Datenbankpraktikum 2020

Abgabe Aufgabe 2

Lukas Hempel & Thomas Pause
Matrikelnummern: 3739268 & 3720245
Bachelor Informatik
Betreuer: Martin Franke

Termin 2.Testat: 02. Juli 2020

2a) Sichtenerstellung

Die Freundschaftsbeziehung ist als gerichtete Beziehung gespeichert, um Anfragen bzgl. der Freundschaftsbeziehung komfortabel zu lösen sollen die Beziehungen ungerichtet gespeichert werden. Diesbezüglich sollen Sie eine Sicht „pkp_symmetric“ erstellen, die beide Richtungen enthält.

```
CREATE OR REPLACE VIEW pkp_symmetric AS
SELECT person_1_id as Person1Id, person_2_id as Freund1Id, person_2_id
   as Person2Id, person_1_id as Freund2Id, creationdate
FROM person_knows_person;
```

2b) Anfragen an die Datenbank

Formulieren Sie SQL-Anfragen, um folgende Fragen zu beantworten:

```
-- (1) In wie vielen verschiedenen afrikanischen Städten gibt es eine
   Universität?
SELECT COUNT(DISTINCT City.id) AS anzahl
FROM (University JOIN City ON University.city_id = City.id JOIN Country
    ON City.country_id = Country.id JOIN Continent ON Country.
   continent_id = Continent.id)
WHERE Continent.name = 'Africa';

-- Ergebnis: 100


-- (2) Wie viele Forenbeiträge (Posts) hat die älteste Person verfasst
   (Ausgabe: Name, #Forenbeiträge)?
SELECT person.firstname, person.lastname, COUNT(post.id) AS
   Forenbeitraege
FROM person LEFT JOIN post ON person.id = post.author_id
WHERE person.birthday = (SELECT MIN(birthday) FROM person)
GROUP BY person.firstname, person.lastname;

-- Ergebnis:
-- Joakim Larsson 0



-- (3) Wie viele Kommentare zu Posts gibt es aus jedem Land (Ausgabe
   aufsteigend sortiert nach Kommentaranzahl)? Die Liste soll auch Lä
   nder enthalten, für die keine Post-Kommentare existieren, d.h. die
   Kommentaranzahl = 0 ist! (Funktion Coalesce)
SELECT Country.name, COALESCE(COUNT(Comment.id), 0) AS NumberOfComments
FROM (Comment RIGHT JOIN Country on Comment.country_id = Country.id)
GROUP BY Country.name
ORDER BY 2 ASC;
```

```
-- Ergenis: 111 Zeilen,
-- Northern_Ireland                    |                  0
-- England                             |                  0
-- Scotland                            |                  0
-- Wales                               |                  0
-- Nepal                               |                  6
-- Singapore                           |                  8
-- Tanzania                            |                 11
-- Malta                               |                 11
-- Thailand                            |                 11
-- Mongolia                            |                 11
-- Hong_Kong                           |                 12
-- France                              |                 12
-- Mauritania                          |                 13
-- ...

-- WHERE Comment.reply_to_post_id IS NOT NULL -- Kills the 0 - Countrys


-- (4) Aus welchen Städten stammen die meisten Nutzer (Ausgabe Name +
   Einwohnerzahl)?
SELECT City.name, COUNT(Person.id) AS Einwohnerzahl
FROM Person RIGHT JOIN City ON Person.city_id = City.id
GROUP BY City.Name
ORDER BY 2 DESC;

-- Ergebnis: 1349 Zeilen (so viele wie es Cities gibt)
-- Ludwigsburg       |   2
-- Rahim_Yar_Khan    |   2
-- Chernivtsi        |   1
-- Nugegoda          |   1
-- Hefei             |   1
-- Tainan            |   1
-- Tlatelolco        |   1
-- Xi`an             |   1
-- Saltillo          |   1
-- Baishan           |   1
-- ...
```

```
-- (5) Mit wem ist Hans Johansson befreundet?
