TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Dr. Thomas Neumann

Exercises for *Foundations in Data Engineering*, WiSe 23/24

Alexander Beischl, Maximilian Reif (i3fde@in.tum.de)

http://db.in.tum.de/teaching/ws2324/foundationsde

Sheet Nr. 07

Exercise 1 In the last lectures you have learned about window functions. Now, we want to solve sheet 5's `exercise 3.2` & `3.3` using *window functions*. Again, use this WebInterface for the task. By clicking on the button `UniSchema` you can see the different relations. Use the expanded `examination` relation and the solution from `task 3.1`:

```
WITH examination(MatrNr,CourseNr,PersNr,Grade) as (
    SELECT * FROM pruefen
    UNION
    VALUES (29120,0,0,3.0), (29555,0,0,2.0),
           (29555,0,0,1.3), (29555,0,0,1.0)
),
grades(Name,MatrNr,Semester,Grade) as (
    SELECT s.name, s.matrnr, semester, avg(Grade)
    FROM studenten s, examination p
    WHERE s.matrnr = p.matrnr
    GROUP BY s.name, s.matrnr, semester
)
```

1. Based on the individual average grade, determine each student's rank within the student's cohort (students in the same semester) using *window functions*.
2. Additionally, for each student calculate the difference between the student's average grade and the cohort's average using *window functions*. (The cohort's average is the average of individual averages.)

Exercise 2 Window Functions in SQL

Analyze the salaries of all professors using *window functions* on the university schema. You may test your queries in the WebInterface. Use the example relation *Professors*:

```
WITH Professors (persnr, name, paygrade, room, salary, taxclass) as
(
  VALUES (2125,'Sokrates','C4',226,85000,1),
         (2126,'Russel','C4',232,80000,3),
         (2127,'Kopernikus','C3',310,65000,5),
         (2128,'Aristoteles','C4',250,85000,1),
         (2133,'Popper','C3',52,68000,1),
         (2134,'Augustinus','C3',309,55000,5),
         (2136,'Curie','C4',36,95000,3),
         (2137,'Kant','C4',7,98000,1)
)
```

1. Assign a rank to each professor, depending on their salary. Professors with the same salary should receive the same rank.

2. Assign a rank within their pay grade to each professor, depending on their salary. Professors with the same salary should receive the same rank.
3. Calculate the running sum for each professor so that each professor's sum includes the salaries of those earning less or equal than they within their pay grade.
4. Calculate the running average of each professor and the 2 above and below sorted by salary and partitioned by grade.
5. Calculate the running average of each professor and the professors within a range of 5000€ more and less sorted by salary and partitioned by grade.
6. For each professor return the name of the one professor before and after him/her in the salary ranking. Professors with equal salary should be ordered by their name.
7. Calculate the top 3 with and without window functions.

Exercise 3 The Hiking Quest

Charlie is quite fond of the idea of completing challenges. One of her current adventures includes completing very long hiking trails. Being a very organized person, but always short on spare time, she divides the trails into day-hikes. Whenever the sun is shining and time allows, she sets of to complete a section. On the Munich-Venice trail, she already completed section 1-3, 8-9 and 22-23. Furthermore, she completed some sections on a mediterranean island.

Her achievements can be given to a database like this:

```
WITH trails (id, leg) as (
   SELECT 1, leg FROM generate_series(1,28) leg
  UNION ALL
   SELECT 2, leg FROM generate_series(1,15) leg ),
completed (trail_id, leg) as (
   VALUES (1,1),(1,2),(1,3),(1,8),(1,9),(1,22),(1,23),
          (2,1), (2,2), (2,11), (2,12)
)
SELECT * FROM completed;
```

Support her cause by providing some SQL-queries. (Hint: Use window functions)

1. Create some motivating statistics:
   a) How many sections did she complete? (A section is a series of consecutive day-hikes)
   b) What is the average, minimum, maximum length of all completed sections?
2. Help her plan the next trip: A long weekend is coming up, so Charlie can spend 3 days hiking. List uncompleted sections that consist of 3 or more day-hikes.

