Database System Project – Phase 2 (Group#11)

By

Alice Chen (40176279), aliceykchen01@gmail.com

Maxx Freund (40174065), maxxf@live.ca

Ryan Kim (40175423), rck2001@hotmail,com

Maria Rivas (40174860), mafer.rivas@bell.net

William Tremblay (40174212), wt22@videotron.ca

A report submitted in partial fulfillment of the requirements of SOEN 363

Concordia University

March 2023

1. Queries

See the attached .txt and .csv files for the queries and their result.

2. Performance

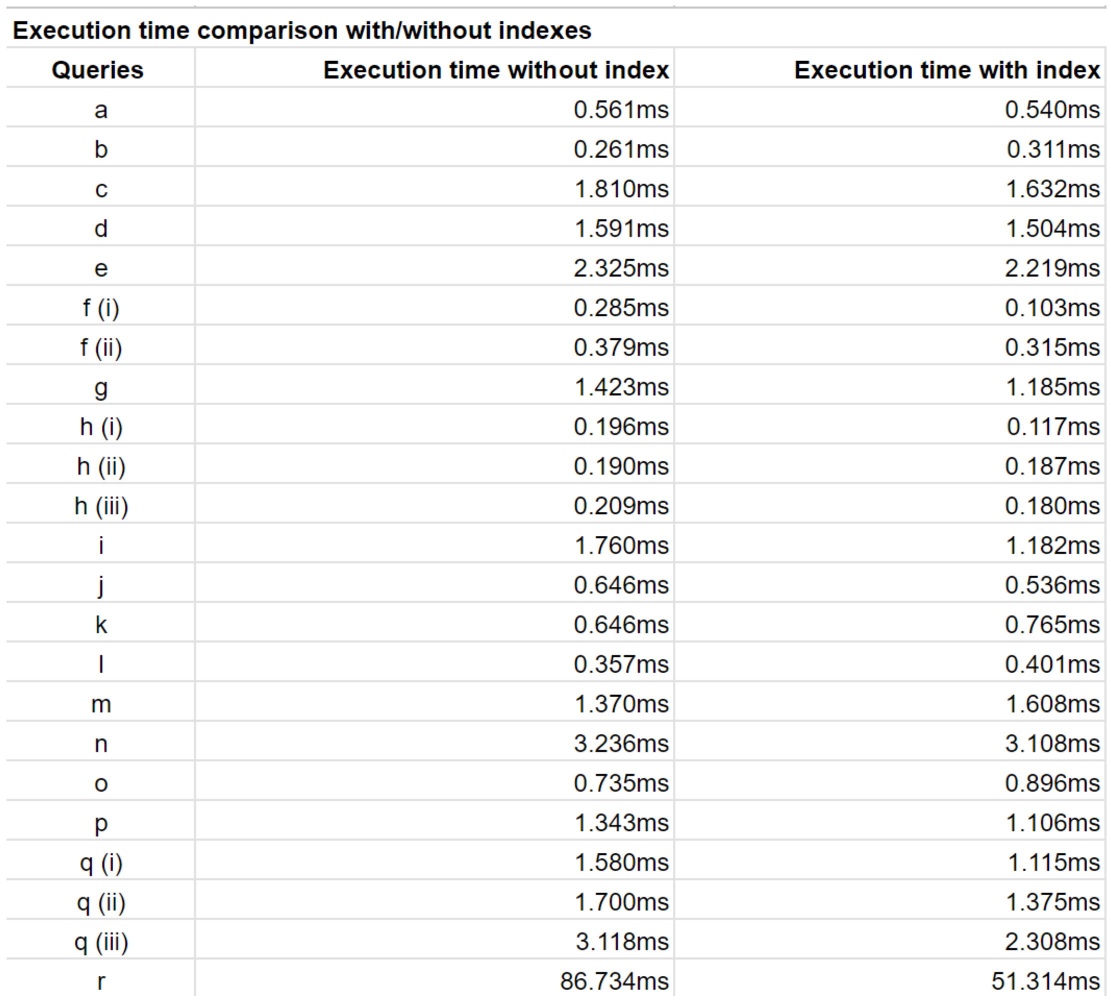
a) Indexes:

```
CREATE INDEX idx1
ON hashtag (tweetid, hashtagid);

CREATE INDEX idx5
ON tweets (tweetid, userid);

CREATE INDEX idx6
ON usertwitter (userid);
```

