

Projektrapport- SQL

Datalagring, IV1351

Mattias Sandberg matsandb@kth.se

2022-12-15

1 Introduktion

Syftet med detta projekt är att designa en applikation och databas för Soundgood Music School. Applikationen kommer att hantera transaktionerna som uthyrningen av instrument medan databasen kommer hantera all information som musikskolan innehåller. I detta projekt så kommer jag att jobba nära Viktor Björken för att underlätta och förbättra min inläring. Projektet kommer att vara indelat i fyra delar vilket är följande (Konceptuell modell, Logisk och fysisk modell, SQL, Programmatisk åtkomst). Astah är det dataprogrammet som jag använder för att konstruera den konceptuella modellen över musikskolan. I Astah så visas alla relationer, enheter och attribut som är väsentliga för musikskolan ska kunna fungera.

2 Litteraturstudier & förberedelser

2.1 Uppgift 3 – SQL

Den första saken som jag gjorde för att förbereda mig inför denna uppgift var att kolla på samtliga föreläsningar. Därefter så läste jag dokumentet "tips-and-tricks-task3.pdf" från projektets hemsida på canvas. Eftersom jag inte var bekant med SQL queries så var jag tvungen att söka ytterligare information om hur man bygger dessa databaser. Denna information hittade jag på Youtube från andra universitet. Hemsidan "geeksforgeeks" hjälpte mig att bättre förstå hur exempelvis "CASE" implementeras i SQL script.

3 Metod

3.1 Uppgift 3 – SQL

Målet med denna uppgift var att skapa OLAP queries som exikverar vissa funktioner enligt beskrivningen på canvas. Under detta projekt använde vi PostgreSQL med psql-shell när vi skrev SQL kod och hanterade databasen, vi använde även pgAdmin som GUI för enkelhetens skull och för att bättre förstå hur saker fungerade.

Vi behövde skapa mer slumpmässig data till vårt inser-data program för att OLAP queries skulle kunna köras korrekt. För att kolla om skripten var korrekta så räknade vi manuellt från insert-data programmet och såg om informationen stämde överens med datan som de nya funktionerna gav. Det var ingen step-by-step metod i denna uppgift. Vi skapade queries i ordning enligt canvas och gjorde ändringar i databasen när det var nödvändigt.

4 Resultat

4.1 uppgift 3 – SQL

Länk till GitHub: <https://gits-15.sys.kth.se/matsandb/IV1351>

Första uppgiften var: Show the number of lessons given per month during a specified year. This query is expected to be performed a few times per week. It shall be possible to retrieve the total number of lessons per month (just one number per month) and the specific number of individual lessons, group lessons and ensembles (three numbers per month). It's not required that all four numbers (total plus one per lesson type) for a particular month are on the same row; you're allowed to have one row for each number as long as it's clear to which month each number belongs. However, it's most likely easier to understand the result if you do place all numbers for a particular month on the same row, and it's an interesting exercise, therefore you're encouraged to try.

Denna kod räknar antalet lektioner, såväl individuella som grupp- och ensembleslektioner, som instruktörer har gett under det angivna året. Den kombinerar alla lektioner i en tabell och filtrerar bort de lektioner som inte gavs under det angivna året. Sedan räknar den antalet grupplektioner, individuella lektioner och ensembleslektioner som gavs under varje månad och presenterar resultatet i en tabell. Nedan presenteras resultatet i formen av ett urklipp från pgAdmin där den första bilden representerar 2022 och den andra bilden representerar 2023:



| | month numeric | total_lessons bigint | individual_lessons bigint | group_lessons bigint | ensemble_lessons bigint |
|---|------------------|-------------------------|------------------------------|-------------------------|----------------------------|
| 1 | 12 | 13 | 5 | 5 | 3 |

| | month numeric | total_lessons bigint | individual_lessons bigint | group_lessons bigint | ensemble_lessons bigint |
|---|------------------|-------------------------|------------------------------|-------------------------|----------------------------|
| 1 | 2 | 2 | 0 | 0 | 2 |

Andra uppgiften var: Show how many students there are with no sibling, with one sibling, with two siblings, etc. This query is expected to be performed a few times per week. The database must contain students with no sibling, one sibling and two siblings, but doesn't have to contain students with more than two siblings. Note that it's not allowed to solve this by just adding a column with sibling count (maybe called no_of_siblings or something similar) to the student table. Such a solution would be almost impossible to maintain since it doesn't tell who's a sibling of who. If a student quits, there won't be any way to update the sibling count of that student's siblings.



Denna query räknar antalet syskon för varje elev och grupperar resultaten efter elev och antal syskon och sorterar sedan resultaten efter antalet syskon. Den förutsätter att syskontabellen innehåller en rad per syskonpar. Den slutliga resultattabellen nedan visar varje elev och antalet syskon de har. Enligt denna tabell så har student nummer ett, två och

tre 2 syskon och student nummer fyra och fem ett syskon. Detta stämmer överens vår insertdata.

| | student_id integer  | no_of_siblings bigint  |
|---|--|---|
| 1 | 4 | 1 |
| 2 | 5 | 1 |
| 3 | 1 | 2 |
| 4 | 2 | 2 |
| 5 | 3 | 2 |

Tredje uppgiften var: List all instructors who has given more than a specific number of lessons during the current month. Sum all lessons, independent of type, and sort the result by the number of given lessons. This query will be used to find instructors risking to work too much, and will be executed daily.