Exercise 4 For the graph in Figure 1, state SQL queries that answer these questions. You can use the following `with-statements` to query the graph in SQL:

```
WITH recursive singleDirection (a,b) as (
   SELECT *
   FROM (VALUES (1,2), (2,4), (1,3), (3,4), (2,5), (5,6),
                (4,6)) as graph ),
undirectedGraph as (
   SELECT *
```

```
 FROM singleDirection
UNION ALL
 SELECT b, a
 FROM singleDirection )
```

1. Is 6 reachable from 1?
2. How long is the shortest path from 1 to 6? (To end recursion, use this information: The diameter of the graph is 4.)

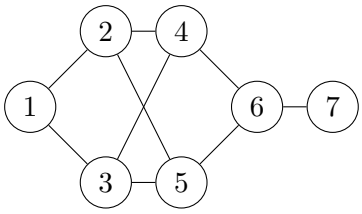
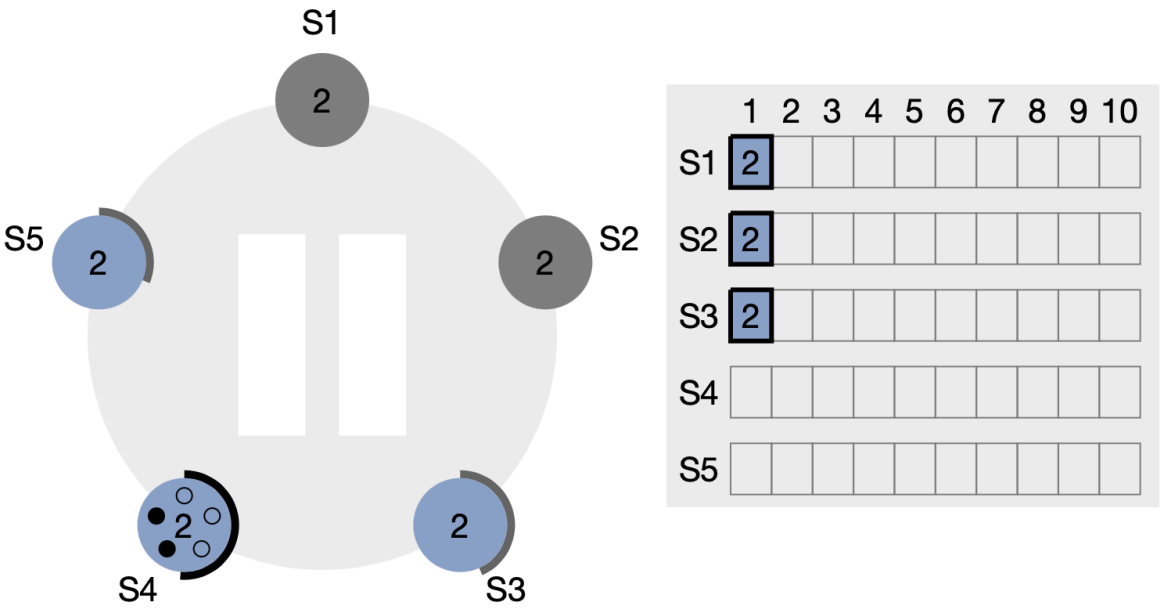


Figure 1: Example graph

Exercise 5 The Raft Consensus Protocol

To get a feeling for the *Raft consensus protocol*, you can play with the *RaftScope* at https://raft.github.io/. You can trigger events by right-clicking on servers.

1. Using the Raft visualization, simulate the following events:
   a) Wait for the first leader election.
   b) Send a request to the leader. What do you observe?
   c) Stop the leader and newly elected leaders until only two servers are left. What happens?
   d) How many servers have to fail to kill a cluster of n servers?
2. How can we be sure that a newly elected leader is holding the newest committed log entry?


S1
2
S5
2
S2
2
2
S4
2
S3
1 2 3 4 5 6 7 8 9 10
S1 2
S2 2
S3 2
S4
S5