Tentamen Databases —5 april 2004

De gecorrigeerde tentamens zijn af te halen bij het Onderwijsbureau, kamer 36 IWI, in de (rode) map Databases, resp. bij het Buro Onderwijs en Examens van TBK/TM, WSN 640.

Opmerkingen:

- Schrijf **netjes** en duidelijk, met zwarte of blauwe pen.
- Zet op het eerste blad alle gegevens als naam, studentnummer etc., en het totaal aantal ingeleverde bladen, en nummer de ingeleverde bladen.
- LET GOED OP WELKE OPGAVEN U NIET HOEFT TE MAKEN! Dit tentamen bevat opgaven voor het hertentamen oude stijl (de Brock), en opgaven voor het tentamen nieuwe stijl (Balsters).
- Dit is een gesloten boek tentamen, hetgeen inhoudt dat u **geen** gebruik mag maken van een boek en/of aantekeningen tijdens het tentamen.
- Bij multiple choice vragen is er slechts één goed antwoord.
- Motiveer uw antwoorden bij open vragen.
- Niet alle opgaven tellen even zwaar mee!

1. **(Alleen voor Informatici, oude en nieuwe stijl)**

   Gegeven is de tabel-heading $R\{A, B, C, D, E\}$, en de verzameling functionele afhankelijkheden $F = \{AB \rightarrow C, DE \rightarrow C, B \rightarrow D\}$.
   Welke van onderstaande decomposities levert relaties in BCNF?
   a) $\{\{C, D\}, \{A, B, C\}, \{B, E\}\}$
   b) $\{\{D, B\}, \{C, D\}, \{A, E\}\}$
   c) $\{\{B, D\}, \{A, B, E\}, \{A, B, C\}\}$
   d) geen van a), b) of c).

2. **(Alleen voor Informatici, oude en nieuwe stijl)**

   Welke van de volgende beweringen is waar?
   a) 3NF impliceert BCNF
   b) BCNF impliceert zowel 3NF als 4NF
   c) 3NF samen met 4NF impliceert BCNF
   d) noch a), b) of c).

3. **(Alleen voor Informatici, oude en nieuwe stijl)**

Welke van de volgende beweringen is waar?
a) een dependency preserving decompositie heeft de nonadditive join eigenschap
b) een lossless join decompositie is in derde normaalvorm
c) in een BCNF tabel is alles uitsluitend functioneel afhankelijk van de primaire sleutel
d) er bestaat altijd een lossless join dependency preserving decompositie in BCNF

4. **(Zowel voor Informatici als TBK/TM nieuwe stijl)**

Database architectuur.
a) Beschrijf de 3-schema architectuur
b) Waarom zijn er binnen de 3-schema architectuur afbeeldingen nodig tussen de verschillende schemaniveaus?
c) Op welke wijze ondersteunen verschillende schemadefinities de 3-schema architectuur?

5. **(Zowel voor Informatici als TBK/TM nieuwe stijl)**

Conceptueel ontwerp en afbeelding naar het relationele model

Beschouw de volgende informele beschrijving van een bank:

- De bank is georganiseerd in filialen. Elk filiaal is gelokaliseerd in een specifieke plaats en geïdentificeerd door een unieke naam. Van elk filiaal wordt bijgehouden welke diensten het levert.
- Klanten van de bank worden geïdentificeerd door hun sofi-nummer. Van een klant wordt bijgehouden naam, alsmede straat en huisnummer waar de klant woont. Een klant kan diverse rekeningen hebben, en kan leningen aangaan bij de bank. Een klant kan gekoppeld zijn aan een specifieke werknemer (de zogenaamde *contactpersoon* van de bank.
- Werknemers van de bank worden geïdentificeerd door hun sofinummer. Van een werknemer wordt bijgehouden naam, alsmede telefoonnummer, namen van de gezinsleden van de werknemer, en het sofi-nummer van de betrokken manager van de werknemer. Verder worden van een werknemer bijgehouden de datum van indiensttreding en salarisschaal.
- De bank biedt twee soorten rekeningen, te weten spaarrekeningen en lopende rekeningen. Rekeningen kunnen op naam staan van meerdere klanten, en een klant kan meerdere rekeningen hebben. Elke rekening heeft een uniek nummer. Verder wordt van

een rekening bijgehouden het saldo en de laatste datum dat een klant met de betrokken rekening van de rekening gebruik heeft gemaakt. Van een spaarrekening wordt bijgehouden het rentepercentage, alsmede alle stortingen en opnames.

