

05.12.2023

Übungsblatt 8 – Lösungsvorschlag

Diskussionsteil (im PS zu lösen; keine Abgabe nötig)

- a) ☐ ★ Diskutieren Sie folgenden vier Fragen mit Ihren Kolleg*innen:

- a) Was ist eine sogenannte Common Table Expression (CTE)?

Lösung



Ein Common Table Expression (CTE) ist eine temporäre benannte Ergebnismenge, die aus einer einfachen `SELECT`-Anweisung erstellt wird und in einer nachfolgenden `SELECT`-Anweisung verwendet werden kann. Jeder CTE ist wie eine benannte Abfrage, deren Ergebnis in einer virtuellen Tabelle, d. h. einer CTE, gespeichert wird, um später in der Hauptabfrage referenziert zu werden. Aus diesem Grund werden CTEs nicht zur späteren Verwendung gespeichert und können nur innerhalb der Abfrage, in der sie definiert wurden, referenziert werden. CTEs werden mit der `WITH`-Klausel erstellt.

- b) Wofür werden CTEs eingesetzt?

Lösung



CTEs erlauben es uns, komplexe SQL-Abfragen zu vereinfachen und hierarchische Daten abzufragen.

- c) Wie hängen CTEs mit rekursiven SQL-Abfragen zusammen?

Lösung




Rekursive SQL-Abfragen sind CTEs, die auf sich selbst verweisen.

- d) Was sind zwei (verschiedene) Anwendungsfälle für rekursive SQL-Abfragen?

Lösung



Zwei (verschiedene) Anwendungsfälle für rekursive SQL-Abfragen sind beispielsweise die Wegsuche und die Vorfahrenermittlung.

- b)  Öffnen Sie das webbasierte Übungstool für rekursive SQL-Abfragen *SQL-Recursion-Tool*¹ und wählen Sie die Datenbank „Flights simple“ aus.

- a) Führen Sie folgende rekursive SQL-Abfrage (vgl. „Example 1“) zur Bestimmung der Flughäfen, die von Wien aus erreicht werden können, schrittweise aus und erklären Sie Ihren Kolleg*innen dessen Funktionsweise.

```
1  /* This query calculates all airports that can be reached from Vienna. */
2  WITH RecRel(departure, destination) AS
3  (
4      SELECT departure, destination FROM flight WHERE departure = "VIE"
5      UNION ALL
6      SELECT step.departure, step.destination
7      FROM RecRel AS rec JOIN flight AS step ON (rec.destination = step.departure)
8  )
9  SELECT * FROM RecRel
```

- b) Führen Sie folgende rekursive SQL-Abfrage (vgl. „Example 2“) zur Bestimmung der Flughäfen, die von Wien aus erreicht werden können, sowie der die jeweiligen Flughäfen verbindenden Flugroute schrittweise aus und erklären Sie Ihren Kolleg*innen dessen Funktionsweise.

```
1  /* Calculates all airports reachable from Vienna, counts the steps and
   calculates the path taken. */
2  WITH RecRel(step, path, departure, destination) AS
3  (
4      SELECT 1, concat(departure," - ",destination),departure, destination FROM
           flight WHERE departure = "VIE"
5      UNION ALL
6      SELECT rec.step+1, concat(rec.path," - ",step.destination),step.departure,
           step.destination
7      FROM RecRel AS rec JOIN flight AS step ON (rec.destination = step.departure)
8  )
9  SELECT * FROM RecRel
```

¹<https://dbis-uibk.github.io/recursion>

Hausaufgabenteil (Zuhause zu lösen; Abgabe nötig)

In diesem Aufgabenblatt werden rekursive SQL-Abfragen behandelt. Dazu sei eine Datenbank zum Verwalten eines sozialen Netzwerks mit folgenden Tabellen gegeben:

```
person(id, firstname, lastname, year_of_birth, country_id)
friendship(person1_id, person2_id, friends_since)
follow(person_id, followed_person_id, follows_since)
```

Ein Eintrag in der friendship-Relation besagt, dass person1_id mit person2_id befreundet ist. Da Freundschaften bidirektional sind, ist immer nur ein Paar pro Freundschaft verzeichnet, d. h. das Tupel (x, y) gibt an, dass sowohl Person x mit Person y befreundet ist als auch Person y mit Person x . Beachten Sie bitte, dass die Tabelle friendship in diesem Fall dementsprechend keinen expliziten Eintrag (y, x) aufweist.