SELECT P2.id AS FreundID, P2.firstName AS freundVorname, P2.lastName AS
    freundNachname
FROM (person p2 LEFT JOIN pkp_symmetric ON p2.id = pkp_symmetric.
   freund1id)
WHERE pkp_symmetric.person1id = (
                                SELECT id
                                FROM person P1
                                WHERE (P1.firstName = 'Hans') AND (P1.
                                   lastName = 'Johansson')
                                );
-- Ergebnis: 9 Zeilen
-- 12094627905563 | Wojciech          | Ciesla
-- 12094627905628 | Abdoulaye Khouma | Dia
--  5497558138940 | Paul              | Becker
-- 12094627905550 | Hossein           | Forouhar
--  9895604650000 | Jan               | Zakrzewski
--  8796093022217 | Alim              | Guliyev
-- 10995116277764 | Bryn              | Davies
-- 12094627905567 | Otto              | Richter
--  7696581394474 | Ali               | Achiou


-- (6) Wer sind die echten Freundesfreunde von Hans Johansson?
-- Echte Freundesfreunde dürfen nicht gleichzeitig direkte Freunde von
   Hans Johansson sein.
-- Sortieren Sie die Ausgabe alphabetisch nach dem Nachnamen.

-- erst alle Freunde und deren Freunde von Hans Johansson und dann alle
    direkten (siehe 5) abziehen
SELECT P3.lastname, P3.firstname FROM person P3 JOIN(
SELECT fid1 FROM
(WITH RECURSIVE allFriends(fid1) AS (
  SELECT P2.id AS fid1
  FROM (person p2 LEFT JOIN pkp_symmetric ON p2.id = pkp_symmetric.
     freund1id)
  WHERE pkp_symmetric.person1id = (
                                  SELECT id
                                  FROM person P1
                                  WHERE (P1.firstName = 'Hans') AND (P1
                                     .lastName = 'Johansson')
                                  )
UNION ALL
SELECT pkp_symmetric.freund1id FROM allFriends JOIN pkp_symmetric ON
   allFriends.fid1=pkp_symmetric.person1id
)
SELECT DISTINCT(fid1) FROM allFriends) AS friendFriends
EXCEPT(
  SELECT P2.id
```

```
  FROM (person p2 LEFT JOIN pkp_symmetric ON p2.id = pkp_symmetric.
     freund1id)
  WHERE pkp_symmetric.person1id = (
                                  SELECT id
                                  FROM person P4
                                  WHERE (P4.firstName = 'Hans') AND (P4
                                     .lastName = 'Johansson')
                                  )
)) AS trueFriendFriends ON P3.id = trueFriendFriends.fid1
ORDER BY P3.lastname;

-- Ergebnnis: 26 Zeilen
-- Abouba Ali
-- Bazayev  Oleg
-- Bernal Pablo
-- Chen Amy
-- Diaz Roberto
-- ...


-- (7) Welche Nutzer sind Mitglied in allen Foren, in denen auch Mehmet
    Koksal Mitglied ist (Angabe Name)?
