Project Report 3

Data Storage Paradigms, IV1351

David Holmertz

04/12/2023

1 Introduction

I uppgift 3 för projektet ligger fokus på den praktiska tillämpningen av SQL inom databashantering och analys. Denna uppgift innefattar skapandet av OLAP-frågor som är väsentliga för affärsanalys och rapportgenerering för databassystemet.

För denna uppgift har jag arbetat med Robert Koivusalo.

2 Literature Study

Den huvudsakliga informationen för SQL och var från SQL-föreläsningen. Föreläsningen gav en grundlig inblick i SQL:s roll. Jag använde även ”Tips-and-Tricks-Task3.pdf”, som gav råd och insikter för att skriva SQL-queries och hantera typiska problem som kan uppstå i SQL. Kombinationen av dessa två källor har gett mig en grund för att inte bara utveckla SQL-frågor för projektet men också för att förstå de underliggande principerna som gör dessa queries praktiska i databaser.

3 Method

Under arbetet med Task 3 utvecklade jag min förståelse för skapandet av en databas i SQL och hur queries designas. För att köra olika queries och se tabeller användes pgAdmin och PostgreSQL. Koden skrevs i Visual Studio Code.
Målet var att skapa en vy för varje fråga. En vy resulterar i en logisk vy som ser ut som en vanlig tabell. Denna används för att verifiera att kommandot utför rätt åtgärd på datan. Till exempel, för att verifiera att det första kommandot ”antalet lektioner per månad under ett specifikt år” fungerar, behövdes en manuell kontroll av antalet lektioner under några månader. Detta jämfördes sedan med vyn för att säkerställa att båda metoderna ger samma data och bekräftar således frågans giltighet.

De frågor som skulle kollas upp var:

- Antal lektioner per månad under ett specifikt år.
- Antal olika typer av lektioner per månad.
- Genomsnittligt antal lektioner varje månad.
- Räkna antalet lektioner som olika instruktörer har gett.
- Lista ensembler som kommer att hållas nästa vecka och markera dem om de har 1-2 platser kvar, eller fler platser kvar.

4 Result

Den databasen som skapades är baserad på den tidigare logiska modellen i task 2. Modellen inkluderar det som krävs för att spegla uppgiftens krav. Relationernas kardinalitet och attributens egenskaper har definierats för att enkelt kunna hantera databasstrukturen. För att sedan skapa databasen användes astah för att generera SQL queries och bygga alla tables. Manuella SQL-queries utfördes, och sedan pg_dump för att skapa en dump som sedan kan återställas till en databas. SQL användes för att skapa tabellerna och definiera deras relationer.
Länk till github:
https://github.com/biggaben/DatalagringProjekt/tree/main/Task-3

```sql
-- Task 3.1
CREATE VIEW month_lesson_year AS
SELECT
    CASE EXTRACT(MONTH FROM b.start_time)
        WHEN 1 THEN 'Jan'
        WHEN 2 THEN 'Feb'
        WHEN 3 THEN 'Mars'
        WHEN 4 THEN 'April'
        WHEN 5 THEN 'Maj'
        WHEN 6 THEN 'Juni'
        WHEN 7 THEN 'Juli'
        WHEN 8 THEN 'Aug'
        WHEN 9 THEN 'Sep'
        WHEN 10 THEN 'Okt'
        WHEN 11 THEN 'Nov'
        WHEN 12 THEN 'Dec'
    END AS month,
    COUNT(b.booking_id) AS total_lessons,
    COUNT(sl.solo_lession_id) AS solo_lessons,
    COUNT(gl.group_lesson_id) AS group_lessons,
    COUNT(e.ensamble_id) AS ensamble_lessons
FROM
    booking b
LEFT JOIN
    solo_lesson sl ON b.booking_id = sl.booking_id
LEFT JOIN
    group_lesson gl ON b.booking_id = gl.booking_id
LEFT JOIN
    ensamble e ON b.booking_id = e.booking_id
WHERE
    EXTRACT(YEAR FROM b.start_time) = 2023
GROUP BY
    EXTRACT(MONTH FROM b.start_time);
```

Figure 1: Kod för vyn ”month_lesson_year” som skriver antal lektioner per månad.

	month text	total_lessons bigint	solo_lessons bigint	group_lessons bigint	ensamble_lessons bigint
1	Jan	1	1	0	0
2	Feb	1	1	0	0
3	Mars	1	1	0	0
4	April	1	1	0	0
5	Maj	1	1	0	0
6	Juni	1	0	1	0
7	Juli	1	0	0	1
8	Dec	5	2	0	3

Figure 2: Bild på vyn ”month_lesson_year” som skriver antal lektioner per månad.

