Data & Informatie Toets 2: Databaseontwerp en SQL

Sportcentrum, 16 mei 2014, 15.45 – 17.30 uur

De toets bevat 3 vragen op 3 pagina's die gezamenlijk 100 punten (en 5 bonuspunten) opleveren.
Bij het tentamen mogen geen boeken, aantekeningen of electronische apparaten worden gebruikt.

Opgave 1 (20 punten)

Bij een webshop voor boeken spelen verscheidene entiteiten een rol: boek, winkelwagen, klant, magazijn, etc. We definiëren een relatie *R* die die entiteiten aan elkaar relateert, op de volgende manier. Een tuple (*B, A, W, K, N, V, M*) zit op tijdstip *t* in relatie *R* precies wanneer op tijdstip *t* al het volgende geldt:

1. *B* is een Boek-identificatie (bijvoorbeeld: een isbn-nummer).
2. *A* is het Aantal exemplaren van boek *B* dat in winkelwagen *W* zit.
3. *W* is een Winkelwagen.
4. *K* is een Klant die winkelwagen *W* gebruikt.
5. *N* is een naam van klant *K*.
6. *V* is de Voorraad (aantal exemplaren) van boek *B* in magazijn *M*.
7. *M* is een Magazijn.

Neem het volgende aan:

a) Een klant kan verscheidene winkelwagens tegelijk gebruiken.

b) Verschillende klanten gebruiken verschillende winkelwagens.

c) Een klant heeft precies één naam.

Geef voor ieder van de functionele afhankelijkheden hieronder, met een letter W of O aan of die Waar of Onwaar is in relatie *R*, en motiveer kort uw keuze met verwijzing naar 1...7 en a...c. (De motivatie telt even zwaar mee in de beoordeling als het antwoord W/O.)

Vraag a) $B\,W \rightarrow A$

Vraag b) $W\,A \rightarrow B$

Vraag c) $K \rightarrow N$

Vraag d) $K \rightarrow W$

Vraag e) $B\,M \rightarrow V$

Vraag f) $B \rightarrow V$

Vraag g) $N \rightarrow K$

Vraag h) $W \rightarrow K$

Vraag i) $B \twoheadrightarrow A\ W\ K$ *(Let op, dit is een MVD)*

Vraag j) $B \twoheadrightarrow V\ M$ *(Let op, dit is een MVD)*

Opgave 2 (30 punten)

Beschouw de tabel *R* met 5 attributen A, B, C, D, E en functionele afhankelijkheden $\mathcal{F}$, waarbij:

$$\mathcal{F} = \{AB \rightarrow D,\ CD \rightarrow A,\ B \rightarrow C,\ C \rightarrow B\}$$

Let op: *E* zit wel in *R* maar komt niet voor in $\mathcal{F}$.

Vraag a) Geef alle functionele afhankelijkheden in $\mathcal{F}$ die een schending vormen van de BCNF-conditie voor *R*. Beargumenteer uw antwoord.

Vraag b) Geef een lossless decompositie van *R* in precies **twee** schema's, zeg R_1 en R_2 , zodanig dat R_1 en R_2 samen minstens één schending minder hebben dan *R*. Verklaar uw werkwijze en geef precies aan wat de attributen en functionele afhankelijkheden van R_1 en R_2 zijn.

Vraag c) Construeer een lossless decompositie van *R* tot schema's die ieder in BCNF staan. (Gebruik maken van en verwijzen naar de vorige antwoorden is toegestaan.) Verklaar iedere stap kort maar zodanig dat het voor de corrector duidelijk is hoe u te werk gaat.

Vraag d) Zijn alle functionele afhankelijkheden uit $\mathcal{F}$, van het oorspronkelijke schema *R*, behouden onder bovenstaande decompositie van de oorspronkelijke *R*?