SELECT p4.id, p4.firstName, p4.lastName
FROM person p4
WHERE NOT EXISTS
(
        SELECT *
        FROM
        (
                SELECT Kreuzprodukt.personid as personid
                FROM    (
                        SELECT p2.id as personid, MehemtsForums.id AS
                           Mehmetsforumid
                        FROM Person p2,
                                (
                                SELECT f1.id
                                FROM (forum f1 JOIN
                                   forum_hasMember_person fhp1 ON f1.id
                                    = fhp1.forum_id JOIN person p1 ON
                                   fhp1.person_id = p1.id)
                                WHERE (p1.firstName = 'Mehmet') AND (p1
                                   .lastName = 'Koksal')
                                )
                                AS MehemtsForums
                        )
                        AS Kreuzprodukt
                WHERE NOT EXISTS
                        (
                        SELECT p3.id AS personid, f3.id AS forumid
```

```
                        FROM (forum f3 JOIN forum_hasMember_person fhp3
                            ON f3.id = fhp3.forum_id JOIN person p3 ON
                           fhp3.person_id = p3.id)
                        WHERE (Kreuzprodukt.personid = p3.id) AND (
                           Kreuzprodukt.mehmetsforumid = forum_id)
                        )
                ) AS KriterienTreffenNichtZu
        WHERE Kriterientreffennichtzu.personid = p4.id
);

-- Ergebnis: 4 Zeilen
-- 2199023255565 | Mehmet     | Koksal
-- 5497558138940 | Paul       | Becker
-- 2199023255611 | Chen       | Yang
-- 6597069766688 | Miguel     | Gonzalez

-- Wir sortieren also aus dem Kreuzprodukt alle Paare von (Person,
   Forum) die im Datensatz sind. Leute die in allen Foren von Mehmet
   Mitglieder sind, werden hierdurch aussortiert. Also sind diejenigen,
    die wir noch haben,
-- nicht Teil des Ergebnisses. Seien diese Leute nun Menge C. Dann
   ziehen wir einfach von allen Personen diese Menge ab und erhalten
   diejenigen, die in Mehmets Foren Mitglied sind!


-- (8) Geben Sie die prozentuale Verteilung der Nutzer bzgl. ihrer
   Herkunft aus verschiedenen Kontinenten an!
SELECT Continent.id, Continent.name, (COUNT(P2.id)/ (SELECT COUNT(P1.id
   ) FROM Person P1)::float)*100 AS percentage
FROM Continent JOIN Country ON Continent.id = Country.continent_id
     JOIN City ON Country.id = City.country_id JOIN Person P2 ON City.
        id = P2.city_id
GROUP BY Continent.id, Continent.name
ORDER BY percentage DESC

-- Ergebnis: 5 Zeilen
-- id   name  percentage
-- 1460 Asia  50
-- 1462 Europe 25
-- 1461 Africa  11.363636363636363
-- 1464 North_America 9.090909090909092
-- 1463 South_America 4.545454545454546
```

```
-- (9) Zu welchen Themen ('tag classes') gibt es die meisten Posts?
   Geben Sie die Namen der Top 10 'tag classes' mit ihrer Häufigkeit
   aus!
SELECT tagclass.name, tagclass.id, temp.anzahl
FROM (
        SELECT COUNT(pht1.post_id) AS anzahl, tht1.tagclass_id
        FROM post_hastag_tag pht1 JOIN tag_hastype_tagclass tht1 ON
           pht1.tag_id = tht1.tag_id
        GROUP BY tht1.tagclass_id
        ORDER BY anzahl DESC
        LIMIT 10
     ) AS temp JOIN tagclass ON temp.tagclass_id = tagclass.id
ORDER BY temp.anzahl DESC;

-- Ergebnis: 10 Zeilen
-- Person          | 211 |    110
-- MusicalArtist   | 115 |     99
-- OfficeHolder    | 349 |     76
-- Writer          |  88 |     66
-- TennisPlayer    |  59 |     63
-- BritishRoyalty | 336 |     57
-- Saint           | 193 |     33
-- Single          | 342 |     30
-- Philosopher     |  82 |     28
-- Album           | 182 |     27


-- (10) Welche Personen haben noch nie ein 'Like' für einen Kommentar
   oder Post bekommen? Sortieren Sie die Ausgabe alphabetisch nach dem
   Nachnamen.
SELECT P2.lastname, P2.firstname, P2.id
FROM  ( (SELECT P1.id
        FROM Person P1
        EXCEPT (
        SELECT Person_likes_comment.person_id
        FROM Person_likes_comment
      ))
        EXCEPT (
        SELECT Person_likes_post.person_id
        FROM Person_likes_post
      )) AS inglorious JOIN Person P2 ON inglorious.id = P2.id
ORDER BY P2.lastname


-- (11) Welche Foren enthalten mehr Posts als die durchschnittliche
   Anzahl von Posts in Foren (Ausgabe alphabetisch sortiert nach
   Forumtitel)?
