**Relationele gegevensbanken: Examen** Academiejaar 2021–2022
Prof. dr. Guy De Tré 2e zittijd: 23/08/2022

---

- Het examen is volledig open boek. Je mag al het materiaal gebruiken dat voorzien is tijdens de lessen (bv. het boek 'Principes van databases' van prof. dr. Guy De Tré, slides, opgaves, (opgeloste) oefeningen,...) of dat je zelf hebt gebruikt ter verwerking van de leerstof (bv. samenvattingen, notities,...).
- Alle communicatie met personen (behalve de examenbegeleiders) is strikt verboden en zal bestraft worden met uitsluiting van het examen.
- Het examen dient opgelost te worden op witte bladeren. Gebruik voor elk deel van het examen (sectie 1-4) een apart blad. Vermeld zeker op elk oplossingsblad je studentennummer, naam en richting.
- Indien er vragen of onduidelijkheden zijn, schrijf deze dan nauwgezet op je oplossingsblad, zodat wij hier tijdens het quoteren rekening mee kunnen houden.

# 1 Open vragen

Beantwoord onderstaande vragen.

1. (Databanksystemen) De meeste databanksystemen zijn opgebouwd volgens een drielagenarchitectuur. Toch gebeurt de bevraging van een relationele databank meestal rechtstreeks op de basisrelaties.
   a) Wat is de reden hiervoor?
   b) Geef twee goede argumenten waarom alle bevraging best via views zou gebeuren.
2. (Het relationeel databankmodel) Beschouw een basisrelatie met schema `Wedstrijddeelname({RennerID:varchar, Wedstrijd:varchar, Jaar:integer, Uitslag:varchar})`. Geef de algebraïsche expressie waarmee je de twee tuples
   - `(RennerID:10012, Wedstrijd:Parijs-Roubaix, Jaar:2022, Uitslag:3)`, en
   - `(RennerID:10032, Wedstrijd:Parijs-Roubaix, Jaar:2022, Uitslag:32)`

   kan verwijderen uit de basisrelatie. Indien je gebruik maakt van extra basisrelaties moet je deze volledig en correct meegeven.

3. (Herstel na falen) Beschouw de concurrente uitvoering van transacties $T_1$, $T_2$ en $T_3$ zoals weergegeven in Tabel 1. Initieel is salaris=1 en taks=2. Aanpassingen van salaris en taks gebeuren in lokale databankbuffers en zijn pas zichtbaar in de databank na flushing van de databankbuffers. Ga uit van een 'steal, no force' flushingstrategie. De gebruikte hersteltechniek is met *onmiddellijke aanpassing*.

| Tijd | $T_1$ | $T_2$ | $T_3$ |
|---|---|---|---|
| 1 | | | start |
| 2 | | | lees taks |
| 3 | | | taks := taks+1 |
| 4 | start | | |
| 5 | lees salaris | | |
| 6 | salaris := salaris+1 | | |
| 7 | | | schrijf taks |
| 8 | | | commit |
| 9 | | start | |
| 10 | | lees taks | |
| 11 | | lees salaris | |
| 12 | | salaris := salaris + taks | |
| 13 | | schrijf taks | |
| 14 | | commit | |
| 15 | | -------controlepunt------- | |
| 16 | lees taks | | |
| 17 | salaris := salaris + taks | | |
| 18 | schrijf salaris | | |
| 19 | commit | | |

Tabel 1: Concurrente uitvoering van transacties $T_1$, $T_2$ en $T_3$.

a) Veronderstel dat er zich een soft crash voordoet onmiddellijk na tijdstip 9, maar voor tijdstip 10 en dat alle logrecords tot en met dit tijdstip correct naar het logbestand zijn geschreven. Welke transacties moeten dan ongedaan gemaakt worden? Welke transacties moeten opnieuw uitgevoerd worden?

b) Veronderstel nu dat de soft crash zich voordoet onmiddellijk na tijdstip 18, maar voor tijdstip 19 en dat alle logrecords tot en met dit tijdstip correct naar het logbestand zijn geschreven. Welke transacties moeten nu ongedaan gemaakt worden? Welke transacties moeten opnieuw uitgevoerd worden?

4. (Delen van gegevens)

a) Kunnen ‘livelocks’ en ‘deadlocks’ samen voorkomen? Verklaar je antwoord.

b) Waarom worden bij de ‘wait-die’ en ‘wound-wait’ deadlock preventietechnieken steeds de jongere transacties afgebroken?

c) Wat is het voordeel om bij deze deadlock preventietechnieken te werken volgens een principe van veroudering?