Ein Eintrag in der follow-Relation besagt, dass person_id, d. h. Follower, der Person followed_person_id, d. h. Followee, folgt. Im Gegensatz zu Freundschaften sind follow-Beziehungen unidirektional, d. h. das Tupel (x, y) gibt lediglich an, dass Person x Person y folgt.

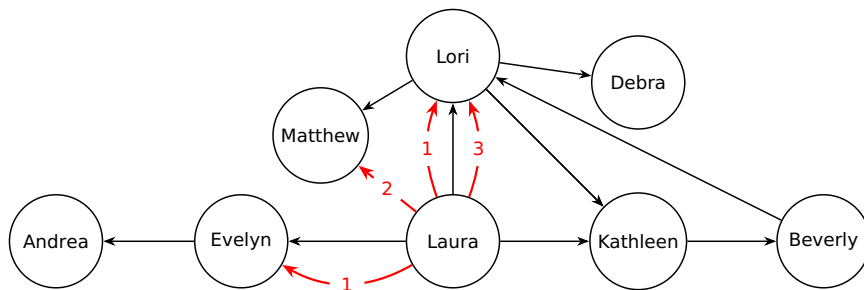


Abbildung 1: Exemplarischer follow-Graph

friendship- und follow-Beziehungen können wir anschaulich in einem Graph darstellen. Der Graph in Abbildung 1 zeigt ein Beispiel der follow-Beziehungen. In Abbildung 1 können wir zum Beispiel erkennen, dass Laura 3 Personen direkt folgt, d. h. Follower von Lori, Kathleen und Evelyn ist, die jeweils Followees von Laura sind. Indirekt – über follow-Beziehungen anderer Personen – folgt Laura über Lori beispielsweise Matthew.

Die Erreichbarkeit bzw. Nähe von anderen Knoten – in diesem Fall Personen – können wir über die Anzahl sogenannter Hops, d. h. Anzahl der Kanten des jeweiligen follow-Weges, in einem Graph bestimmen. Wie in Abbildung 1 für einige Beispiele in Rot gekennzeichnet, folgt Laura Evelyn über einen Hop, d. h. direkt, und Matthew über zwei Hops. Darüber hinaus folgt Laura Lori nicht nur über einen Hop, sondern auch über die 3 Hops Kathleen, Beverly und Lori.

Aufgabe 1 (Victor Halls Followees)

[4 Punkte]

Erstellen Sie eine neue Datenbank fakebook und laden Sie den Inhalt der Datei `fakebook.sql`^{OLAT} in diese.

Bestimmen Sie anschließend die Followees von Victor Hall, denen er maximal über 3 Hops folgt. Geben Sie dazu entsprechenden follow-Weg an, aus dem hervorgeht, wie seine jeweilige follow-Beziehung über maximal 3 Hops zustande kommt.

Reihenfolge und Bezeichnung der Ergebnisspalten:

- person_id (ID von Victor Halls)
- following (ID von Followee)

- path (follow-Weg als String im Format Victor Halls' ID->ID->...->ID->Followee's ID ohne Leerzeichen)
- hops (Anzahl der Hops als Integer)

Sortieren Sie das Resultat aufsteigend nach hops, wie folgende exemplarische Ausgabe zeigt:

person_id	following	path	hops
33	45	33->45	1
...
33	313	33->215->313	2
...
33	682	33->422->716->682	3

Tabelle 1: Exemplarisches Resultat der Aufgabe 1.

Hinweis



Beachten Sie bitte, dass in der follow-Relation Followee in der Spalte followed_person_id zu finden sind.

Hinweis



Sie können zum Verketteten von Zeichenketten die Funktion CONCAT^a verwenden.

^a<https://www.postgresql.org/docs/13/functions-string.html>

Hinweis



Bereiten Sie sich mithilfe einer Common Table Expression (CTE) die Ausgangsrelation so vor, dass sie für eine anschließende Rekursion geeignet ist. Beachten Sie darüber hinaus bitte, dass Sie nur mit der Verwendung des Schlüsselworts RECURSIVE eine WITH-Klausel auf ihre eigene Ausgabe verweisen lassen können.

