, s{:}S \mid$ "s is uit $cntry$"$\bullet$"s voor 1600 te water"$)\ \bullet\ cntry\}$
$= \{cntry{:}txt;\ c{:}C \mid c.c = cntry \quad \wedge (\exists\, s{:}S \mid$ "s is uit $c.c$"$\bullet$"s voor 1600 te water"$)\ \bullet\ c.c\}$
$= \{c{:}C \mid (\exists\, cntry{:}txt \bullet c.c = cntry) \wedge (\exists\, s{:}S \mid$ "s is uit $c.c$"$\bullet$"s voor 1600 te water"$)\ \bullet\ c.c\}$
$= \{c{:}C \mid c.c \in txt \quad \wedge (\exists\, s{:}S \mid$ "s is uit $c.c$"$\bullet$"s voor 1600 te water"$)\ \bullet\ c.c\}$
$= \{c{:}C \mid \quad (\exists\, s{:}S \mid$ "s is uit $c.c$" $\bullet$ "s voor 1600 te water"$)\ \bullet\ c.c\}$
$= \{c{:}C;\ s{:}S \mid$ "s is uit $c.c$" $\wedge$ "s voor 1600 te water" $\bullet\ c.c\}$
$= \{c{:}C;\ s{:}S \mid (\exists\, c'{:}C \mid s.c = c'.n \bullet c'.c = c.c) \wedge s.l < 1600 \bullet c.c\}$
$= \{c, c' : C;\ s : S \mid s.c = c'.n \wedge c'.c = c.c \wedge s.l < 1600 \bullet c.c\}$
$= \{c' : C;\ s : S \mid s.c = c'.n \wedge (\exists\, c : C \bullet c'.c = c.c) \wedge s.l < 1600 \bullet c'.c\}$
$= \{c' : C;\ s : S \mid s.c = c'.n \wedge s.l < 1600 \bullet c'.c\}$

Geef een SQL formulering van de beschouwde vraag; de SQL formulering moet dicht aansluiten bij de zojuist gegeven uitdrukking (en dus zo weinig mogelijk subqueries hebben). Gebruik *DISTINCT* alleen wanneer het nodig is.

Onleesbare tekst wordt fout gerekend.

select distinct $c1.c$ from C $c1$, S s where $s.c = c1.n$ and $s.l < 1600$

Geef ook een formulering van de vraag in Relationele Algebra op zodanige manier dat het Cartesisch product $\times$ daarin niet voorkomt:

Onleesbare tekst wordt fout gerekend.

$\pi_{country}\ (\sigma_{launched<1600}\ (C \bowtie_{name=classname} S))$

$=\ \pi_{c}\ (\sigma_{sl<1600}\ (C' \bowtie S'))$

Hierbij zijn met renaming nieuwe relaties gedefinieerd:

$S' = S[\ldots, cn, sl]$ en $C' = C[cn, c, \ldots]$.

Class (<u>name</u>, type, country, guns, bore, displacement)	*C (<u>n</u>, t, c, g, b, d)*
Ship (<u>name</u>, classname, launched)	*S (<u>n</u>, c, l)*
Battle (<u>name</u>, date)	*B (<u>n</u>, d)*
Outcome (<u>shipname, battlename</u>, result)	*O (<u>s, b</u>, r)*

10p. **Opgave 8.** Formuleer in SQL met een group-by query:

Geef van iedere klasse waarvan alle schepen tussen 1700 en 1800 te water zijn gelaten, het aantal schepen dat in een of andere zeeslag gezonken is, mits dat aantal minstens drie is.

Veronderstel dat datums (zoals het *launched* attribuut van een schip) vergeleken kunnen worden met $<$ ('kleiner dan'), etc, en dus ook *min* en *max* daarop van toepassing zijn.

Onleesbare tekst wordt fout gerekend.

```
select s.classname, count (distinct s.name)
from ship s, outcome o
where s.name = o.shipname and o.result = sunk and s.classname not in
                (select s.classname from ship s where not (s.launched between 1700 and 1800))
group by s.classname
having count (distinct s.name)>2
(Wanneer een schip maar éénmaal kan zinken, is 'distinct' overbodig, en kan zelfs 'count
(distinct s.name)' vervangen worden door 'count (*)'.)                    (Zie Toelichting.)
```

5p. **Opgave 9.** (Goede beantwoording levert 5 bonuspunten boven op de 5 punten die voor deze opgave gegeven worden. Daardoor kan het totaal aantal behaalde punten op 105 uitkomen.) Formuleer *in Verzamelingsnotatie* ***én*** *in SQL*:

> Geef iedere zeeslag waarvoor geldt dat voor iedere klasse van aan die zeelag deelnemende schepen, er een *andere* klasse is met een gelijk aantal kanonnen.

