

## BLATT 8

DANIEL SCHMIDT & PAMELA FLEISCHMANN

**Aufgabe 1.** connect to UEBUNG;

```
CREATE VIEW family (X,Y) AS
WITH MOTHERANCESTRY(Ancestor , Descendant) AS
  ((SELECT Mother , Child FROM MOTHERHOOD) UNION ALL
   (SELECT x.Mother , y.Descendant
    FROM MOTHERHOOD x, MOTHERANCESTRY y
    WHERE x.Child = y.Ancestor)),
  FATHERANCESTRY(Ancestor , Descendant) AS
  ((SELECT Father , Child FROM FATHERHOOD) UNION ALL
   (SELECT x.Father , y.Descendant
    FROM FATHERHOOD x, FATHERANCESTRY y
    WHERE x.Child = y.Ancestor)),
  ANCESTRY(X,Y) AS
  ((SELECT Ancestor , Descendant FROM MOTHERANCESTRY) UNION
   (SELECT Descendant , Ancestor FROM MOTHERANCESTRY) UNION
   (SELECT Ancestor , Descendant FROM FATHERANCESTRY) UNION
   (SELECT Descendant , Ancestor FROM FATHERANCESTRY)),
  COMMONANCESTRY(X,Y) AS
  (SELECT DISTINCT a.X,b.X FROM ANCESTRY a, ANCESTRY b
   WHERE a.Y = b.Y),
  COMMONANDMARRIAGE(X,Y) AS
  ((SELECT * FROM COMMONANCESTRY) UNION
   (SELECT MAN, WOMAN FROM married) UNION
   (SELECT WOMAN, MAN FROM married)),
  RELATED(X, Y) AS
  (SELECT DISTINCT a.X, b.X
   FROM COMMONANDMARRIAGE a, COMMONANDMARRIAGE b
   WHERE a.Y = b.Y)
  (SELECT * FROM RELATED);
```

terminate;

Die entstprechende Anfrage zu “Wieviele Verwandte hat Franziska?” ist `SELECT COUNT(Y) FROM family WHERE X='Franziska';` mit dem Resultat 28.

**Aufgabe 2.** ad. a

```

WITH REISEN(
    Abflugzeit ,
    Endposition ,
    Ankunftszeit ,
    Route ,
    Anzteilstr ,
    Gesamtkosten) AS
((SELECT
    depTime ,
    arrival ,
    arrTime ,
    cast(departure || '└─└' || fNo AS
        VARCHAR(60)) ,
    1 ,
    price
FROM flights WHERE departure = 'LBC') UNION
ALL
(SELECT
    depTime ,
    arrival ,
    arrTime ,
    cast(departure || '└─└' || tNo AS
        VARCHAR(60)) ,
    1 ,
    price
FROM rail WHERE departure = 'LBC') UNION ALL
(SELECT
    r.Abflugzeit ,
    f.arrival ,
    f.arrTime ,
    cast(r.Route || '→' || f.departure || '
        -' || f.fNo AS VARCHAR(60)) ,
    r.AnzTeilstr + 1 ,
    r.Gesamtkosten + f.price
FROM REISEN r, flights f WHERE
    r.Endposition = f.departure AND
    f.arrival <> 'LBC' AND
    f.departure <> 'ALC' AND
    r.Ankunftszeit < f.depTime AND
    r.Anzteilstr < 4) UNION ALL
(SELECT
    r.Abflugzeit ,
    t.arrival ,
    t.arrTime ,

```

```

        cast(r.Route || '>' || t.departure || '
        -' || t.tNo AS VARCHAR(60)),
        r.AnzTeilstre + 1,
        r.Gesamtkosten + t.price
FROM REISEN r, rail t WHERE
        r.Endposition = t.departure AND
        t.arrival <> 'LBC' AND
        t.departure <> 'ALC' AND
        r.Ankunftszeit < t.depTime AND
        r.Anzteilstre < 4)
)
SELECT DISTINCT Abflugzeit, Endposition,
        Ankunftszeit, Route, Anzteilstre,
        Gesamtkosten
FROM REISEN WHERE
        Endposition = 'ALC' AND
        Gesamtkosten = (SELECT min(Gesamtkosten) FROM
        REISEN WHERE Endposition = 'ALC') AND
        timestampdiff(8, Ankunftszeit - Abflugzeit) = (
        SELECT min(timestampdiff(8, Ankunftszeit -
        Abflugzeit))
        FROM REISEN WHERE
        Endposition = 'ALC' AND
        Gesamtkosten = (SELECT min(Gesamtkosten)
        FROM REISEN WHERE Endposition = 'ALC'))
;

