Examen **Databanken** Academiejaar 2019–2020
Prof. dr. Guy De Tré 1e zittijd: 16/01/2020

Het examen is volledig open boek. Er mag gebruik gemaakt worden van al het materiaal voorzien tijdens de lessen. Dit houdt in:

- Het boek ‘Principes van databases’ van prof. dr. Guy De Tré
- Alle slides, opgaves, (opgeloste) oefeningen, notities, examenvoorbereidingen,. . .

Het examen kan worden opgelost op lege witte bladeren. Vermeld zeker op **elk** oplossingsblad je studentennummer.

1

Vraag 1

Waarom gebruiken veel financiële instellingen pre-relationele databanktechnologie voor het beheer van financiële transacties?

Vraag 2

Geef twee belemmeringen waardoor objectgeoriënteerde databanktechnologie de relationele databanktechnologie niet kon verdringen.

2

Sleutels spelen een belangrijke rol in veel databankmodellen.

Vraag 1

Waarom moet een sleutel voldoen aan de irreducibiliteitsvoorwaarde?

Vraag 2

Met welk sleutelconcept uit het relationeel databankmodel correspondeert het sleutelconcept uit het ODMG model?

Vraag 3

Waarom mogen primaire sleutels geen `NULL`-waarden bevatten en vreemde sleutels wel?

3

Tijd	T1	T2	T3	T4
1	S(A)			
2		X(B)		
3				X(C)
4				S(B)
5			S(A)	
6		X(A)		
7				

Tabel 1: Sequentie van ‘lock’-aanvragen

Beschouw de volgende sequentie van ‘lock’-aanvragen op timestamps 1 tot en met 7 (zie Tabel 1), door transacties T1,. . . ,T4. Veronderstel dat het twee-fase locking protocol gebruikt wordt op tupleniveau. Geen enkele transactie krijgt een commit of rollback tijdens de beschouwde tijdsperiode. De betekenissen van de aanvragen is als volgt:

- S(A): de transactie vraagt (maar krijgt niet noodzakelijk) een gedeelde lock op tuple A.
- X(A): de transactie vraagt (maar krijgt niet noodzakelijk) een exclusieve lock op tuple A.

Vraag 1

Teken de ‘wacht-voor’-graaf voor deze sequentie van ‘lock’-aanvragen.

Vraag 2

Waar of niet waar? Er zijn **geen deadlocks** in deze sequentie van ‘lock’-aanvragen. Verklaar.

Vraag 3

Waar of niet waar? Als T1 een gedeelde lock vraag op tuple B op timestamp 7 dan zijn er **geen deadlocks**. Verklaar.

4 Ontwerp

De UGent komt naar jou met de vraag om een databank te ontwikkelen ter ondersteuning van hun systeem voor lokaalreservaties. Stel op basis van onderstaande gegevens een correct en volledig conceptueel ontwerp op.

De UGent heeft verschillende lokalen ter beschikking. Deze lokalen zijn verspreid over verschillende gebouwen die zich bevinden op campussen. Voor ieder gebouw is de combinatie van de naam van de campus met de naam van het gebouw uniek. Verder moet voor ieder gebouw het adres en een centaal telefoonnummer worden opgeslagen.
Per gebouw zijn er een aantal lokalen ter reservatie beschikbaar. Deze lokalen hebben elk een lokaalnummer dat uniek is binnen een gebouw. Daarnaast moet voor elk lokaal de naam, de capaciteit, de aanwezigheid van een projectiescherm en de toegankelijkheid voor mindervaliden worden bijgehouden. Ook heeft ieder lokaal specifieke openingsuren die bestaan uit een datum, een starttijdstip en een eindtijdstip. Openingstijden van eenzelfde lokaal mogen uiteraard niet overlappen. Tot slot bestaan er drie verschillende types lokalen: PC-lokalen, auditoria en vergaderzalen. In het geval van PC-lokalen moet worden bijgehouden hoeveel PC's er beschikbaar zijn en in het geval van auditoria moet worden opgeslagen of het om een plat of een steil auditorium gaat.
Personen kunnen lokalen boeken. Deze boekingen kunnen zowel gedaan worden door UGent personeel (internen) alsook door externen. Voor iedere persoon moet een uniek emailadres, de voor- en achternaam, een telefoonnummer en een adres worden opgeslagen. Voor internen dient bijkomend een functieomschrijving te worden opgeslagen.
Iedere boeking wordt bijgehouden, samen met de periode waarin men het lokaal wil boeken (gedefinieerd door een starttijdstip en een eindtijdstip) en het aantal personen dat men ongeveer verwacht. Uiteraard moet dit tijdstip liggen binnen de openingstijden van het lokaal (in het geval van een boeking voor meerdere dagen, moeten dus meerdere boekingen worden geregistreerd) en moet het verwacht aantal personen lager liggen dan de maximale capaciteit van het lokaal. Verder moet een beschrijving worden opgeslagen van de activiteit die men wil organiseren in het lokaal. Ieder verhuurbaar lokaal heeft een basistarief dat bij een verhuur door internen betaald moet worden en een prijsfactor die wordt aangerekend in het geval van externen. De totaalprijs voor iedere afzonderlijke verhuur moet eenvoudig opgevraagd kunnen worden uit de databank.

