Data & Informatie voorbeeld van Toets 2

Databaseontwerp en SQL

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

Opgave 1 (20 punten)

De Winkel van Sinkel is een winkelketen met vestigingen op diverse locaties, één telefoonnummer per vestiging, verscheidene afdelingen per vestiging, producten, medewerkers, e-mailadressen per medewerker, en zo voorts. We definiëren een relatie *R* die deze entiteiten aan elkaar relateert. Een tuple (*V*, *T*, *A*, *P*, *M*, *E*) zit op tijdstip *t* in relatie *R* precies wanneer op tijdstip *t* al het volgende geldt:

1. *V* is een Vestiging van de winkelketen.
2. *T* is het Telefoonnummer van vestiging *V*.
3. *A* is een Afdeling van vestiging *V* (en niet van een andere vestiging).
4. *P* is een Product dat te koop is op –onder andere– afdeling *A*.
5. *M* is een Medewerker van afdeling *A*.
6. *E* is een E -mailadres van medewerker *M*.

Neem het volgende aan:

a) Verschillende vestigingen hebben verschillende telefoonnummers.
b) Een vestiging kan meerdere afdelingen hebben.
c) Een medewerker kan meerdere e-mailadressen hebben.
d) Medewerkers hebben geen gemeenschappelijke e-mailadressen.
e) Een medewerker behoort niet tot twee of meer afdelingen.

Geef voor ieder van de functionele afhankelijkheden (FD) en multi-valued dependencies (MVD) aan of die Waar of Onwaar is in relatie *R. Motiveer* uw keuze beknopt, zo mogelijk met verwijzing naar 1. .6 en a . . e. De motivatie telt even zwaar mee in de beoordeling als het antwoord.

Vraag a) $V \rightarrow T\,A\,P$

Vraag b) $T\,A\,P \rightarrow V$

Vraag c) $P \rightarrow M$

Vraag d) $M \rightarrow P$

Vraag e) $E \rightarrow T$

Vraag f) $T \rightarrow E$

Vraag g) $V \twoheadrightarrow T$

Vraag h) $A \twoheadrightarrow V\,T\,P$

Vraag i) $E \twoheadrightarrow M$

Opgave 2 (30 punten)

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

$$\mathcal{F} = \{AB \rightarrow CE,\ CD \rightarrow BF,\ E \rightarrow F,\ F \rightarrow E\}$$

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 heel precies aan wat de attributen en functionele afhankelijkheden van R_1 en R_2 zijn.

Vraag c) Zijn in de zojuist gegeven decompositie alle functionele afhankelijkheden behouden? Zo nee, geef dan een afhankelijkheid die niet is behouden:

Vraag d) 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.

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 dat gezonken is in de zeeslag van '1700-04-01'

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

Geef voor ieder land met de eigenschap dat de eerste tewaterlating ('launched') van een schip van dat land vóór 1600 is, het land zelf en het aantal zeeslagen waaraan schepen van dat land deelnemen.

Neem hierbij het volgende aan:

- De conditie "de tewaterlating van schip *s* is *vóór* 1600" luidt in SQL: "*s.launched < 1600*".
- "Het land van een schip" is "het land van de klasse van dat schip".
- Een land dat geen schepen heeft, hoeft niet opgeleverd te worden.

Vraag c) Een applicatie bevat de volgende query voor het vinden van de n*amen van iedere zeeslag waaraan schepen van verschillende klassen deelnemen. (Names of every battle with ships from different classes.)*

```
SELECT b.name FROM Battle b
WHERE EXISTS (
     SELECT * FROM Ship s1, Ship s2
     WHERE s1.classname <> s2.classname
     AND EXISTS(
          SELECT * FROM Outcome o1, Outcome o2
          WHERE o1.shipname = s1.name
          AND o2.shipname = s2.name
          AND o1.battlename = b.name
          AND o2.battlename = b.name
     )
);
```

Geef een afleiding in kleine stappen naar een SQL query die geen subqueries heeft en zo weinig mogelijk tabellen in de FROM clause. Motiveer elke stap door te verwijzen naar een van de SQL herschrijfregels in de appendix.

Vraag d) Beschouw een vraag die het volgende oplevert:

Ieder land waarvan ***elk*** schip in een zeeslag *vóór* 1600 is beschadigd.

Neem daarbij aan: "het land van een schip" is "het land van de klasse van dat schip", en: "datum < 1600" is een geoorloofde SQL conditie.

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;
```