- Een lening wordt verleend bij een specifiek filiaal, en kan worden gekoppeld aan meerdere klanten. Een lening heeft een uniek leningnummer. Van elke lening wordt bijgehouden de hoogte van het geleende bedrag, alsmede de aflossingen die hebben plaatsgevonden. Elke aflossing heeft een aflossingsnummer dat samen met het eraan gekoppelde leningnummer uniek identificerend is voor de aflossing in kwestie. Van elke aflossing wordt tevens bijgehouden de datum en het bedrag van de aflossing.

De opdrachten:
a) Maak een E(E)R-ontwerp van deze bank
b) Construeer een relationeel database schema corresponderend met het E(E)R-ontwerp.

6. **(Alleen voor Informatici, oude en nieuwe stijl)**

Beschouw het volgende **netwerk**model:

```
RECORD NAME IS Passagier
   LOCATION MODE IS CALC USING Pnr
       DUPLICATES ARE NOT ALLOWED FOR Pnr
       Pnr     IS TYPE Integer
       Naam    IS TYPE String(20)
       Gironr  IS TYPE String(7)
       Betaald IS TYPE Boolean

RECORD NAME IS Vlucht
   LOCATION MODE IS CALC USING Vnr
       DUPLICATES ARE NOT ALLOWED FOR Vnr
       Vnr     IS TYPE Integer
       Van     IS TYPE String(20)
       Naar    IS TYPE String(20)

RECORD NAME IS Piloot
   LOCATION MODE IS CALC USING Naam
       DUPLICATES ARE NOT ALLOWED FOR Naam
       Naam    IS TYPE String(20)
       Brevet  IS TYPE String(10)

SET NAME IS Boeking
       OWNER IS Vlucht
```

```
        MEMBER IS Passagier
               INSERTION IS MANUAL
               RETENTION IS OPTIONAL

SET NAME IS Vliegt
        OWNER IS Piloot
        MEMBER IS Vlucht
               INSERTION IS MANUAL
               RETENTION IS OPTIONAL
```

Maak een applicatie (netwerk-query), die het volgende beantwoordt: "Geef de namen van de piloten die een vlucht maken van Schiphol met een passagier met de naam Jansen aan boord."

(Hint: U dient dus gebruik te maken van commando's als `FIND FIRST, FIND NEXT, FIND OWNER, GET` etc.etc.)

Alleen voor Informatici/TBW/TBK oude stijl:
We definiëren een database–universum UBP in een bijlage (vraag deze aan de surveillanten!), ter registratie van de medewerkersinzet bij projekten van de afdelingen van een organisatie.

(Alleen voor TBW/TBK oude stijl)

Geef, uitgaande van het database-universum UBP gedefinieerd in de bijgevoegde bijlage (afkomstig uit het boek), de volgende query in gewoon nederlands weer:

$$\lambda v \in UBP\text{: } (\sum t \in v(PROD) \text{ en } 2 < \mid \{y(SRT) \mid y \in v(PROD) \text{ en } \exists x \in v(BOM) : (x(PNR) = t(PNR) \text{ en } x(ONR) = y(PNR))\} \mid : t(VRD))$$

(Zowel voor TBW/TBK als Informatici oude stijl)

Geef van de formele weergave van de query in opgave 7 een zo rechtstreeks mogelijke "doorvertaling" naar SQL(2).

(Zowel voor TBW/TBK als Informatici oude stijl)

Geef de hieronder gevraagde query formeel weer in termen van het database-universum UBP:

"Geef het totale aantal exemplaren van in voorraad zijnde produkten die een produkt van de soort 'AUX' als direct onderdeel hebben."

Hierna volgen alleen nog sommen voor het tentamen nieuwe stijl!

(10) **(Zowel voor Informatici als TBK/TM nieuwe stijl)**

Stel dat R, S en T relaties zijn die elk uitsluitend het attribuut A bevatten. Welke van de onderstaande SQL-queries berekent onder alle omstandigheden op de juiste wijze de relatie $R \cap (S \cup T)$?

```
a) SELECT x
   FROM R as x
   WHERE EXISTS(SELECT y
                FROM S as y, T as z
                WHERE x.A=y.A OR x.A = z.A)

b) SELECT x
   FROM R as x, S as y, T as z
   WHERE x.A=y.A OR x.A=z.A

c) SELECT x
   FROM R as x
   WHERE EXISTS(SELECT y FROM S as y WHERE x.A=y.A) OR
         EXISTS(SELECT z FROM T as z WHERE x.A=z.A) OR

d) noch a), b) of c).
```

