Institut für Informatik
Datenbanken und Informationssysteme
Prof. Dr. Stefan Conrad
Thomas Germer

WiSe 24/25
Assignment 7
Due December 3, 8:00 AM

Relational Databases and Data Analysis

- As usual, use `zipme.py` to create the archive to upload in ILIAS.
- If you are one of the few students that have not read assignment 0 yet, read it now.

Exercise 1 *Query optimization* (2 + 2 Points)

In lecture 4, you have seen that performance can vary drastically depending on join order, even if the final query result is the same. You are given two query plans, `queryplan1` and `queryplan2` in `test_exercise_1_a.py`, which correspond to the query plans on slides 15 and 16 in lecture 4. To measure how well an execution plan performs, we will sum up the number of rows of all intermediate relations after each join operation.

(a) To demonstrate the difference in performance, choose values for the relations in file `exercise_1_a.py` such that the first query plan is 100 times worse than the second query plan, i.e. the sum of numbers of rows of intermediate relations is 100 times larger.

(b) There usually is no perfect query plan. Instead, the best query plan can only be determined with respect to specific data. Show that the performance of the two query plans can also be the opposite, i.e. define relations in `exercise_1_b.py` such that the second query plan is 100 times worse than the first query plan.

Exercise 2 *OLAP Operators* (1 + 1 + 2 + 1 + 1 Points)

This exercise uses the same data as exercise 2 from assignment 6. Start a PostgreSQL server and run `test_populate_database.py` before running the tests.

Computation of subtotals and totals is a common operation. The naive approach of `UNION`-ing the result of multiple subqueries for each combination of attributes is a lot of work to write. Therefore, specialized operators have been developed.

(a) With your knowledge from the lecture, you should be able to realize that the following query to count total and subtotal sales does not make sense. Fix it.

```sql
SELECT
    category, subcategory, year, COUNT(*) AS n
FROM
    sale
    JOIN product ON sale.product_id = product.product_id
    JOIN timestamp ON sale.timestamp_id = timestamp.timestamp_id
GROUP BY
    CUBE(category, subcategory, year);
```

(b) Rewrite your solution from the previous exercise using `GROUPING SETS`.

(c) Rewrite your solution from the previous exercise using `UNION ALL`. No string formatting!

(d) Imagine a cube similar to lecture 3, page 9. The time dimension has the hierarchy

hour $\rightarrow$ day $\rightarrow$ month $\rightarrow$ quarter $\rightarrow$ year.

The current granularity of the time dimension is *month*, so the cube has 12 subcubes along the time dimension. Now, two *Drill-Down* operations, four *Roll-Up* operations, then three *Drill-Down* operations and finally two *Roll-Up* operations are performed on the cube in succession along its time dimension. After those operations, how many subcubes does the cube have along its time dimension? Write that number `exercise_2_d.txt`. Assume that a second has 1000 milliseconds, a minute has 60 seconds, an hour has 60 minutes, a day has 24 hours, a month has 31 days, a quarter has 3 months, a year has 4 quarters and there are 55 years in the database.

(e) In `exercise_2_e.txt`, you will find four beautiful ASCII cubes. Demonstrate your understanding of the slice operation by marking a different slice on each cube by placing an alphabetic character of your choice on any face of the corresponding visible subcubes.