(U hoeft geen afleiding te geven, maar een afleiding, hieronder of op kladpapier, zou u wel kunnen helpen.)

Onleesbare tekst wordt fout gerekend.

Kortheidshalve laten we de typering achterwege (en korten dus $b : B$ af tot b, et cetera).

$\{b \mid (\forall\, c \mid$ "c neemt deel aan b" $\bullet$ "er is een *andere* klasse met gelijk kanonnental"$) \bullet b.n\}$

$= \{b \mid (\forall\, c \mid (\exists\, s \mid$ "s behoort tot c" $\bullet$ "s neemt deel aan b"$) \bullet (\exists\, c' \mid c' \neq c \bullet c'.g = c.g)) \bullet b.n\}$

$= \{b \mid (\forall\, c \mid (\exists\, s \mid s.c = c.n \bullet (\exists\, o \bullet s.n = o.s \wedge o.b = b.n)) \bullet (\exists\, c' \mid c' \neq c \bullet c'.g = c.g)) \bullet b.n\}$

$= \{b \mid (\forall\, c \mid (\exists\, s;\, o \bullet s.c = c.n \wedge s.n = o.s \wedge o.b = b.n) \bullet (\exists\, c' \mid c' \neq c \bullet c'.g = c.g)) \bullet b.n\}$

$= \{b \mid (\forall\, c;\, s;\, o \mid s.c = c.n \wedge s.n = o.s \wedge o.b = b.n \bullet (\exists\, c' \mid c' \neq c \bullet c'.g = c.g)) \bullet b.n\}$

$= \{b \mid \neg\, (\exists\, c;\, s;\, o \mid s.c = c.n \wedge s.n = o.s \wedge o.b = b.n \bullet \neg\, (\exists\, c' \mid c' \neq c \bullet c'.g = c.g)) \bullet b.n\}$

= select $b.n$ from B b where not exists (

  select $*$ from C c, S s, O o where

    $s.c = c.n$ and $s.n = o.s$ and $o.b = b.n$ and

    not exists (select $*$ from C c' where $c'.n \neq c.n$ and $c'.g = c.g$))

10p. **Opgave 10.** Beschouw het volgende databaseschema:

Person (id, name)
Student(id, university)

De bedoeling is dat iedere student ook een persoon is. Daarom willen we dat voortdurend de volgende acties worden uitgevoerd:

1. Wanneer een rij wordt verwijderd uit tabel *Person*, dan wordt de rij met hetzelfde *id* ook verwijderd uit tabel *Student*.

2. Wanneer er gepoogd wordt een rij toe te voegen aan tabel *Student*, dan wordt getest of er een rij met hetzelfde *id* bestaat in *Person*; is dat niet het geval, dan vindt de toevoeging niet plaats.

Voeg key constraints toe aan het schema waardoor de gewenste acties automatisch door het databasesysteem worden uitgevoerd. Ter herinnering, key constraints zien er als volgt uit:

- primary key $(x, x', \ldots)$
- foreign key $(x, x', ...)$ references $T(y, y', ...)$ on *event action* on *event′ action′* ...

Hierbij staat x voor een attribuutnaam, T voor een tabelnaam, *event* voor een gebeurtenis (*delete*, *update*) en *action* voor een actie (*set null*, *set default*, *no action*, *cascade*).

Onleesbare tekst wordt fout gerekend.

Person (*id*, *name*, primary key *id*)

Student (*id*, *name*, foreign key (*id*) references *Person*(*id*) on delete cascade)

Het deel 'on delete cascade' zorgt voor actie 1,

het deel 'foreign key(*id*) references *Person*(*id*)' zorgt voor actie 2.

Er werd niet gevraagd om in *Student* de *id* tot primary key te verklaren, en ook niet om de actie die opgeroepen wordt door 'on update cascade' toe te voegen aan de foreign key constraint.