(11) **(Alleen voor TBK/TM nieuwe stijl)**

Beschouw het volgende database schema

Product(maker, model)
PC(model, prijs)
Laptop(model, schermgrootte, prijs)

Hierbij wordt de view V gegeven door

```
CREATE VIEW V AS
   (PC UNION (SELECT x.model,x.prijs FROM Laptop as x)
```

Welke van de onderstaande SQL-queries berekent op de juiste wijze de volgende informele vraag: "geef alle makers van producten (PC of laptop) met de laagste prijs?"

```
a) SELECT x.maker
   FROM Product as x
   WHERE x=(SELECT MIN(prijs) FROM V)

b) SELECT x.maker
   FROM Product as x
   WHERE ALL(SELECT y.prijs
             FROM V as y
             WHERE x.prijs<=y.prijs)

c) SELECT x.maker
   FROM Product as x
   WHERE EXISTS(SELECT y
                FROM V as y
                WHERE x.model=y.model AND
                      NOT EXISTS(SELECT z
                                 FROM V as z
                                 WHERE y.prijs>z.prijs))

d) noch a), b) of c).
```

12 **(Zowel voor Informatici als TBK/TM nieuwe stijl)**

Beschouw het database schema van vraag 11. Welke van de onderstaande SQL-queries berekent op de juiste wijze de volgende informele vraag: "vind alle makers van laptops die tevens PC's maken tegen een hogere prijs dan deze laptop."

```
a) SELECT x.maker
   FROM Product as x, Laptop as y
   WHERE x.model=y.model AND
         NOT EXISTS(SELECT z.model
                    FROM PC as z
                    WHERE x.model=z.model AND z.prijs<=y.prijs)

b) SELECT x.maker
   FROM Product as x, Laptop as y, PC as z
   WHERE x.model=y.model AND x.model=z.model AND z.prijs>y.prijs

c) SELECT x.maker
   FROM Product as x
   WHERE EXISTS(SELECT y
                FROM Laptop as y
                WHERE x.model=y.model AND
```

```
                    EXISTS(SELECT z.model
                           FROM PC as z
                           WHERE x.model=z.model AND z.prijs>y.prijs))
```

d) noch a), b) of c).

13. **(Alleen voor TBK/TM nieuwe stijl)**

Beschouw de volgende update-statement

```
UPDATE R
SET b=b-a, a=a-b
WHERE b>a
```

waarbij R de relatie is met de volgende tabelinhoud

a	b
2	5
3	7
7	5

Hoe ziet de inhoud van R eruit na de update-statement?

a)

a	b
-1	3
-1	4
9	2

b)

a	b
-3	3
-4	4
2	-2

c)

a	b
-3	8
-4	11
2	3

d) noch a), b) of c).

14. **(Alleen voor TBK/TM nieuwe stijl)**

Beschouw het database schema van vraag 11. Welke van de onderstaande SQL-queries berekent op de juiste wijze de volgende informele vraag: "vind per maker die wel PC's maar geen laptops met schermgrootte kleiner dan 17 maken, de minimale prijs van hun PC's."

```
a) SELECT maker MIN(prijs)
   FROM Product x, PC
   WHERE PC.model=x.model
   GROUP BY maker
   HAVING COUNT(SELECT *
                FROM Laptop y
                WHERE y.model=x.model AND y.schermgrootte<17)=0

b) CREATE VIEW V AS
   SELECT *
   FROM Product
   WHERE NOT EXISTS(SELECT *
                    FROOM Laptop
                    WHERE Laptop.model=Product.model AND
                          Laptop.schermgrootte<17);

   SELECT maker MIN(prijs)
   FROM V as x, PC as y
   WHERE x.model=y.model
   GROUP BY maker

c) SELECT maker MIN(prijs)
   FROM Laptop x, PC, Product z
   WHERE x.model=z.model AND PC.model=z.model AND x.schermgrootte>=17
   GROUP BY maker

d) noch a), b) of c).
```