Opgave 3 (50 punten)

In de volgende opgavenserie wordt het volgende databaseschema gebruikt:

Class (<u>name</u>, type, country, guns, bore, displacement)

Ship (<u>name</u>, classname, launched)

Battle (<u>name</u>, date)

Outcome (<u>shipname, battlename</u>, result)

De attributen die tot de sleutel behoren zijn onderstreept. Verder geldt:

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*): bs (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 kanonloop (*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. De 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. Beantwoord de volgende vragen, waarbij DISTINCT alleen gebruikt wordt als het noodzakelijk is:

Vraag a) Formuleer in SQL de volgende vraag:

> *De naam van ieder schip van type bc (battle ship) dat voor '1700-01-01' tewater gelaten (launched) is.*

Vraag b) Formuleer in SQL met een group-by query:

> *Geef voor ieder land dat in minstens 10 zeeslagen betrokken was, de datums van de eerste en de laatste zeeslag waarin dat land betrokken was.*

Hierbij staat “land *x* is betrokken in zeeslag *y*” voor “er is een schip, van een klasse van land *x*, dat betrokken is in zeeslag *y*”. Veronderstel dat de waarden van het date-attribuut van zeeslagen met min en max geaggregeerd kunnen worden (tot een “eerste” en “laatste” datum).

Vraag c) Beschouw een vraag die het volgende oplevert:

> *Iedere zeeslag waarbij er een schip uit NL betrokken is.*

(Een schip komt uit land NL als voor de klasse van dat schip geldt: country = NL.) Iemand heeft het volgende begin gemaakt van de afleiding van de SQL query:

Stap 1
```
SELECT b.name FROM Battle b
WHERE 'er is een schip uit NL betrokken bij b';
```

Stap 2
```
SELECT b.name FROM Battle b
WHERE EXISTS (
     SELECT * FROM Ship s
     WHERE 's is uit NL'
     AND 's is betrokken bij b'
);
```

Geef voor deze vraag een afleiding in kleine stappen naar een SQL query die geen subqueries heeft en zo weinig mogelijk tabellen in de from clause. Het gegeven begin moet je gebruiken. Motiveer elke stap door te verwijzen naar één van de SQL herschrijfregels in de appendix.

Vraag d) Geef de SQL query voor de volgende vraag:

> *Geef iedere klasse waarvoor er een zeeslag is aan te wijzen zodanig dat **elk** schip van die klasse dat in die zeeslag zinkt, ook een zeeslag overleeft.*

(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.)

Appendix: SQL rewrite rules and definitions

One point rule:

```
EXITS(SELECT E FROM D WHERE D.a=3 AND D.b=2);
      <=>
3 IN (SELECT D.a FROM D WHERE D.b=2) ;
      <=>
2 IN (SELECT D.b FROM D where D.a=3) ;
```

Shunting rule

```
SELECT D1.a FROM D1 WHERE EXISTS (SELECT E FROM D2 WHERE P)
      <=>
SELECT DISTINCT D1.a FROM D1, D2 WHERE P
```

Foreign key rule: (given that in D1: FOREIGN KEY (did) REFERENCES D2(id))

```
SELECT D1.a FROM D1 WHERE EXISTS (SELECT * FROM D2 WHERE D2.id = D1.did) ;
      <=>
SELECT D1.a FROM D 1;
```

Join definition

```
SELECT E FROM D1, D2 WHERE P ;
      <=>
SELECT E FROM D1 JOIN D2 ON P;
```

Union definition

```
SELECT E FROM D WHERE P1 OR P2
      <=>
(SELECT E FROM D WHERE P1) UNION (SELECT E FROM D WHERE P2);
```

Intersect definition

```
SELECT E FROM D WHERE P1 AND P2 ;
      <=>
(SELECT E FROM D WHERE P1) INTERSECT (SELECT E FROM D WHERE P2) ;
```

Except definition

```
SELECT E FROM D WHERE P1 AND NOT P2 ;
      <=>
(SELECT E FROM D WHERE P1) EXCEPT (SELECT E FROM D WHERE P2) ;
```

Group by definition

```
SELECT mo.grouper,
      (SELECT SUM(col) FROM mytable mi WHERE mi.grouper = mo.grouper)
FROM  (SELECT DISTINCT grouper FROM mytable) mo ;
      <=>
SELECT mo.grouper, SUM(col) FROM mytable mo GROUP BY mo.grouper;
```