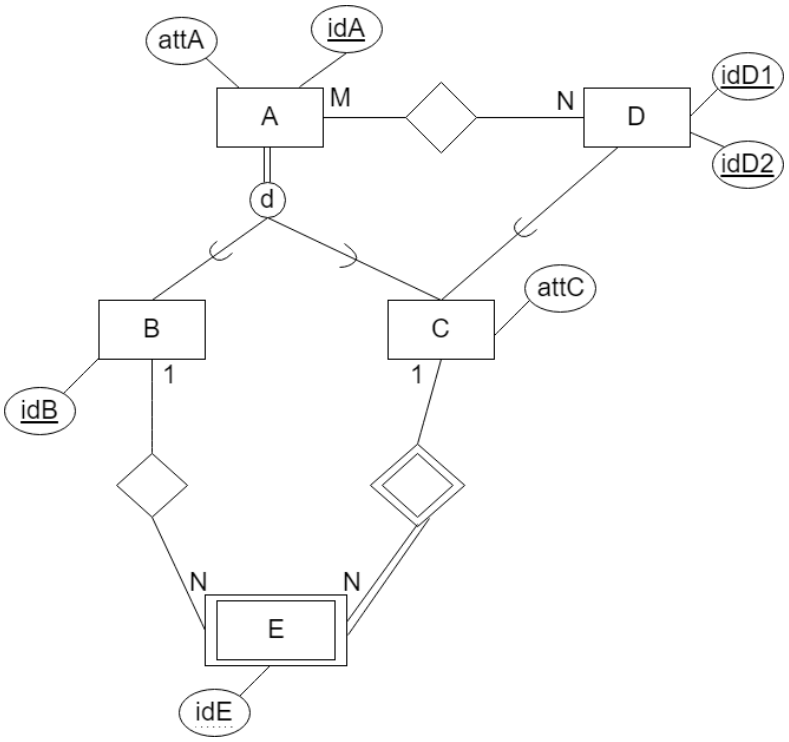
## 2 Conceptueel ontwerp

Om te vermijden dat je gedurende een lange, warme periode vergeet om de plantjes in jouw kamer te verzorgen, beslis je om een applicatie te ontwikkelen die je hiermee helpt door herinneringen te sturen. Een eerste stap in de ontwikkeling van deze applicatie is het opstellen van het conceptueel ontwerp van de onderliggende databank. Gelukkig voorzien wij tijdens dit examen tijd om dit conceptueel ontwerp uit te werken.

Om jouw planten te kunnen registreren dien je, in eerste instantie, een profiel aan te maken in de applicatie. Dit profiel wordt uniek geïdentificeerd door een gebruikersnaam. Daarnaast wordt er ook om een wachtwoord gevraagd (dat in hashvorm wordt opgeslagen) en kan je (optioneel) jouw woonplaats en e-mailadres meegeven. In tweede instantie moet er een overzicht zijn van alle mogelijke planten waaruit je jouw planten kan selecteren. Elke plant heeft een unieke naam, een beschrijving en behoort tot een bepaalde soort. Daarnaast wordt er (optioneel) een foto toegevoegd. Aan elke plant zijn er een of meerdere taken gelinkt in verband met de verzorging van de plant. Voor elke taak wordt er eerst en vooral bijgehouden om de hoeveel dagen de taak uitgevoerd moet worden. Daarnaast wordt er een onderscheid gemaakt tussen taken die gerelateerd zijn aan voedsel en taken die gerelateerd zijn aan water geven. Voor voedsel wordt de soort en hoeveelheid (in g) bijgehouden, voor water wordt enkel de hoeveelheid (in ml) bijgehouden. Ook wordt elke plant gelinkt aan minstens 1 gelijkaardige plant, zodat de applicatie eenvoudig nieuwe planten kan aanraden aan gebruikers.
Natuurlijk is het mogelijk om, als gebruiker, de planten die je in jouw bezit hebt te registreren. Aangezien het mogelijk is dat je eenzelfde plant meerdere keren bezit, kan je bij een registratie een bijnaam meegeven aan elke plant in jouw bezit. Echter is het wel zo dat elke gebruiker eenzelfde bijnaam slechts eenmalig kan kiezen (maar een bijnaam kan wel door meerdere gebruikers gekozen worden). Ook wordt er bij het bevestigen van een registratie de huidige datum opgeslagen.
Om extra (persoonlijke) informatie bij te houden, kan je een of meerdere notities schrijven en linken aan een registratie. Deze notities hebben een titel die uniek is per registratie, een categorie, een tijdstip en een beschrijving. Een notitie kan uiteraard enkel worden toegevoegd nadat de plant is geregistreerd.