15 **(Zowel voor Informatici als voor TBK/TM nieuwe stijl)**

Beschouw het database schema van vraag 11. Welke van de onderstaande update-statements geeft op de juiste wijze de volgende modificatie weer: "voor elke laptop gemaakt door een fabrikant die geen PC's maakt, verhoog de schermgrootte met 20%?"

```
a) UPDATE Laptop
   SET schermgrootte=(1.2 * schermgrootte)
   WHERE model IN (SELECT model
                   FROM Product p
                   WHERE NOT EXISTS(SELECT *
                                    FROM PC
                                    WHERE PC.model=p.model))

b) UPDATE Laptop, Product
```

```
   SET schermgrootte=(1.2 * schermgrootte)
   WHERE Laptop.model<>Product.model

c) UPDATE schermgrootte BY 1.2
   FROM Laptop
   WHERE NOT EXIST(SELECT model
                   FROM Product
                   WHERE Product.model=Laptop.model)

d) noch a), b) of c).
```

~~16.~~ **(Alleen voor TBK/TM nieuwe stijl)**

Beschouw de relaties R(a,b) en S(b,c) beide met uitsluitend integer waardige attributen, met daarop gedefinieerd de volgende view

```
CREATE VIEW V AS
SELECT a,c
FROM R, S
WHERE R.b=S.b;
```

Stel men wil een insert-opdracht plegen op de view V, volgens

```
INSERT INTO V
VALUES (4,2)
```

Wat is dan het effect van deze insert-opdracht op de relaties R en S?
a) R, S worden resp. uitgebreid met de tupels (4,4) en (2,2)
b) R, S worden resp. uitgebreid met de tupels (4,2) en (2,2)
c) R, S worden resp. uitgebreid met de tupels (4,NULL) en (NULL,2)
d) noch a), b) of c).

~~17.~~ **(Alleen voor TBK/TM nieuwe stijl)**

Stel R(a,b) is een relatie met de volgende tabelinhoud

a	b
5	1
2	NULL
3	3

Wat is de uitkomst van de volgende query:

```
SELECT a
FROM R
WHERE a>2 and b<>0?
```

a) $\{2, 3, 5\}$
b) $\{3, 5\}$
c) $\emptyset$
d) noch a), b) of c).

18 **(Alleen TBK/TM nieuwe stijl)**

Beschouw het database schema van vraag 11. Welke van onderstaande constraints geeft op de juiste wijze de volgende eis weer: "minstens één fabrikant maakt zowel een PC als een laptop?"

a) *attribute constraint* op relatie Product:

```
model STRING CHECK
      model IN (SELECT model FROM
                (PC UNION (SELECT x.model, x.prijs
                           FROM Laptop as x))
```

b) *global constraint*:

```
CHECK EXISTS(SELECT *
             FROM Product x, PC y
             WHERE x.model=y.model AND
                   EXISTS(SELECT *
                          FROM Laptop z
                          WHERE x.model=z.model))
```

c) *referential integrity constraint* vanuit de relatie Product:

```
model STRING REFERENCES Laptop(model) NOT NULL;
model STRING REFERENCES PC(model) NOT NULL
```

d) noch a), b) of c).

19 **(Zowel voor Informatici als TBK/TM nieuwe stijl)**

Beschouw het database schema van vraag 11. Welke van onderstaande constraints geeft op de juiste wijse de volgende eis weer: "geen twee PC's hebben dezelfde fabrikant."

a) *global constraint*:

```
COUNT(SELECT maker
      FROM Product, PC
      WHERE Product.model=PC.model
      GROUP BY maker
      HAVING COUNT(*)>1) = 0
```

b) *attribute constraint* op Product:

```
model STRING CHECK
   model NOT IN (SELECT x.model
                 FROM PC as x, PC as y
                 WHERE Product.model=x.model AND x.model<>y.model
```

c) *attribute constraint* op PC:

```
model STRING CHECK
   COUNT(SELECT model FROM Product
         WHERE Product.model=PC.model)=1
```

d) noch a), b) of c).

~~20.~~ **(Alleen voor TBK/TM nieuwe stijl)**

Beschouw het database schema van vraag 11. Welke van onderstaande constraints geeft op de juiste wijse de volgende eis weer: "PC fabrikanten en Laptop fabrikanten zijn altijd verschillend."

a) *attribute constraint* op PC:

```
model STRING CHECK
   model NOT IN (SELECT model FROM Laptop)
```

b) *global constraint*:

```
COUNT(SELECT maker
      FROM Product
      WHERE model IN ((SELECT model FROM Laptop) INTERSECT
                      (SELECT model FROM PC)))=0
```

c) *global constraint*:

```
COUNT(SELECT maker
      FROM Product
      WHERE model IN ((SELECT model FROM Laptop) UNION
                      (SELECT model FROM PC)))=1
```

d) noch a), b) of c).