5 Gebruik

In Appendix 1 is een PostgreSQL DDL-script gegeven voor de opbouw van een chatdatabank. Hieronder volgen er een aantal vragen in verband met het gebruik van deze databank. Je mag voor alle vragen veronderstellen dat de volgende twee SQL-commando's reeds zijn uitgevoerd.

```
1. INSERT INTO gebruiker VALUES ('piet.pieters@ugent.be', 'Piet', 'Pieters');
2. INSERT INTO gebruiker VALUES ('jan.janssens@ugent.be', 'Jan', 'Janssens');
```

Vraag 1

	zender_email	**tijd**	**inhoud**	**type**
1	piet.pieters@ugent.be	10/01/2020 18:42:15	inhoud1	Tekst
2	jan.janssens@ugent.be	`NULL`	inhoud2	Media
3	john.doe@ugent.be	10/01/2020 19:15:12	inhoud3	Tekst
4	jan.janssens@ugent.be	10/01/2020 19:19:54	inhoud2	Video
5	piet.pieters@ugent.be	10/01/2020 19:19:54	inhoud4	Tekst
6	jan.janssens@ugent.be	10/01/2020 19:19:54	inhoud2	Media

Tabel 2: Data die men probeert te importeren in vraag 2.1

In Tabel 2 is data gegeven die geïmporteerd moeten worden in de relatie `bericht`, nadat de sequentie SQL-commando's van hierboven al werd uitgevoerd. Welke rijen in de gegeven tabel zullen voor een foutmelding zorgen bij het importeren? Geef per falende rij het rijnummer, een beschrijving van de foutmelding en een gedetailleerde verklaring waarom deze foutmelding wordt teruggegeven. Je mag veronderstellen dat we buiten de gegeven toevoegingen vertrekken van een lege databank, de data in de gegeven tabel correct geformateerd zijn en de rijen in de gegeven volgorde een voor een worden toegevoegd.

Vraag 2

In Figuur 1 is een (genummerde) sequentie van SQL-commando's gegeven die inwerken op de chatdatabank. Hoe zal de resultatentabel `view1` eruit zien na uitvoering van het laatste commando? Teken deze tabel **volledig**. Je mag veronderstellen dat we vertrekken van een lege databank en de syntax van de commando's correct is. Ook worden alle statements een voor een uitgevoerd (en dus niet in één keer). Statements die falen mag je dus negeren. We houden geen rekening met de correcte formatering van waarden in de getekende tabel.

```
1. INSERT INTO bericht VALUES
        (‘jan.janssens@ugent.be’, ‘15/12/2019 10:30:02’::timestamp,
        ‘bericht1’, ‘Tekst’);
2. INSERT INTO ontvangers VALUES
        (‘jan.janssens@ugent.be’, ‘15/12/2019 10:30:02’::timestamp,
        ‘piet.pieters@ugent.be’, false);
3. INSERT INTO bericht VALUES
        (‘piet.pieters@ugent.be’, ‘15/12/2019 10:32:44’::timestamp,
        ‘bericht2’, ‘Media’);
4. INSERT INTO bericht VALUES
        (‘piet.pieters@ugent.be’, ‘15/12/2019 10:34:12’::timestamp,
        ‘bericht3’, ‘Media’);
5. INSERT INTO ontvangers VALUES
        (‘piet.pieters@ugent.be’, ‘15/12/2019 10:32:44’::timestamp,
        ‘jan.janssens@ugent.be’, false);
6. INSERT INTO ontvangers VALUES
        (‘piet.pieters@ugent.be’, ‘15/12/2019 10:34:12’::timestamp,
        ‘jan.janssens@ugent.be’, true);
7. INSERT INTO ontvangers VALUES
        (‘piet.pieters@ugent.be’, ‘15/12/2019 10:32:44’::timestamp,
        ‘piet.pieters@ugent.be’, true);
8. UPDATE gebruiker SET email = ‘piet_pieters@gmail.com’
       WHERE email = ‘piet.pieters@ugent.be’;
9. SELECT * FROM view1;
```