## 3 Logisch ontwerp

Zet het (abstract) EER-diagram gegeven in Figuur 1 om naar een logisch databankschema. Lijst in dit logisch databankschema enkel de tabellen, attributen, primaire sleutels, vreemde sleutels en uniciteitsvoorwaarden op. Gebruik de notatie die we hebben gebruikt tijdens de oefeningenlessen.




Figuur 1: Abstract EER-diagram

## 4 Workshops

In Appendix 1 is de DDL-code gegeven voor de aanmaak van een databank waarin data in verband met de dagelijkse werking van een bibliotheek worden opgeslagen. Bekijk deze DDL-code nauwgezet en los daarna onderstaande vragen op.

1. De primaire sleutel van de tabel `schap` bestaat momenteel uit kolommen `kastnummer` en `schapnummer`. Leg uit waarom de ontwerpers van de databank voor deze primaire sleutel gekozen zouden hebben en wat de consequenties met betrekking tot dataopslag zouden zijn wanneer je de primaire sleutel zou aanpassen naar `{schapnummer}`.

2. Uit de huidige implementatie is het voor de eindgebruiker niet onmiddelijk duidelijk waarom de triggerfunctie `func` een exceptie geeft. Vervang de `...` in de huidige triggerfunctie met een eigen boodschap die duidelijk maakt waarom deze exceptie voorkomt en wat er al dan niet verandert aan de databank wanneer deze exceptie zich voordoet.

3. Beschouw onderstaande sequentie SQL-statements.

```
1. INSERT INTO boekenkast VALUES (1, 12.5);
2. INSERT INTO boekenkast VALUES (2, 15.0);
3. INSERT INTO schap VALUES (1, 1, 'roman');
4. INSERT INTO schap VALUES (2, 0, 'thriller');
5. INSERT INTO schap VALUES (2, 1, 'sci-fi');
6. INSERT INTO boek VALUES ('Love', 'beschrijving 1', 5.6, 1, 1);
7. INSERT INTO boek (titel, breedte, kastnummer, schapnummer)
   VALUES ('Love 2', 4.5, 1, 1);
8. INSERT INTO boek (title, beschrijving, kastnummer, schapnummer)
   VALUES ('Love 3', 'beschrijving 3', 1, 1);
9. INSERT INTO boek (titel, breedte, kastnummer, schapnummer)
   VALUES ('The Da Vinci Code', 8.3, 2, 0);
10. INSERT INTO boek (titel, breedte, kastnummer, schapnummer)
   VALUES ('Lord of the rings 1', 9.8, 2, 1);
11. INSERT INTO boek (titel, breedte, kastnummer, schapnummer)
    VALUES ('Lord of the rings 2', 7.3, 2, 1);
12. INSERT INTO boek (titel, breedte, kastnummer, schapnummer)
    VALUES ('Lord of the rings 3', 4.7, 2, 1);
```

   Ervan uitgaande dat elk van deze SQL-statements afzonderlijk wordt uitgevoerd in de gegeven volgorde, geef een lijst terug van alle SQL-statements die falen en de reden waarom deze falen.

## Appendix 1: DDL-code `bibliotheek` databank

```
CREATE TABLE boekenkast (
    kastnummer INTEGER PRIMARY KEY CHECK (kastnummer > 0),
    breedte NUMERIC NOT NULL CHECK (breedte > 0)
);

CREATE TABLE schap (
    kastnummer INTEGER REFERENCES boekenkast(kastnummer),
    schapnummer INTEGER CHECK (schapnummer > 0),
    genre VARCHAR NOT NULL,
    CONSTRAINT schap_pk PRIMARY KEY(kastnummer, schapnummer)
);

CREATE TABLE boek (
    titel VARCHAR PRIMARY KEY,
    beschrijving VARCHAR,
    breedte NUMERIC NOT NULL CHECK (breedte > 0),
    kastnummer INTEGER NOT NULL,
    schapnummer INTEGER NOT NULL,
    CONSTRAINT boek_schap_fk FOREIGN KEY(kastnummer, schapnummer)
        REFERENCES schap(kastnummer, schapnummer)
);

CREATE FUNCTION func() RETURNS trigger AS $BODY$
DECLARE
    b1 NUMERIC;
    b2 NUMERIC;
BEGIN
    SELECT sum(breedte) INTO b1 FROM boek
        WHERE kastnummer = NEW.kastnummer AND schapnummer = NEW.schapnummer;

    SELECT breedte INTO b2 FROM boekenkast
        WHERE kastnummer = NEW.kastnummer;

    IF (b1 + NEW.breedte > b2) THEN
        RAISE EXCEPTION '...';
    END IF;

    RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER trigger1 BEFORE INSERT ON boek FOR EACH ROW EXECUTE PROCEDURE func();
```