Departement Informatica en Informatiekunde, Faculteit Bètawetenschappen, UU.
In elektronische vorm beschikbaar gemaakt door de TBC van A–Eskwadraat.
Het college INFODB werd in 2009-2010 gegeven door drs. H. Philippi.

Databases (INFODB)
20 april 2010

Bij elke vraag wordt verwacht dat je laat zien hoe je aan het antwoord komt (tenzij anders wordt vermeld). Je mag een A4 met aantekeningen raadplegen.

Veelgebruikte afkortingen:

2PL Two-phase locking

2PC Two-phase commit

BCN F Boyce-Codd normaalvorm

COORD Coordinator

CTP Cooperative termination protocol

DP dependency preserving

FD functional dependency

RA Relationele algebra

SQL SQL (Structured Query Language)

Puntentelling:

1: 20 punten

2: 15 punten

3: 15 punten

4: 20 punten

5: 15 punten

6: 15 punten

Opgave 1 Algemeen

Geef van de volgende beweringen aan of zij correct zijn of niet. Een simpel JA of NEE volstaat. Er hoeft geen toelichting gegeven te worden.

1. Als elke FD $X \rightarrow Y$ uit de oorspronkelijke FD-set F in één van de relatieschema's past, is dat een voldoende voorwaarde voor de DP-eigenschap.
2. Dat elke FD $X \rightarrow Y$ uit de oorspronkelijke FD-set F in één van de relatieschema's past, is een noodzakelijke voorwaarde voor de DP-eigenschap.
3. Gegeven een FD-set F. Er bestaat slechts één unieke minimal cover van F.
4. Het recovery-mechanisme op basis van UNDO geeft de IO-manager de meeste vrijheid.
5. Het recovery-mechanisme op basis van REDO geeft de IO-manager de meeste vrijheid.

6. Met Datalog kun je queries formuleren die in SQL niet uit te drukken zijn.
7. Het is cruciaal dat je een log-file op een aparte disk onderbrengt.
8. Logging speelt een cruciale rol bij media failures.
9. Het 2PC-protocol kan geblokkeerd, raken als één van de participants nog niet gestemd heeft.
10. De B-tree is een geschikte indexstruetuur voor range-queries.

Opgave 2 Functionele afhankelijkheden

Twee attribute sets X en Y zijn equivalent met betrekking tot een set FDs F als $X \rightarrow Y$ en $Y \rightarrow X$ beide gelden.

a) Beschrijf kort hoe je equivalentie via een algoritme kunt testen.

b) Stel we hebben de volgende set FDs:
$\{A- > BC, G- > EC, GC- > H, GH- > A\}$
Geef een attribute set die equivalent is met $\{A\}$.Licht uw antwoord kort toe.
$\{A\}$ wordt niet beschouwd als een correct antwoord.)

Opgave 3 Normaalvormen

Stel we hebben het relatieschema $S(ABCDEFGH)$.
$F_S = \{BCE \rightarrow D, DF \rightarrow B, D \rightarrow EG, G \rightarrow FHA\}$

a) Geef een verliesvrije, dependency-preserving 3NF-decompositic van S. Geef aan wat de keys zijn. Licht toe hoe u aan uw antwoord komt.

b) ls het opgeleverde schema ook in $BCNF$? Licht uw antwoord toe.

Opgave 4 Queries

Een zweefvliegclub heeft een database met daarin een tabel waarin gegevens van de vliegers worden bijgehouden, een tabel waarin gegevens van vliegtuigen worden bijgehouden en een tabel waarin gegevens van vluchten worden bijgehouden. Elke vlieger heeft een uniek brevetnummer. Daarnaast wordt van elke vlieger de naam, het geslacht en de geboortedatum bijgehouden. Van elk vliegtuig wordt het merk, het type en het bouwjaar geregistreerd. Eveneens heeft elk vliegtuig een uniek callsign (van de gedaante PH-123). De registratie van een vlucht gaat vergezeld van de datum, de starttijd, de duur (in minuten) en het starttype (sleepstart of lierstart). Het databaseschema is als volgt:

Vlieger (*brevetnr*, naam, geslacht, geboortedatum)
Vlucht (*brevetnr*, *callsign*, *datum*, *starttijd*, duur, starttype)
Vliegtuig (*callsign*, merk, type, bouwjaar)

We hebben de volgende queries:
Q1: Geef de namen van de vliegers die tenminste één vlucht gemaakt hebben.
Q2: Geef de namen en de totale vluchtduur voor de vliegers die het grootste aantal vluchten gemaakt hebben.
Q3: Geef de namen van de vliegers die in alle vliegtuigen gevlogen hebben.
Q4: Geef de namen van de vliegers die geen enkele vlucht gemaakt hebben.

Hieronder volgen expressies in de RA of in SQL. Geef aan welke queries corresponderen met welke expressies. De relatie tussen queries en expressies is many-to-many en optioneel.
E1: $\pi_{callsign}((\pi_{brevetnr,callsign}(\text{Vlucht}) \div \pi_{brevetnr}(\text{Vlieger})) \bowtie \text{Vliegtuig})$
E2: $\pi_{naam}((\pi_{brevetnr}(\text{Vlieger}) - \pi_{brevetnr}(\text{Vlucht})) \bowtie \text{Vlieger})$
E3: $\pi_{naam}((\text{Vlieger} \bowtie \text{Vlucht}) \div \pi_{callsign}(\text{Vliegtuig}))$
E4: $\pi_{naam}((\pi_{brevetnr}(\text{Vlieger}) \cap \pi_{brevetnr}(\text{Vlucht})) \bowtie \text{Vlieger})$

E5: π_{naam} (($\pi_{brevetnr,callsign}$(Vlucht) $\div$ $\pi_{callsign}$ (Vliegtuig)) $\bowtie$ Vlieger)
E6: π_{naam} (($\pi_{brevetnr}$ (Vlieger) $\cup$ (Vlucht)) $\bowtie$ Vlieger)

E7:

```
SELECT Vlieger.uaam
FROM Vlieger
WHERE Vlieger.brevetnr IN
     (SELECT Vlucht.brevetnr
     FROM Vlucht)
```

E8:

```
SELECT Vlieger.naam, SUM(Vlucht.duur)
FROM Vlieger, Vlucht
WHERE Vlieger.brevetnr = Vlucht.brevetnr
GROUP BY Vlucht.brevetnr
HAVING COUNT(*) > ALL
     (SELECT COUNT(*)
     FROM Vlucht
     GROUP BY Vlucht.brevetnr)
```

E9:

```
SELECT Vlieger.naam
FROM Vlieger
     WHERE NOT EXISTS
     (SELECT Vlucht.brevetnr
     FROM Vlucht)
```

E10:

```
SELECT Vlieger.naam, SUM(Vlucht.duur)
FROM Vlieger, Vlucht
WHERE Vlieger.brevetnr = Vlucht.brevetnr
GROUP BY Vlucht.brevetnr
HAVING MAx(COUNT(*))
```

Ell:

```
SELECT Vlieger.naam
FROM Vlieger
WHERE Vlieger.brevetnr NOT IN
     (SELECT Vlucht.brevetnr
     FROM Vlucht
     WHERE Vlucht.brevetnr <> Vlieger.brevetnr)
```

E12:

```
SELECT Vlieger.naam, SUM(Vlucht.duur)
FROM Vlieger, Vlucht
WHERE Vlieger.brevetnr = Vlucht.brevetnr
GROUP BY Vlucht.brevetnr
HAVING COUNT(*) >= ALL
     (SELECT COUNT(*)
     FROM Vlucht
```