```

ad. b

Um das Ergebnis zu beschleunigen muss die Gesamtmenge der Resultate in einem möglichst frühen Schritt eingeschränkt werden. Dies wäre möglich wenn man mehrere WITH statements miteinander kombiniert, sodass es mehrere Instanzen von Reise gibt, Reise-1, Reise-2, Reise-3 & Reise-4. Die Indizes würden hierbei genau darlegen wie viele Fahrten in einer Reise enthalten ist. Der Vorteil hierbei wäre, wie in der folgenden Abbildung beispielsweise in den Zeilen 37-40 zu sehen ist, dass man in den rekursiven Selects eine NOT EXISTS Abfrage auf eine günstigere und zugleich schnellere Verbindung abfragen kann, ohne einen Reference Error zu bekommen. Hierdurch lässt sich die Ergebnismenge früh beschränken und zu teure / langsame Lösungen werden nicht weiter verfolgt.

```

1 WITH REISEN_N(
2     Abflugzeit,
3     Endposition,
4     Ankunftszeit,

```

```

5      Route ,
6      Anzteilstr ,
7      Gesamtkosten) AS
8      ((SELECT
9          depTime ,
10         arrival ,
11         arrTime ,
12         cast(departure || '└─└' || fNo AS
13             VARCHAR(60)) ,
14         1 ,
15         price
16     FROM flights WHERE departure = 'LBC') UNION
17     ALL
18     (SELECT
19         depTime ,
20         arrival ,
21         arrTime ,
22         cast(departure || '└─└' || tNo AS
23             VARCHAR(60)) ,
24         1 ,
25         price
26     FROM rail WHERE departure = 'LBC') UNION ALL
27     (SELECT
28         r.Abflugzeit ,
29         f.arrival ,
30         f.arrTime ,
31         cast(r.Route || '→' || f.departure || '
32             ─' || f.fNo AS VARCHAR(60)) ,
33         r.AnzTeilstr + 1 ,
34         r.Gesamtkosten + f.price
35     FROM REISEN_N r, flights f WHERE
36         r.Endposition = f.departure AND
37         f.arrival < 'LBC' AND
38         f.departure < 'ALC' AND
39         r.Ankunftszeit < f.depTime AND
40         r.Anzteilstr < N AND
41         NOT EXISTS (SELECT * from REISEN_N-1 x
42             WHERE
43                 x.Ankunftszeit < f.arrTime AND
44                 x.Gesamtkosten < r.
45                     Gesamtkosten + f.price
46         )) UNION ALL
47     (SELECT
48         r.Abflugzeit ,
49         t.arrival ,

```

```

44         t.arrTime,
45         cast(r.Route || '→' || t.departure || '
           -' || t.tNo AS VARCHAR(60)),
46         r.AnzTeilstr + 1,
47         r.Gesamtkosten + t.price
48     FROM REISEN_N r, rail t WHERE
49         r.Endposition = t.departure AND
50         t.arrival <> 'LBC' AND
51         t.departure <> 'ALC' AND
52         r.Ankunftszeit < t.depTime AND
53         r.Anzteilstre < N AND
54         NOT EXISTS (SELECT * from REISEN_N-1 x
                       WHERE
55                         x.Ankunftszeit < f.arrTime AND
56                         x.Gesamtkosten < r.
                           Gesamtkosten + f.price
57                     ))
58 )
59 SELECT DISTINCT Abflugzeit, Endposition,
        Ankunftszeit, Route, Anzteilstre,
        Gesamtkosten
60 FROM REISEN_N WHERE
61     Endposition = 'ALC' AND
62     Gesamtkosten = (SELECT min(Gesamtkosten) FROM
                       REISEN_4 WHERE Endposition = 'ALC') AND
63     timestampdiff(8, Ankunftszeit-Abflugzeit) = (
        SELECT min(timestampdiff(8, Ankunftszeit-
        Abflugzeit))
64     FROM REISEN_4 WHERE
65     Endposition = 'ALC' AND
66     Gesamtkosten = (SELECT min(Gesamtkosten)
        FROM REISEN_N WHERE Endposition = 'ALC'
        ));

```

Da diese Lösung jedoch die Rekursion aus der Aufgabe nimmt gehe ich davon aus, dass man eine ähnliche Query auch rekursiv hinkommen könnte, allerdings habe ich das leider nicht geschafft. Aus diesem Grund habe ich mir auch den (sehr langen) Aufschrieb der kompletten Anfrage gespart, ich denke die Grundidee ist auch so klar geworden.

### Aufgabe 3.