Detta är en SQL- query som räknar antalet lektioner som instruktören har gett under den aktuella månaden. Frågan börjar med en SELECT sats som specificerar att instruktörens ID och summan av antalet lektioner ska returneras. Detta följs av en under-query som använder UNION ALL för att kombinera resultaten från tre olika tabeller: group_lesson, individual_lesson och ensemble. För varje tabell räknar under-query antalet lektioner som instruktören har gett under den aktuella månaden och grupperar dem enligt instruktörens ID. Den slutliga query använder GROUP BY och HAVING för att lägga resultaten efter instruktörens ID och bara returnera resultat där instruktören har gett fler än en lektion under den aktuella månaden. Till sist sorteras resultatet efter det totala antalet lektioner i nedåtgående ordning. Nedan presenteras resultatet i formen av ett urklipp från pgAdmin:

| | instructor_id integer  | total_lessons numeric  |
|---|---|---|
| 1 | 1 | 3 |
| 2 | 2 | 3 |
| 3 | 3 | 3 |
| 4 | 4 | 2 |
| 5 | 5 | 2 |

Fjärde uppgiften var: List all ensembles held during the next week, sorted by music genre and weekday. For each ensemble tell whether it's full booked, has 1-2 seats left or has more seats left. Hint: you might want to use a CASE statement in your query to produce the desired output.

Koden till detta program hämtar data från tabellen "ensemble" och filtrerar den för att bara visa de rader där den schemalagda tiden ligger inom de kommande 7 dagarna från nuvarande tidpunkt. Den hämtar id, målgenre, skol-ID, schemalagd tidslucka och instruktörs-ID. Sedan används CASE för att avgöra hur många platser som är kvar i klassen. Resultatet sorteras sedan genom veckodag och målgenre. Nedan presenteras resultatet i formen av ett urklipp från pgAdmin:

| | id [PK] integer | scheduled_time_slot [PK] timestamp without time zone | target_genre character varying (500) | instructor_id [PK] integer | school_id [PK] integer | weekday numeric | seat_status text |
|---|--------------------|---|---|-------------------------------|---------------------------|--------------------|---------------------|
| 1 | 1 | 2022-12-16 10:00:00 | Classical | 1 | 1 | 5 | has more seats left |
| 2 | 2 | 2022-12-18 11:00:00 | Jazz | 2 | 1 | 0 | has more seats left |
| 3 | 3 | 2022-12-20 12:00:00 | Rock | 3 | 2 | 2 | has more seats left |

5 Diskussion

5.1 Uppgift 3 – SQL

Is there any correlated subquery, that is a subquery using values from the outer query? Remember that correlated subqueries are slow since they are evaluated once for each row processed in the outer query.

Ja, det finns en korrelerad under-query i TASK3.2. Den korrelerade under-queryn är den inre SELECT-sasten som använder GROUP BY-satsen för att gruppera rader från tabellen syskon efter värdet på kolumnen student_id. Under-queryn är korrelerad eftersom den använder kolumnen student_id i sin WHERE-sats som är ett värde från den yttre-queryn. I den yttre-queryn används denna under-queryn som en härledd tabell. Den yttre-queryn grupperar sedan raderna i den härledda tabellen efter värdet i kolumnen no_of_siblings och räknar antalet rader i varje grupp, vilket i sin tur ger antalet elever med ett visst antal syskon. Denna korrelerade subquery är den enda som jag identifierar i våra queries.

Are there unnecessarily long and complicated queries? Are you for example using a UNION clause where it's not required?

Jag bedömer att våra queries är rimligt långa i förhållande till dess syfte. Detta baseras givetvis på våra nuvarande kunskaper. Men vi har gått igenom vår kod och ser inga självklara förbättringar.

Did you change the database design to simplify these queries? If so, was the database design worsened in any way just to make it easier to write these particular queries?

Vi valde att modifiera vår databas och insertdata program för att våra queries skulle kunna uppfylla det önskvärda operationerna. Några exempel på detta var att vi behövde byta

tiderna på när våra ensembles skulle hållas. Vi ändrade även strukturen något på hur syskon räknas så man kan se vilka som är syskon med vilka så det inte blir något problem om någon elev slutar. Vi flyttade därmed discount till monthly_fee. Dessa var några modifieringar som vi gjorde och våra queries funkar som det ska och det försämrade inte vår databas på något annat sätt vad vi kan se.

Analyze the query plan for at least one of your queries using the command EX-PLAIN (or EXPLAIN ANALYZE), which is available in both Postgres and MySQL. Where in the query does the DBMS spend most time? Is that reasonable? If you have time, also consider if the query can be rewritten to execute faster, but you're not required to do that. The postgres documentation is found at <https://www.postgresql.org/docs/current/using-explain.html> and <https://www.postgresql.org/docs/current/sql-explain.html>. There's also some explanation of EX-PLAIN in the document Tips and Tricks for Project Task 3.

"cost" är en variabel som identifierar exekveringstid. Det är den faktiska tiden det krävs för att hämta en sekvens från disken. Det är viktigt att jämföra kostnader då man kan se eventuella problem tidsmässigt i koden. Med detta sagt så kan vi inte direkt översätta "cost" med exekveringstid. I urklippet nedan från TASK3.4 så har rad nummer ett högst "cost" vilket är benämnt följande: cost=13.17..13.18. Detta betyder att det första värdet 13.17 är den estimerade tiden innan output startar och det andra värdet 13.18 är tiden då noden är slutförd.

| | QUERY PLAN | |
|---|--|---|
| | text | 🔒 |
| 1 | Sort (cost=13.17..13.18 rows=1 width=600) | |
| 2 | Sort Key: target_genre, (EXTRACT(dow FROM scheduled_time_slot)) | |
| 3 | -> Seq Scan on ensemble (cost=0.00..13.16 rows=1 width=600) | |
| 4 | Filter: ((scheduled_time_slot >= CURRENT_TIMESTAMP) AND (scheduled_time_slot <= (CURRENT_TIMESTAMP + '7 days'::interval))) | |