Informationssysteme I
Übungsblatt 6

Christine Pries, Jonas Jacobi, Felix Oppermann (Gruppe 42)

15. Januar 2005

Aufgabe 1 - Funktionale Abhängigkeiten (50 Punkte)

a) Die Schlüsselkandiadaten in R bzgl. F sind: $\{A\}, \{B\}$, da jedes Attribut direkt oder indirekt von diesen abhängig ist. Aus dieser Abhängigkeit folgt die Eindeutigkeit von A und B, was sie zu Schlüsselkandidaten macht.

b)
- Wir wählen als Primärschlüssel A.
- Das Relationenschema ist in erster Normalform, da es nur atomare Einzelwerte gibt.
- Das Relationenschema ist in zweiter Normalform, da der Primärschlüssel nicht mehrere Attribute enthält.
- Das Relationenschema ist nicht in dritter Normalform, da es Nichtschlüsselattribute enthalt, die funktional von einem anderen Nichtschlüsselattribut abhängen (D ist von C abhängig).
- Das Relationenschema ist nicht in Boyce-Codd Normalform, da es Abhängigkeiten von anderen Attributen als dem Primärschlüssel gibt.

c)
1. $G = F = \{A \to \{AB\}, C \to \{CD\}, \{CD\} \to \{CA\}, B \to \{AC\}, A \to C\}$
2. $G = \{A \to A, A \to B, C \to D, \{CD\} \to C, \{CD\} \to A, B \to A, B \to C, A \to C\}$
3. $G = \{A \to A, A \to B, C \to D, C \to C, \{CD\} \to A, B \to A, B \to C, A \to C\}$
4. $G = \{A \to B, C \to D, \{CD\} \to A, B \to A, B \to C\}$

d)
1. $G = \{A \to B, C \to D, \{CD\} \to A, B \to A, B \to C\}$
2. $D_1 = \{\underline{A}, B\}, D_2 = \{\underline{C}, D\}, D_3 = \{\underline{B}, A\}, D_4 = \{\underline{B}, C\}, D_5 = \{\underline{CD}, A\}$
3. $D_1 = \{\underline{A}, B\}, D_2 = \{\underline{C}, D\}, D_3 = \{\underline{B}, A\}, D_4 = \{\underline{B}, C\}, D_5 = \{\underline{CD}, A\}$

e)
- $A \to \{AB\}$ wird verletzt durch $(A1) \to (A1, B1), (A1) \to (A1, B2)$.
- $\{CD\} \to \{CA\}$ wird verletzt durch $(C1, D1) \to (C1, A1), (C1, D1) \to (C1, A2)$.
- $B \to \{AC\}$ wird verletzt durch $(B1) \to (A1, C1), (B1) \to (A2, C1)$

Aufgabe 2 - SQL (50 Punkte)

a)
```
SELECT Country
    FROM is_member
    GROUP BY Country
    HAVING COUNT(Organization) > 65;
```

COUN
CDN
F
GB
I
NL
S
SF
USA

8 rows selected.

b)
```
SELECT Organization
    FROM is_member
    GROUP BY Organization
    HAVING COUNT(DISTINCT(Country)) =
        (SELECT COUNT(DISTINCT(Code)) AS Total FROM Country) ;
```

no rows selected

c)
```
SELECT COUNT(DISTINCT(Country))
    FROM is_member NATURAL JOIN
        (
        SELECT Organization
            FROM is_member
            WHERE Country = 'D'
        )
    WHERE Country <> 'D';
```

COUNT(DISTINCT(COUNTRY))
190

d)
```
CREATE VIEW european_country AS
    SELECT Country
        FROM encompasses
        WHERE Continent = 'Europe';

(
SELECT Country1 AS Country
    FROM borders
    WHERE Country1 NOT IN (SELECT Country FROM european_country)
    AND Country2 IN (SELECT Country FROM european_country)
)
UNION
(
SELECT Country2 AS Country
    FROM borders
    WHERE Country1 IN (SELECT Country FROM european_country)
    AND Country2 NOT IN (SELECT Country FROM european_country)
);

DROP VIEW european_country;
```

View created.

COUN
ARM
AZ
GE
IR
IRQ
KAZ
MNG
NOK
SYR
TJ

10 rows selected.

View dropped.

e)
```
CREATE VIEW Border_Length AS
    SELECT Country, SUM(Length) AS Length FROM
        (
            (
            SELECT Country1 AS Country, Length
                FROM borders
            )
            UNION
            (
            SELECT Country2 AS Country, Length
               FROM borders
            )
        )
    GROUP BY Country;

SELECT Country FROM
  Border_Length,
  (SELECT MAX(Length) AS Max_Length FROM Border_Length)
WHERE Length = MAX_Length;

DROP VIEW Border_Length;
```

View created.

COUN
TJ

View dropped.

f)
```
SELECT Code
    FROM
        (
        SELECT Code, (Population / Area) AS Population_Density
            FROM Country
        ),
        (
        SELECT MAX(Population / Area) AS Max_Density
            FROM Country
        )
    WHERE Population_Density = Max_Density;
```

CODE
MC

g)
```
CREATE VIEW african_country AS
    SELECT Country
        FROM encompasses
        WHERE Continent = 'Africa';

CREATE VIEW african_religion_member AS
    SELECT Religion.Name, SUM(Percentage * Population / 100) AS Members
    FROM african_country
        JOIN Country ON african_country.Country = Code
        JOIN Religion ON Code = Religion.Country
    GROUP BY Religion.Name;

CREATE VIEW african_population AS
    SELECT SUM(Population) AS African_Population
```

```
        FROM african_country
        JOIN Country ON Country = Code;

SELECT Name, (Members / African_Population * 100) AS Percentage
    FROM african_religion_member, african_population
    ORDER BY (Members / African_Population * 100);

DROP VIEW african_population;
DROP VIEW african_religion_member;
DROP VIEW african_country;
```

View created.

View created.

View created.

NAME	PERCENTAGE
Anglican	.000849047
Jewish	.020488097
Christian Jewish	.039925483
Hindu	.081119539
Lutheran	.114732315
Muslim Hindu	.300733965
Coptic Christian	.521864933
Kimbanguist	.636149328
Ethiopian Orthodox	2.7375914
Protestant	4.76715932
Roman Catholic	6.82889572
Sunni Muslim	9.08220619
Christian	12.0543483
Muslim	35.11744

14 rows selected.

View dropped.

View dropped.

View dropped.