-- zuerst die durchschnittliche Anzahl von Posts in Foren
```

```
SELECT AVG(ForumCounts.anzahl)
FROM    (
        SELECT COUNT(p1.id) AS anzahl
        FROM post p1 JOIN forum f1 ON p1.forum_id = f1.id
        GROUP BY f1.id
        ) AS ForumCounts;

-- dann alle die mehr enthalten
SELECT ForumswithCounts.title, ForumswithCounts.anzahl
FROM    (
        SELECT F2.title, COUNT(P2.id) AS anzahl
        FROM Forum F2 JOIN Post P2 ON F2.id = P2.forum_id
        GROUP BY F2.id
        ) AS ForumswithCounts
WHERE ForumswithCounts.anzahl > (
        SELECT AVG(ForumCounts.anzahl)
        FROM    (
                SELECT COUNT(p1.id) AS anzahl
                FROM post p1 JOIN forum f1 ON p1.forum_id = f1.id
                GROUP BY f1.id) AS ForumCounts
                )
        ORDER BY ForumswithCounts.title;

-- Ergebnis: 329 Zeilen
-- Album 0 of Abdul Haris Tobing            |     17
-- Album 0 of Alejandro Rodriguez           |     20
-- Album 0 of Ali Abouba                    |     13
-- Album 0 of Amy Chen                      |     19
-- Album 0 of Celso Oliveira                |     20
-- Album 0 of Djelaludin Zaland             |     15
-- Album 0 of Eric Mettacara                |     13
-- Album 0 of Fritz Engel                   |     13
-- Album 0 of Hao Li                        |     16
-- Album 0 of Jie Li                        |     11
-- Album 0 of John Johnson                  |     14
-- Album 0 of Jun Hu                        |     16
-- Album 0 of Jun Li                        |     18
-- Album 0 of Oleg Bazayev                  |     17
-- Album 0 of Otto Redl                     |     17
-- Album 0 of Wei Hu                        |     15
-- Album 1 of Adrian Bravo                  |     11
-- Album 1 of Alejandro Rodriguez           |     18
-- Album 1 of Aleksandr Dobrunov            |     19
-- ...

-- (12) Welche Personen sind mit der Person befreundet, die die meisten
    Likes auf einen Post bekommen hat? Sortieren Sie die Ausgabe
   alphabetisch nach dem Nachnamen.
-- zuerst die Person holen welche die meisten Likes bekommen hat
```

```
SELECT tempo.author_id
FROM (
        SELECT COUNT (p2.id) as anzahl, p2.author_id
        FROM post p2 JOIN person_likes_post plp2 ON p2.id = plp2.
           post_id
        GROUP BY p2.author_id
        ) AS tempo
WHERE tempo.anzahl = (
                      SELECT MAX(temp.anzahl)
                      FROM    (
                              SELECT COUNT(p1.id) AS anzahl, p1.
                                 author_id
                              FROM post p1 JOIN person_likes_post plp1
                                 ON p1.id = plp1.post_id
                              GROUP BY p1.author_id
                              ) AS temp
                      );

-- dann zugehörige Freunde finden
SELECT person.lastName, person.firstName, friends.person1id
FROM   (
      SELECT pkp1.person1id, pkp1.freund1id, pkp1.creationDate
      FROM pkp_symmetric pkp1
      WHERE pkp1.person1id =  (
            SELECT tempo.author_id
            FROM    (
                    SELECT COUNT (p2.id) as anzahl, p2.author_id
                    FROM post p2 JOIN person_likes_post plp2 ON p2.id =
                        plp2.post_id
                    GROUP BY p2.author_id
                    ) AS tempo
                      WHERE tempo.anzahl = (
                      SELECT MAX(temp.anzahl)
                      FROM    (
                               SELECT COUNT(p1.id) AS anzahl, p1.
