Task 3

Datalagring, IV1351

Magnus Kristoffer Berg
mkberg@ug.kth.se
2022-12-29

Contents

1 Introduction

The purpose of this task was to create OLAP, queries and views. The task was to create three queries that were to be executed manually and one query to be performed programmatically.

2 Method

For this task PostgreSQL was used with pgAdmin 4. To create SQL queries pgAdmin 4 and Visual Studio Code was used. To verify that the queries worked they were tested in pgAdmin 4 and there they could easily be rewritten and tested again and also some of the data had to be changed/rewritten from task 2 to fit in with the task description. To test the queries, they were tested and checked against the data to see if it matched.

3 Result

https://github.com/Mangelol/SoundGood

```
SELECT count(booked_lesson_id), extract(month from date)
from booked_lesson
WHERE extract(year from date) = 2022
group by month
```

Figure 3.1.1 Query booked lessons by month

	count bigint	month numeric
1	1	11
2	12	12

Figure 3.1.2 Result booked lessons by month

```
SELECT count(ensemble_id) as ensembles, extract(month from date) as month
from ensemble, booked_lesson
where ensemble.booked_lesson_id = booked_lesson.booked_lesson_id
AND extract(year from date) = 2022
group by month
```

Figure 3.1.3 Query ensembles per month

	ensembles bigint	month numeric
1	1	12

Figure 3.1.4 Result ensembles per month

```
SELECT count(group_lesson_id) as group_lessons, extract(month from date) as month
from group_lesson, booked_lesson
where group_lesson.booked_lesson_id = booked_lesson.booked_lesson_id
AND extract(year from date) = 2022
group by month
```

Figure 3.1.5 Query group lessons per month

	group_lessons bigint	month numeric
1	1	12

Figure 3.1.6 Result group lessons per month

```
SELECT count(individual_lesson_id) as individual_lessons, extract(month from date)
from individual_lesson, booked_lesson
where individual_lesson.booked_lesson_id = booked_lesson.booked_lesson_id
AND extract(year from date) = 2022
group by month
```

Figure 3.1.7 Query individual lessons per month

	individual_lessons bigint	month numeric
1	1	12

Figure 3.1.8 Result individual lessons per month

The first query shows all booked lessons per month. The rest of the queries are by ensembles, group lessons and individual lessons and they are very similar to each other, so the last query will be explained. With “SELECT” we specify two columns to be included in the result, the count of rows from individual lesson which is calculated using count(individual_lesson_id) and the month is extracted from the date column. With “from” we select the two tables to be included in the query. “where” specifies that individual lesson must be equal to booked lesson id in the booked_lesson table. The year is extracted from the date column. Then we group the lessons by month.

```
SELECT COUNT(DISTINCT s.student_id) as num_students,
       s.sibling_count as num_siblings
FROM (
  SELECT student.student_id as student_id, COUNT(sibling.student_id_2) as sibling_count
  FROM student
  INNER JOIN sibling
  ON student.student_id = sibling.student_id_1
  WHERE sibling.student_id_2 IS NOT NULL
  GROUP BY student.student_id
  UNION ALL
  SELECT student.student_id as student_id, COUNT(sibling.student_id_2) as sibling_count
  FROM student
  INNER JOIN sibling
  ON student.student_id = sibling.student_id_2
  WHERE sibling.student_id_2 IS NOT NULL
  GROUP BY student.student_id
  UNION ALL
  SELECT student.student_id as student_id, 0 as sibling_count
  FROM student
  INNER JOIN sibling
  ON student.student_id = sibling.student_id_1
  WHERE sibling.student_id_2 IS NULL
) s
GROUP BY s.sibling_count
ORDER BY s.sibling_count ASC;
```

Figure 3.2.1 Query to show amount of students with 0, 1, 2 or more siblings

	num_students bigint	num_siblings bigint
1	1	0
2	8	1

Figure 3.2.2 Result that shows amount of students with 0, 1, 2 or more siblings

As can be seen in the result 1 student had no siblings and 8 students had 1 sibling, no student had 2 or more siblings. The query counts the number of students with each number of siblings. The first “SELECT” statement selects all rows from the student and siblings tables where a student has a sibling and counts the number of siblings for each student. The second “SELECT” statement is like the first one but uses the “student_id_2” column instead of “student_id_1” to join the tables. The third “SELECT” statement selects all rows from the student and sibling tables where there is no sibling for each student.

```
SELECT count(booked_lesson_id), booked_lesson.instructor_id
from booked_lesson, instructor
where instructor.instructor_id = booked_lesson.instructor_id
AND extract(month from date) = extract(month from NOW())
group by booked_lesson.instructor_id
having count(booked_lesson_id) > 8
```

Figure 3.3.1 Query for instructors who has given more than a specific amount of lessons

	count bigint	instructor_id integer
1	12	1

Figure 3.3.2 Result showing the amount of instructors with more than 8 lessons and how many lessons

To get all the lessons from the current month we use the “NOW()” function. It is used together with the comparison of instructor_id. To get instructors that have given more than a specific number of lessons, in this case 8, the term “having” is used that check for conditions on count. Only one instructor has given more than 8 lessons so then it displays that 1 instructor has given 12 lessons.

```
SELECT e.ensemble_id, e.date, e.genre,
       CASE
         WHEN e.current_enrollments = e.maximum_enrollments THEN 'full booked'
         WHEN e.current_enrollments + 2 >= e.maximum_enrollments THEN '1-2 seats left'
         ELSE 'more seats left'
       END as seats_status
FROM ensemble e
WHERE e.date BETWEEN (CURRENT_TIMESTAMP + INTERVAL '1' DAY) AND (CURRENT_TIMESTAMP + INTERVAL '7' DAY)
ORDER BY e.genre, e.date;
```

Table 3.4.1 Query for ensembles held next week

	ensemble_id [PK] integer	date timestamp without time zone	genre character varying (50)	seats_status text
1	3	2023-01-04 10:00:00	jazz	full booked
2	2	2023-01-02 10:00:00	rock	1-2 seats left

Table 3.4.2 Result showing the ensembles held next week

The query above gets the ensembles held next week, the date, genre and if they are fully booked or have seats left. The query selects all rows from the ensemble table that have a date value between the current date + 1 day to include ensembles held tomorrow and the current time plus seven days to include the ensembles held within the next week. The “CASE” statement is used to determine the available seats for each ensemble based on the number of current enrollments and the maximum number of enrollments. If the number of current enrollments is equal to the maximum number of enrollments, the seat status is set to fully booked. If the number of current enrollments is greater than or equal to the maximum number of enrollments minus 2, the seat status is set to 1-2 seats left. Else the seat status is set to “more seats left”.

4 Discussion

A view has many upsides to them, especially because it does not take up any extra space. A view was not created for any of these queries simply because the need for them was not felt. However for more complex queries in the future using views will be in the back of my head. A materialized view however does take up space because the result is stored in the database. The reason I did not use any of these in the task was because a materialized view needs to be updated in order to get the latest data. It can be discussed if using a materialized view would have been effective when the data had come from earlier dates for instance for the instructors that held a lot of lessons. The database design was not changed in order to simplify these queries, however some parts of the database had to be redone because the data that some of the queries wanted to have was insufficient and needed to be added in before the queries, for instance date of ensembles. Some other tables also needed to get more data in order to provide reasonable amounts of data for the queries.