5 Discussion

Views har använts för queries för att göra dessa hanterbara. Till exempel, i frågan att räkna antalet lektioner per månad, används en view för att sammanfatta de initiala beräkningarna. Detta gjorde det enklare att förstå den övergripande logiken i frågan.

Vissa delar i databasdesignen ändrades för att förenkla queries och få databasen att fungera som den ska. Fokus låg även på att skriva effektiva SQL-kod som fungerar bra

```
60
61  -- TASK 3.2 Siblings-tabell
62  CREATE VIEW student_siblings AS
63  SELECT
64    No_of_Siblings,
65    COUNT(*) AS No_of_Students
66  FROM (
67    SELECT
68      s.student_id,
69      COUNT(DISTINCT sg.sibling_group_id) AS No_of_Siblings
70    FROM
71      student s
72    LEFT JOIN
73      sibling_group sg ON s.student_id = sg.student_id
74    GROUP BY
75      s.student_id
76  ) AS SiblingCounts
77  GROUP BY
78    No_of_Siblings
79  ORDER BY
80    No_of_Siblings;
81
82
```

Figure 3: Kod för vyn ”student_siblings” som skriver antal studenter med syskon, och antal syskon.

	no_of_siblings bigint	no_of_students bigint
1	0	18
2	1	6
3	2	3

Figure 4: Bild på vyn ”student_siblings” som skriver antal studenter med syskon, och antal syskon.

med databasstrukturen. EXPLAIN-kommandot användes för att analysera querien som räknar antalet lektioner per månad. Mycket tid spenderades på att joina tabeller och utföra aggregeringar. Jag övervägde möjligheten att omskriva frågan för att ytterligare förbättra dess prestanda, till exempel genom att optimera join-kommandon, men nuvarande lösning kändes rimlig.

```sql

-- Task 3.3
CREATE VIEW list_instructor_id AS
SELECT
  i.instructor_id,
  i.first_name,
  i.last_name,
  COUNT(*) AS No_of_Lessons
FROM
  booking b
INNER JOIN instructor i ON b.instructor_id = i.instructor_id
INNER JOIN booking_lesson bl ON b.booking_id = bl.booking_id
WHERE
  b.start_time >= date_trunc('month', current_date)
  AND b.end_time < date_trunc('month', current_date) + interval '1 month'
GROUP BY
  i.instructor_id
HAVING
  COUNT(*) > 0
ORDER BY
  No_of_Lessons DESC;


```

Figure 5: Kod för vyn ”list_instructor_id” som skriver antal lektioner varje lärare har.

	instructor_id integer	first_name character varying (50)	last_name character varying (50)	no_of_lessons bigint
1	8012	Erika	Eriksson	8
2	8010	Jakob	Johansson	4
3	8008	Hans	Hansson	3
4	8001	David	Dahl	1

Figure 6: Bild på vyn ”list_instructor_id” som skriver antal lektioner varje lärare har.

```sql

--Task 3.4
CREATE VIEW ensambles_comming_week AS
SELECT
  to_char(b.start_time, 'Day') AS Day,
  e.genre AS Genre,
  CASE
    WHEN max_students - COUNT(bl.student_id) <= 0 THEN 'No Seats'
    WHEN max_students - COUNT(bl.student_id) BETWEEN 1 AND 2 THEN '1 or 2 Seats'
    ELSE 'Many Seats'
  END AS "No of Free Seats"
FROM
  booking b
JOIN
  ensamble e ON b.booking_id = e.booking_id
LEFT JOIN
  booking_lesson bl ON e.booking_id = bl.booking_id
WHERE
  b.start_time >= current_date + interval '1 day' * EXTRACT(DOW FROM current_date) -- nästa vecka
  AND b.start_time < current_date + interval '1 day' * (7 - EXTRACT(DOW FROM current_date)) + interval '7 days'
GROUP BY
  e.genre, b.start_time, e.max_students
ORDER BY
  e.genre, to_char(b.start_time, 'Day');


```

Figure 7: Kod för vyn ”ensambles_comming_week” som skriver antal fria platser på ensembels.

	day text	genre character varying (50)	No of Free Seats text
1	Tuesday	house	No Seats
2	Sunday	punk	1 or 2 Seats
3	Monday	rock	Many Seats

Figure 8: Bild på vyn ”ensambles_comming_week” som skriver antal fria platser på ensembels.