Geef nu, voor het oorspronkelijk gegeven schema, een of meer triggers waardoor de gewenste acties automatisch door het databasesysteem worden uitgevoerd. Ter herinnering, de syntaxis van een trigger creatie luidt als volgt:

create trigger *trigger-name*
{before | after} {insert | delete | update [of *column-name-list*] } on *table-name*
[referencing [old as *var-to-refer-to-old-tuple*]
[new as *var-to-refer-to-new-tuple*]
[old table as *name-to-refer-to-old-table*]
[new table as *name-to-refer-to-new-table*]]
[for each { row | statement }]
[when (*precondition*)]
statement-list

Hierbij staat $\{x \mid \ldots\}$ voor een keuze uit $x, \ldots$; en $[x]$ staat voor een keuze uit x of niets.

Onleesbare tekst wordt fout gerekend.

create trigger *OnPersonDelete*

after delete on *Person*

referencing old as *DeletedPerson*

for each row

delete from *Student s* where *s.id* = *DeletedPerson.id*;

create trigger *OnStudentInsert*

before insert on *Student*

referencing new as *NewStudent*

for each row

when (*NewStudent.id* not in (select *id* from *Person*))

abort;

Varianten met statement level granularity ('for each statement', de default) zijn ook mogelijk.

(Zie Toelichting.)

10p. **Opgave 11.** Zet in onderstaande tabel een merkteken '×' bij iedere combinatie van isolatieniveaus voor transacties T_1 en T_2 waarbij T_2 geen phantom kan ervaren.

Onleesbare tekst wordt fout gerekend.

Op de met '×' gemerkte niveaus kan T_2 geen phantom ervaren:

Isolatieniveau voor T_1: ↓	Isolatieniveau voor T_2: read uncommitted	read committed	repeatable read	serializable
read uncommitted				×
read committed				×
repeatable read				×
serializable				×

Het niveau voor T_1 doet niet terzake!

Leg uit wat er wordt verstaan onder 'transactie T ervaart geen phantoms':

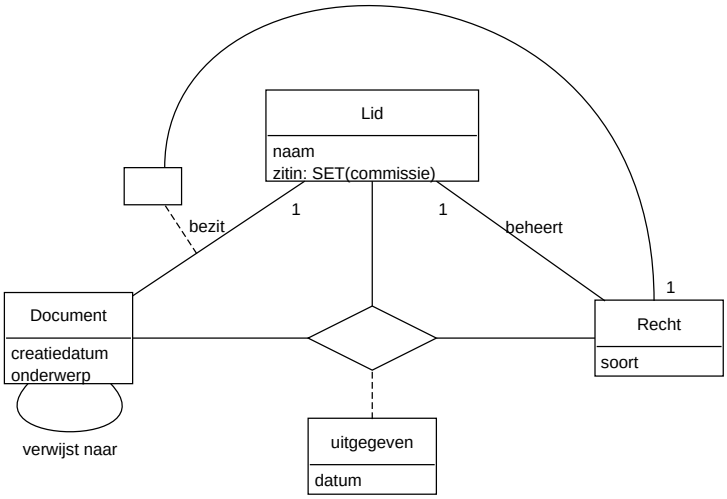
Onleesbare tekst wordt fout gerekend.

'Transactie T ervaart geen phantoms' precies wanneer het volgende *niet* gebeurt:

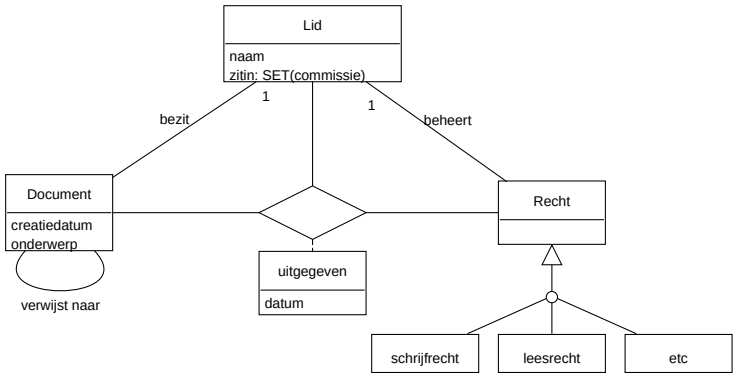
T selecteert rijen van een tabel met een selectiecriterium P en gedurende de executie van T wordt die tabel door een andere transactie uitgebreid met een rij die aan selectiecriterium P voldoet — waarna T opnieuw rijen van die tabel volgens hetzelfde criterium P selecteert (en dus "een nieuwe rij", de phantom, krijgt opgeleverd).