b) Table 1



Execution time comparison with/without indexes		
Queries	**Execution time without index**	**Execution time with index**
a	0.561ms	0.540ms
b	0.261ms	0.311ms
c	1.810ms	1.632ms
d	1.591ms	1.504ms
e	2.325ms	2.219ms
f (i)	0.285ms	0.103ms
f (ii)	0.379ms	0.315ms
g	1.423ms	1.185ms
h (i)	0.196ms	0.117ms
h (ii)	0.190ms	0.187ms
h (iii)	0.209ms	0.180ms
i	1.760ms	1.182ms
j	0.646ms	0.536ms
k	0.646ms	0.765ms
l	0.357ms	0.401ms
m	1.370ms	1.608ms
n	3.236ms	3.108ms
o	0.735ms	0.896ms
p	1.343ms	1.106ms
q (i)	1.580ms	1.115ms
q (ii)	1.700ms	1.375ms
q (iii)	3.118ms	2.308ms
r	86.734ms	51.314ms

c) Materialized views:

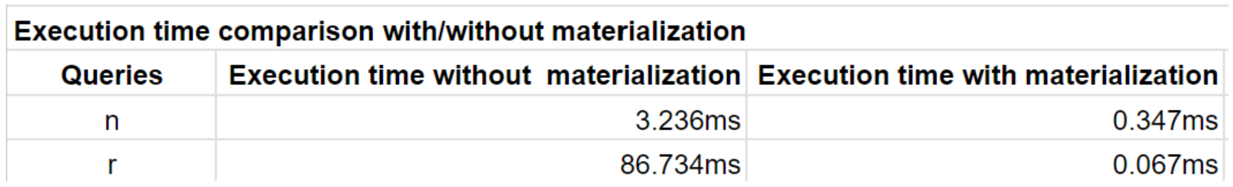
```
--Query R
Create materialized view mv1
As
SELECT usertwitter.userid, hashtag.text
FROM hashtag
JOIN tweets ON tweets.tweetid = hashtag.tweetid -- match table1 & table 2
JOIN usertwitter ON usertwitter.userid = tweets.userid -- match table2 & table 3
WHERE hashtag.text ~ '(\w).*\1.*\1.*\1';

Select * from mv1
```

```
--Query N
Create materialized view mv2
As
SELECT UT.username, T.text
FROM tweet T
JOIN tweets TTs ON TTs.tweetId = T.tweetId
JOIN usertwitter UT ON UT.userId = TTs.userId

Select * from mv2
```

Table 2



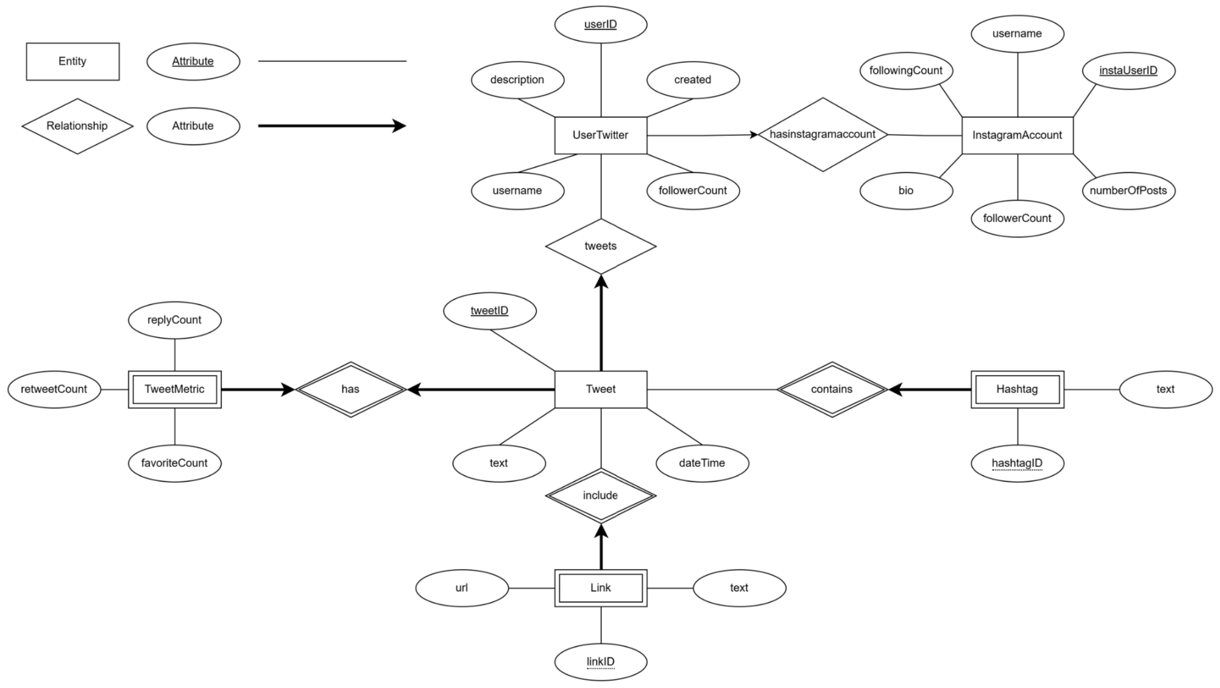
Execution time comparison with/without materialization		
Queries	**Execution time without materialization**	**Execution time with materialization**
n	3.236ms	0.347ms
r	86.734ms	0.067ms