El3:

```
SELECT Vlieger.naam
FROM Vlieger
WHERE NOT EXISTS
     (SELECT Vlucht.brevetnr
     FROM Vlucht
     WHERE Vlucht.brevetnr = Vlieger.brevetnr)
```

E14:

```
SELECT Vlieger.naam
FROM Vlieger
WHERE EXISTS
     (SELECT Vlucht.brevetnr
     FROM Vlucht
     WHERE Vlucht.brevetnr = Vlieger.brevetnr)
```

E15:

```
SELECT Vlieger.naam
FROM Vlieger
WHERE NOT EXISTS
     (SELECT Vliegtuig.callsign
     FROM Vliegtuig
     WHERE NOT EXISTS
        (SELECT Vlucht.brevetnr
        FROM Vlucht
        WHERE Vlucht.brevetnr = Vlieger.brevetnr))
```

E16:

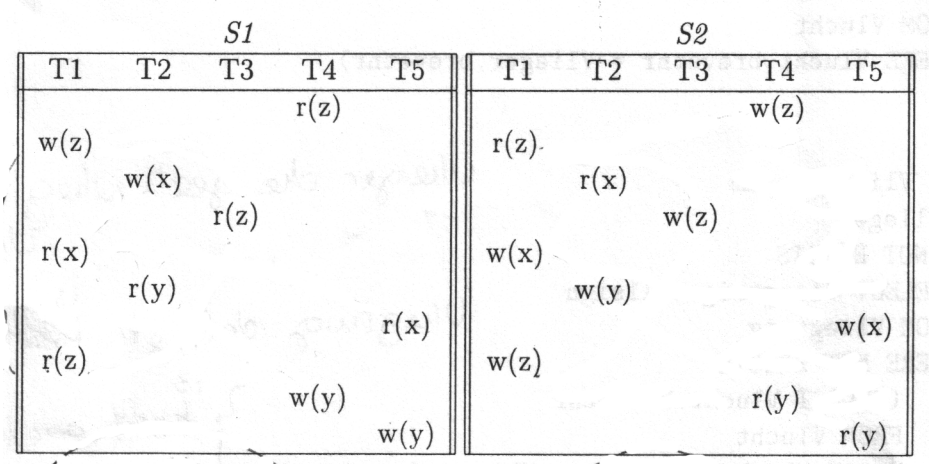
```
SELECT Vlieger.naam
FROM Vlieger
WHERE NOT EXISTS
     (SELECT Vliegtuig.callsign
     FROM Vliegtuig
     WHERE NOT EXISTS
        (SELECT Vlucht.brevetnr
        FROM Vlucht
        WHERE Vlucht.brevetnr = Vlieger.brevetnr
        AND Vlucht.callsign = Vliegtuig.callsign))
```

Opgave 5 Concurrency

Hieronder zijn twee schedules gegeven.

a) Stel voor elk van de schedules de complete precedentiegraaf op. Geef aan of deze schedules serializeerbaar zijn of niet. Licht toe. Geef zo mogelijk de equivalente seriële schedules.

b) Geef eveneens aan of de schedules getolereerd worden door een 2PL-Scheduler. Geef hierbij een korte toelichting.



Opgave 6 Query processing

We gaan uit van twee relatieschema's R en S die één attribuut A gemeenschappelijk hebben. We definiëren een algebraïsche operator ▶ (anti-join) als volgt:

$R \blacktriangleright S$ bevat de tuples r in R waarvoor geldt dat er geen tuple s in S bestaat met $r.A = s.A$.

a) Druk één van de queries $Q1 \ldots Q4$ van opgave 4 uit in de RA met behulp van de Q antijoin.

b) Hoe distribueert de selectie over de antijoin? (Gevalsonderscheid kan nodig zijn.)

c) Hoe distribueert de projectie over de antijoin?

d) Beschrijf hoe de hash-join aangepast zouden kunnen worden om een anti-join uit te rekenen.

Je hoeft bij b) en c) geen formele bewijzen te leveren.