Abgabe



1_query.sql

1_result.txt

Lösung



Query

```

1  WITH RECURSIVE RecFollowees(person_id, following, path, hops)
2  AS
3  (
4      SELECT      f.person_id,
5                  f.followed_person_id,
6                  CONCAT(f.person_id, '->', f.followed_person_id),

```

```

7          1
8      FROM      follow AS f
9      INNER JOIN person AS p ON f.person_id = p.id
10     WHERE      p.firstname = 'Victor'
11     AND         p.lastname = 'Hall'
12
13     UNION ALL
14
15     SELECT      t.person_id,
16                 follow.followed_person_id,
17                 CONCAT(path, '->', follow.followed_person_id),
18                 hops + 1
19     FROM        RecFollowees AS t
20     INNER JOIN follow ON t.following = follow.person_id
21     WHERE       t.hops < 3
22 )
23 SELECT *
24 FROM      RecFollowees
25 ORDER BY  hops ASC

```

Result

id	following	path	hops
57	15	57->15	1
57	33	57->33	1
57	786	57->786	1
57	81	57->15->81	2
57	328	57->15->328	2
57	641	57->15->641	2
57	157	57->33->157	2
...			
57	952	57->786->880->952	3

(94 rows)

Aufgabe 2 (Dazugewonnene Follower)

[6 Punkte]

Erstellen Sie – sollten Sie dies mit der vorherigen Aufgabe noch nicht gemacht haben – eine neue Datenbank fakebook und laden Sie den Inhalt der Datei `fakebookOLAT.sql` in diese.

- a) 4 Punkte Bestimmen Sie für alle im Jahre 1990 geborenen Personen, wie viele *unterschiedliche* Follower sie je Hop im Zeitraum zwischen 01.06.2015 und 01.06.2016 dazugewonnen haben.

Dabei gilt, dass

- indirekte Follower über Hops nur gezählt werden, wenn sie allen entsprechenden Personen in diesem Zeitraum gefolgt sind,
- maximal 5 Hops zu betrachten sind und
- für jede Person und jeden Hop die Anzahl der Follower ausgegeben werden soll.

Reihenfolge und Bezeichnung der Ergebnisspalten:

- `root_person_id` (ID von Followee)
- `firstname` (Vorname von Followee)
- `lastname` (Nachname von Followee)
- `hops` (Anzahl der Hops als Integer)
- `follower_count` (Anzahl der dazugewonnenen Follower als Integer)

Sortieren Sie die Ausgabe nach `root_person_id` und anschließend nach `hops`, wie folgende exemplarische Ausgabe zeigt:

<code>root_person_id</code>	<code>firstname</code>	<code>lastname</code>	<code>hops</code>	<code>follower_count</code>
11	Sharon	Moore	1	20
11	Sharon	Moore	2	31
11	Sharon	Moore	3	83
11	Sharon	Moore	4	172
11	Sharon	Moore	5	635
65	Mike	Patton	1	13
65	Mike	Patton	2	17
...
65	Mike	Patton	5	352
156	Maria	Lee	1	23
156	Maria	Lee	2	65
...

Tabelle 2: Exemplarisches Resultat der Aufgabe 2.

Hinweis



Beachten Sie bitte, dass in der `follow`-Relation Follower in der Spalte `person_id` zu finden sind.

Abgabe



`exercise2/a_query.sql`
`exercise2/a_result.txt`

Lösung



Query

```
1 WITH RECURSIVE RecFollower(root_person_id, follower_id, hops)
2 AS
3 (
```

```

4      SELECT      followed_person_id, person_id, 1
5      FROM        follow f
6      INNER JOIN  person p ON (f.followed_person_id = p.id)
7      WHERE       p.year_of_birth = 1990
8      AND         f.follows_since BETWEEN '2015-06-01' AND '2016-06-01'
9
10     UNION ALL
11
12     SELECT      r.root_person_id,
13                f.person_id,
14                r.hops + 1
15     FROM        RecFollower r
16     INNER JOIN  follow f ON (r.follower_id = f.followed_person_id)
17     WHERE       f.follows_since BETWEEN '2015-06-01' AND '2016-06-01'
18     AND         r.hops < 5
19 )
20 SELECT      r.root_person_id,
21            p.firstname,
22            p.lastname,
23            r.hops,
24            COUNT(DISTINCT r.follower_id) AS follower_count
25 FROM        RecFollower r
26 INNER JOIN  person p ON (r.root_person_id = p.id)
27 GROUP BY   r.root_person_id, p.firstname, p.lastname, r.hops
28 ORDER BY   r.root_person_id, hops