                                  author_id
                               FROM post p1 JOIN person_likes_post plp1
                                   ON p1.id = plp1.post_id
                               GROUP BY p1.author_id
                              ) AS temp
                            )
                        )
        ) AS friends
JOIN person ON friends.freund1id = person.id
ORDER BY person.lastName;
```

```
-- Ergebnis: 9 Zeilen
-- Achiou    | Ali       | 2199023255611
-- Bravo     | Adrian    | 2199023255611
-- Chen      | Cheng     | 2199023255611
-- Davies    | Bryn      | 2199023255611
-- Li        | Jie       | 2199023255611
-- Liu       | Chong     | 2199023255611
-- Loan      | Cam       | 2199023255611
-- Oliveira  | Celso     | 2199023255611
-- Zhang     | Zhi       | 2199023255611


-- (13) Welche Personen sind direkt oder indirekt mit 'Jun Hu' (id 94)
   verbunden (befreundet)? Geben Sie für jede Person die Distanz zu Jun
    an.
SELECT p3.id, p3.firstname, p3.lastname, fidsAndDistances.distance FROM
        (WITH tempo AS
                (SELECT DISTINCT(friendFriends.fid1), friendFriends.
                   distance FROM
                        (WITH RECURSIVE allFriends AS (
                        SELECT P2.id AS fid1, 1 distance
                        FROM (person p2 LEFT JOIN pkp_symmetric ON p2.
                           id = pkp_symmetric.freund1id)
                        WHERE pkp_symmetric.person1id = (
                              SELECT id
                              FROM person P1
                              WHERE (P1.firstName = 'Jun') AND (P1.
                                 lastName = 'Hu')
                              )
                        UNION ALL
                        SELECT pkp_symmetric.freund1id, distance + 1
                           FROM allFriends JOIN pkp_symmetric ON
                           allFriends.fid1=pkp_symmetric.person1id
                        )
                        SELECT * FROM allFriends) AS friendFriends
                ORDER BY 1)
        SELECT *
                FROM tempo t1
                WHERE distance =
                        (SELECT MIN(distance)
                        FROM tempo t2
                        WHERE t1.fid1 = t2.fid1))
        AS fidsAndDistances JOIN Person p3 ON fidsAndDistances.fid1 =
           p3.id
ORDER BY 4;
```

```
--Ergebnis: 37 Zeilen
-- id            firstname  lastname  distance
-- 2199023255625 Cheng      Chen      1
-- 96            Anson      Chen      1
-- 8796093022217 Alim       Guliyev   1
-- 8796093022251 Chen       Li        1
-- 10995116277851Chong      Liu       1
-- ...


-- (14) Erweitern Sie die Anfrage zu Aufgabe 13 indem Sie zusätzlich
   zur Distanz den Pfad zwischen den Nutzern ausgeben.
WITH RECURSIVE allFriends AS (
      SELECT P2.id AS fid1, 1 distance, 'Jun Hu -> ' || p2.firstName ||
          ' ' || p2.lastName  friendPath
      FROM (person p2 LEFT JOIN pkp_symmetric ON p2.id = pkp_symmetric.
         freund1id)
      WHERE pkp_symmetric.person1id = (
                          SELECT id
                          FROM person P1
                          WHERE (P1.firstName = 'Jun') AND (P1.lastName
                              = 'Hu')
                          )
    UNION ALL
    SELECT pkp_symmetric.freund1id, distance + 1,
      (
      friendPath ||
      ' -> ' ||
      (SELECT firstName FROM Person p4 WHERE p4.id = pkp_symmetric.
         freund1id) ||
      ' ' ||
      (SELECT lastName FROM Person p4 WHERE p4.id = pkp_symmetric.
         freund1id)
              ) FROM allFriends JOIN pkp_symmetric ON allFriends.fid1
                 =pkp_symmetric.person1id
      )
SELECT p5.id, p5.firstName, p5.lastName, allFriends.distance,
   allFriends.friendPath FROM allFriends JOIN Person p5 on allfriends.
   fid1 = p5.id;

-- Ergebnis: 552 Zeilen (da manche Freunde über mehrere Wege erreichbar
    sind!)