Discussion

Both indexes and materialized views help improve the performance. As seen in both tables above, the execution times with these methods are lower for the queries that are affected by the specified indexes and materialized views. However, as seen in *Table 2*, the materialized views significantly improve the performance of the queries n and r more so than indexes.

3. Data Enrichment

a) We decided to enrich our data by using Instagram. Indeed, we obtained the Instagram profiles of the users whose username matched one from twitter that we already had in our database. We then put this data into two new tables, defined by the following relations:
   - instagramaccount(<u>instagramuserid</u>, username, bio, numberofposts, followercount, followingcount)
   - hasinstagramaccount(<u>userid</u>, <u>instagramuserid</u>)



This is how our new conceptual model looks like with the addition of the new tables:

The *instagramaccount* table contains information about the instagram profiles, such as the instagram user id, the username, the bio, the number of posts published, and the number of accounts that follows the user and that the user follows. The *hasinstagramaccount* table relates a twitter user with its instagram account by matching the twitter user id with its corresponding instagram user id.

b) Not only does integrating new databases add to the breadth of information stored, but it can also allow for more precise queries based on the new data acquired. The process of combining data from different sources into a consolidated database is known as database integration. This allows for users of a database to access information from a common location, resulting in quicker query results, as well as allows database managers to more effectively manage data tables in a centralized manner. To elaborate, as database managers, we stand to benefit from database integration, since all collected data would be consolidated and standardized in one location. There would be no need to reference other outside databases, which may have deprecated information; thus increasing team productivity and understanding.

In terms of the queries, we have three of them that demonstrate essential information regarding the database that was integrated. They are located in the submitted folder with their query and result (Q3-b-i, Q3-b-ii, and Q3-b-iii).

The first query (Q3-b-i) returns a list of all the UserTwitter usernames that have matching usernames in the InstagramAccount table, with duplicates that are removed. These two tables combined into one single query help to identify patterns and relationships that might not be apparent from looking at each database separately by its own.

The second query (Q3-b-ii) joins the new table InstagramAccount with the UserTwitter and Tweets tables. This allows one to get a list of Instagram usernames and the hashtags used in their posts from Twitter. This helps us to understand the nature of users on Instagram and what their posts contain, which is the topic of our project.

For the last query (Q3-b-iii), it provides additional information about the number of tweets and the average and maximum follower count for each user who has a Twitter and Instagram account.

Hence, with the queries and the explanation, it shows how to benefit from integrating new databases by using SQL queries with the aggregate functions, Group By, and Order By. It discusses three specific queries that provide insights into relationships between tables and the users' nature on Instagram and their posts, which is the topic of our project. These queries illustrate the benefits of integrating databases and using SQL to extract useful information.

c) New indexes:

```
CREATE INDEX idx7
ON instagramaccount (instagramuserid);

CREATE INDEX idx8
ON hasinstagramaccount (userid, instagramuserid);
```

Table 3

Query	Execution Time without Index	Execution Time with Indexes
Q3-b-i	0.881 ms	0.558 ms
Q3-b-ii	1.745 ms	1.634 ms
Q3-b-iii	1.965 ms	1.870 ms

In this section, we introduced two indexes, one for each newly created table. As it is possible to notice in Table 3, the execution time is improved for every query defined in b). We can imagine that if we implemented materialized views for those tables as well, we would find a more significant improvement, as we saw in