```

Result

root_person_id	firstname	lastname	hops	follower_count
2	Carol	Garcia	1	6
2	Carol	Garcia	2	17
2	Carol	Garcia	3	52
2	Carol	Garcia	4	148
2	Carol	Garcia	5	353
15	Pamela	Barnes	1	5
15	Pamela	Barnes	2	20
...				
15	Pamela	Barnes	5	368
65	Louis	Murphy	1	1
65	Louis	Murphy	2	4
...				
995	Jose	Mitchell	1	2
995	Jose	Mitchell	2	9
995	Jose	Mitchell	3	20

995	Jose	Mitchell	4	51	
995	Jose	Mitchell	5	124	

(165 rows)

- b) **2 Punkte** Erweitern Sie die von Ihnen in der vorherigen Unteraufgabe formulierte SQL-Abfrage, um nur die Ergebnisse von Personen zu bestimmen, die mit dem letzten betrachteten, 5. Hop mehr als 345 Follower dazugewonnen haben.

Die Reihenfolge und Bezeichnung der Ergebnisspalten entspricht dabei der in der vorherigen Unteraufgabe angegebenen, sodass ein exemplarisches Resultat mit dem bereits in Tabelle 2 angeführten übereinstimmt.

Hinweis



Bedenken Sie bitte, dass Sie – analog zur vorherigen Unteraufgabe – die Anzahl der im Zeitraum zwischen 01.06.2015 und 01.06.2016 dazugewonnen unterschiedlichen Follower je Hop bestimmen sollen. Erweitern Sie dementsprechend die über die Rekursion bereits bestehende CTE um das Ergebnis der vorherigen Unteraufgabe, anstatt lediglich auf eine HAVING-Klausel zurückzugreifen.

Abgabe



exercise2/b_query.sql
 exercise2/b_result.txt

Lösung



Query

```

1  WITH RECURSIVE RecFollower(root_person_id, person_id, follower_id, steps)
2  AS
3  (
4      SELECT      followed_person_id, followed_person_id, person_id, 1
5      FROM        follow f
6      INNER JOIN  person p ON (f.followed_person_id = p.id)
7      WHERE       p.year_of_birth = 1990
8      AND         f.follows_since BETWEEN '2015-06-01' AND '2016-06-01'
9
10     UNION ALL
11
12     SELECT      r.root_person_id,
13                f.followed_person_id,
14                f.person_id,
15                r.steps + 1
16     FROM        RecFollower r

```



```

17     INNER JOIN follow f ON (r.follower_id = f.followed_person_id)
18     WHERE      f.follows_since BETWEEN '2015-06-01' AND '2016-06-01'
19     AND        r.steps < 5
20 ),
21 RecFollowerResult AS
22 (
23     SELECT      r.root_person_id, p.firstname, p.lastname, r.steps,
24     COUNT      (DISTINCT r.follower_id) AS follower_count
25     FROM        RecFollower r
26     JOIN        person p ON (r.root_person_id = p.id)
27     GROUP BY    r.root_person_id, p.firstname, p.lastname, r.steps
28     ORDER BY    r.root_person_id, steps
29 )
30 SELECT *
31 FROM    RecFollowerResult
32 WHERE    root_person_id IN
33 (
34     SELECT root_person_id
35     FROM    RecFollowerResult
36     WHERE    steps = 5 AND follower_count > 345
37 )

```

Result

root_person_id	firstname	lastname	steps	follower_count
2	Carol	Garcia	1	6
2	Carol	Garcia	2	17
2	Carol	Garcia	3	52
2	Carol	Garcia	4	148
2	Carol	Garcia	5	353
15	Pamela	Barnes	1	5
15	Pamela	Barnes	2	20
15	Pamela	Barnes	3	55
15	Pamela	Barnes	4	162
15	Pamela	Barnes	5	368
815	Deborah	Brown	1	10
815	Deborah	Brown	2	37
815	Deborah	Brown	3	94
815	Deborah	Brown	4	240
815	Deborah	Brown	5	496

(15 rows)

Ein einfacher Rückgriff auf `HAVING steps = 5 AND follower_count > 345` funktioniert nicht, da ausschließlich diese Zeilen angezeigt werden würden. Da aber alle Zwischenschritte angezeigt werden sollen, wird das Ergebnis als weitere CTE definiert, worauf am Ende einfach selektiert werden kann.

Wichtig: Laden Sie bitte Ihre Lösung in OLAT hoch und geben Sie mittels der Ankreuzliste auch unbedingt an, welche Aufgaben Sie gelöst haben. Die Deadline dafür läuft am Vortag des Proseminars um 16:00 ab.