Figuur 1: SQL-commando’s behorend bij vraag 2.2

Vraag 3

Stel dat je een extra attribuut `gelezen_op` wil toevoegen aan de relatie `ontvangers` dat aangeeft wanneer een ontvanger een bericht heeft gelezen. Dit attribuut is van het datatype `timestamp`. Geef een overzicht van alle beperkingen die je op de waarden van dit attribuut moet leggen.

6 Analyse

In Appendix 2 is een logisch databankschema te vinden van (een deel van) de Resto databank. Hieronder is een SELECT-query gegeven die inwerkt op deze databank. Analyseer de SELECT-query aandachtig en beantwoord vervolgens onderstaande vragen.

```
1.     SELECT DISTINCT productnaam
2.     FROM productsoort
3.         INNER JOIN aanbod USING(productnaam, soortnaam)
4.     WHERE productnaam NOT IN (
5.         SELECT productnaam FROM aanbod a1
6.         WHERE a1.restonaam = 'Resto De Brug'
7.             AND EXISTS(
8.                 SELECT 1 FROM aanbod a2
9.                 WHERE a2.restonaam = 'Resto Campus Merelbeke'
10.                    AND a1.productnaam = a2.productnaam
11.            )
12.    );
```

Vraag 1

De tabel die het resultaat is na het uitvoeren van bovenstaande query is een antwoord op een specifieke analysevraag. Geef een gedetailleerde en een zo volledig mogelijke beschrijving van deze vraag.

Vraag 2

Een aantal zaken in de gegeven query zijn overbodig of kunnen geoptimaliseerd worden. Geef 2 mogelijke optimalisaties die ervoor zouden kunnen zorgen dat de uiteindelijke kost zal dalen zonder dat de resultatentabel verschilt. Geef een gedetailleerde en een zo volledig mogelijke beschrijving van elke voorgestelde optimalisatie (zodat we eenduidig kunnen bepalen of dit correct is). Je mag ervan uit gaan dat de code correct uitvoert. Ga dus niet op zoek naar mogelijke syntaxfouten.