--              96 | Anson   | Chen     |  1 | Jun Hu -> Anson Chen
--  2199023255625 | Cheng   | Chen     |  1 | Jun Hu -> Cheng Chen
--  8796093022251 | Chen    | Li       |  1 | Jun Hu -> Chen Li
--  8796093022217 | Alim    | Guliyev  |  1 | Jun Hu -> Alim Guliyev
-- 10995116277851 | Chong   | Liu      |  1 | Jun Hu -> Chong Liu
--  3298534883365 | Wei     | Wei      |  1 | Jun Hu -> Wei Wei
```

```
--  9895604649984 | Yang     | Li        |  2 | Jun Hu -> Cheng Chen ->
    Yang Li
--  8796093022217 | Alim     | Guliyev   |  2 | Jun Hu -> Cheng Chen ->
    Alim Guliyev
-- 15393162788888 | Jie      | Yang      |  2 | Jun Hu -> Cheng Chen ->
    Jie Yang
--  9895604650020 | Yang     | Liu       |  2 | Jun Hu -> Cheng Chen ->
    Yang Liu
-- 16492674416689 | Lin      | Zhang     |  2 | Jun Hu -> Cheng Chen ->
    Lin Zhang
--  7696581394520 | Chong    | Zhang     |  2 | Jun Hu -> Cheng Chen ->
    Chong Zhang
-- 10995116277851 | Chong    | Liu       |  2 | Jun Hu -> Cheng Chen ->
    Chong Liu
--  7696581394474 | Ali      | Achiou    |  2 | Jun Hu -> Anson Chen ->
    Ali Achiou
--  7696581394490 | Amy      | Chen      |  2 | Jun Hu -> Anson Chen ->
    Amy Chen
-- 10995116277764 | Bryn     | Davies    |  2 | Jun Hu -> Alim Guliyev ->
     Bryn Davies
-- 12094627905550 | Hossein  | Forouhar  |  2 | Jun Hu -> Alim Guliyev ->
     Hossein Forouhar
-- 13194139533342 | Joakim   | Larsson   |  2 | Jun Hu -> Alim Guliyev ->
     Joakim Larsson
-- 12094627905580 | Alexei   | Kahnovich |  2 | Jun Hu -> Alim Guliyev ->
     Alexei Kahnovich
--  8796093022253 | Ali      | Abouba    |  2 | Jun Hu -> Alim Guliyev ->
     Ali Abouba
-- 16492674416689 | Lin      | Zhang     |  2 | Jun Hu -> Alim Guliyev ->
     Lin Zhang
--  9895604650036 | Akira    | Yamamoto  |  2 | Jun Hu -> Alim Guliyev ->
     Akira Yamamoto
-- ...
```

2c) Änderungen in der erzeugten Datenbank

Es soll ein Mechanismus umgesetzt werden, um die Beendigung eines Arbeitsverhältnisses zu dokumentieren. Der entsprechende Eintrag in `person_workAt_company` soll mittels SQL-Anweisung gelöscht werden. Um die Datenmanipulation nachvollziehen zu können, soll der Löschvorgang in einer separaten Tabelle protokolliert werden. Dabei soll zusätzlich hinterlegt werden, wann das Arbeitsverhältnis beendet wurde (orientieren Sie sich am Löschzeitpunkt). Die Protokollierung soll automatisch erfolgen, wenn ein Mitarbeiter sein Arbeitsverhältnis bei einem Unternehmen beendet (Löschung in `person_workAt_company`).

```
-- first create the new table for the documentation of the delete
   process
CREATE TABLE IF NOT EXISTS deletedWorkers (
  deletedOn    timestamp NOT NULL,
  person_id    bigint    NOT NULL,
        company_id   bigint    NOT NULL
);

-- function that writes a timestamp and the row to delete into the new
   table
--
CREATE FUNCTION saveEndOfWork() RETURNS trigger AS $endOfWork$
BEGIN
  INSERT INTO deletedWorkers SELECT now(), OLD.person_id, OLD.
     company_id;
  RETURN OLD;
END;
$endOfWork$ LANGUAGE plpgsql;

CREATE TRIGGER endOfWork BEFORE DELETE
ON person_workat_company
   FOR EACH ROW
    EXECUTE PROCEDURE saveEndOfWork();
```