Toelichtingen

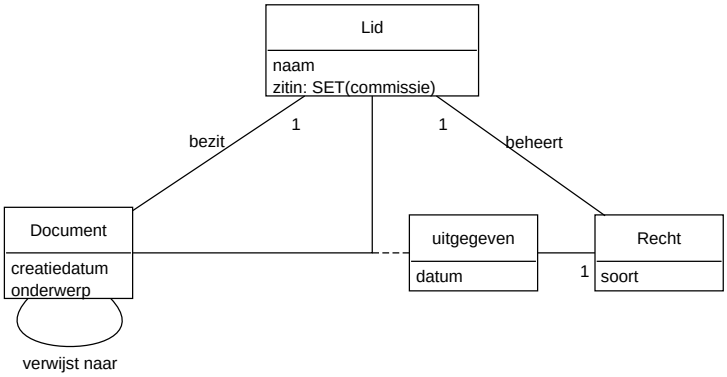
Toelichting bij antwoord 2.
Er is géén afzonderlijke entiteit *Commissie* (met een associatie naar *Lid*) omdat er volgens de casus van commissies geen informatie wordt bijgehouden.
Een variant waarin ook het laatste punt (bullet) uit de casus wordt vastgelegd:



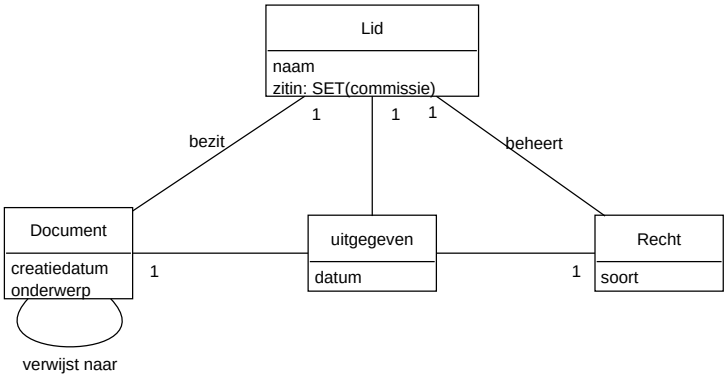
Een variant waarin de soorten rechten als afzonderlijke entiteiten worden geformaliseerd:



Een foute formalisering. In het volgende ERD is het niet zo dat "Per (*drietal*) lid, document en recht staat vast of dat recht verleend is aan dat lid betreffende dat document, en tot wanneer" maar in plaats daarvan wel dat "Per (*tweetal*) lid en document staat vast of aan dat lid betreffende dat document een recht verleend is en tot wanneer":



Nog een fout. Volgens het volgende diagram kan er meermalen "per lid, document en recht" een datum vastgesteld zijn:



Toelichting bij antwoord 3.
Entiteiten A en B zijn niet samengenomen in één tabel $A(a1, a2, b1, b2)$ omdat dan NULLs nodig zijn om een A-instantie te representeren die geen B-instantie is.

Entiteit R is niet als aparte tabel nodig, omdat R-instanties gerepresenteerd kunnen worden in een attribuut (namelijk het $a1$-attribuut) van C (vanwege de eigenschap dat er bij iedere C-instantie precies één B-instantie is waarmee die in relatie R zit).

Toelichting bij antwoord 5.
Let op: iedere key is ook een superkey, omgekeerd is dit niet het geval.

Toelichting bij antwoord 8.
Een alternatief met meer group by clausules:
select $s.classname$, count (distinct $s.name$)
from *ship* s, *outcome* o
where $s.name = o.shipname$ and $o.result = sunk$ and $s.classname$ in
(select $s'.classname$ from *ship* s'
group by $s'.classname$
having $1700 <$ min $(s'.launched)$ and max $(s'.launched) < 1800$)
group by $s.classname$
having count (distinct $s.name$) > 2

Toelichting bij antwoord 10.
Varianten met statement level granularity (de default):

create trigger *OnPersonDelete′*
after delete on *Person*
referencing old table as *DeletedPersons*
for each statement
delete from *Student s* where *s.id* in (select *id* from *DeletedPersons*);
create trigger *OnStudentInsert′*
before insert on *Student*
referencing new table as *NewStudents*
for each statement
when (exists ((select *id* from *NewStudents*) except (select *id* from *Person*)))
abort;