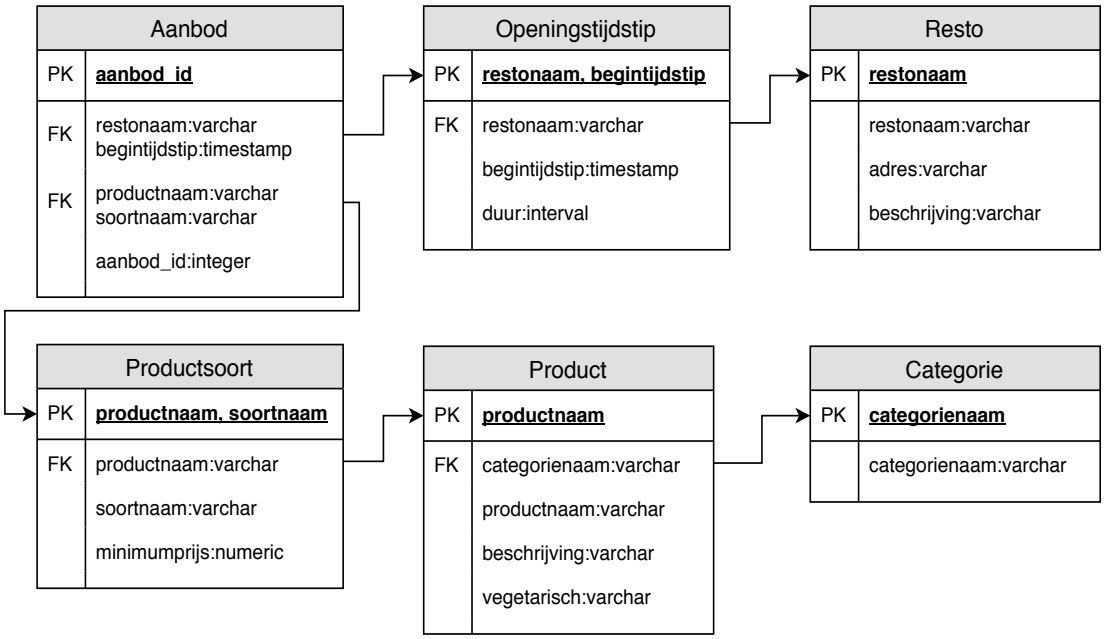
Appendix 1: DDL-script chatdatabank

```
CREATE TABLE gebruiker (
    email character varying,
    voornaam character varying,
    achternaam character varying,
    CONSTRAINT gebruiker_pkey PRIMARY KEY (email)
);

CREATE TABLE bericht (
    zender_email character varying,
    tijd timestamp without time zone,
    inhoud character varying(100) NOT NULL,
    type character varying NOT NULL,
    CONSTRAINT bericht_pkey PRIMARY KEY (zender_email, tijd),
    CONSTRAINT bericht_gebruiker_fkey
        FOREIGN KEY (zender_email) REFERENCES gebruiker(email),
    CONSTRAINT bericht_type_check
        CHECK (type IN ('Tekst','Media','Document'))
);

CREATE TABLE ontvangers (
    zender_email character varying,
    tijd timestamp without time zone,
    ontvanger_email character varying,
    gelezen boolean NOT NULL,
    CONSTRAINT ontvangers_pkey
        PRIMARY KEY (zender_email, tijd, ontvanger_email),
    CONSTRAINT ontvangers_bericht_fkey
        FOREIGN KEY (zender_email, tijd) REFERENCES bericht(zender_email, tijd),
    CONSTRAINT ontvangers_gebruiker_fkey
        FOREIGN KEY (ontvanger_email) REFERENCES gebruiker(email)
);
```

```
CREATE FUNCTION check_ontvangers()
    RETURNS trigger
    LANGUAGE plpgsql
AS $$
BEGIN

    IF (NEW.zender_email = NEW.ontvanger_email) THEN

        DELETE FROM ontvangers
            WHERE zender_email = NEW.zender_email AND tijd = NEW.tijd;
        DELETE FROM bericht
            WHERE zender_email = NEW.zender_email AND tijd = NEW.tijd;

        RETURN NULL;

    ELSE
        RETURN NEW;
    END IF;

END;
$$;

CREATE TRIGGER check_ontvangers_trigger
    BEFORE INSERT ON ontvangers
    FOR EACH ROW EXECUTE PROCEDURE check_ontvangers();

CREATE VIEW view1 AS
    SELECT g.email, g.voornaam, g.achternaam, COUNT(*)
        FROM gebruiker g
        INNER JOIN ontvangers o ON g.email = o.ontvanger_email
    WHERE NOT o.gelezen GROUP BY g.email;
```

Appendix 2: Logisch databankontwerp Resto

In onderstaande figuur vind je een schematische weergave van een mogelijk logisch ontwerp voor (een deel van) de Resto databank. Hierbij wordt iedere basisrelatie weergegeven door een rechthoek, die bovendien een oplijsting van alle attributen met bijhorend datatype bevat. Daarnaast worden de attributen die behoren tot de primaire sleutel bovenaan weergegeven, en worden vreemde sleutels voorgesteld door een pijl tussen de betreffende attribuutverzamelingen. Alle extra beperkingen die niet kunnen weergegeven worden in dit schema worden onderaan opgelijst.



Extra beperkingen

1. Resto:
    - `UNIQUE` en `NOT NULL`: adres
    - Optioneel: beschrijving
2. Product:
    - `CHECK`: vegetarisch $\in$ {'geen van beide', 'vegetarisch', 'veganistisch'}
    - Optioneel: beschrijving
3. Productsoort:
    - `CHECK`: minimumprijs ≥ 0
4. Openingstijdstip:

- `CHECK`: duur > 0
- Insert: check dat openingsperiode niet overlapt met andere openingsperiode van dezelfde resto.

5. Aanbod:
    - Insert: check dat, indien de openingsperiode verder loopt dan 11u30, er geen producten uit categorie ‘Ontbijt’ aangeboden worden.