HW3

Paul de Fusco

DSE201

Cats Queries:

1.

Time with no modifications: 235 msc

Modifications:

First tried to create following indexes for the subqueries:

create index user\_id\_index on cats.likes(user\_id);

create index video\_id\_index on cats.likes(video\_id);

create index userid\_index on cats.watch(user\_id);

create index videoid\_index on cats.watch(video\_id);

This caused no improvement as query time was 247 msc

There already are primary keys on video\_id in the video table, user\_id in the user table. Video\_id and user\_id in the likes table are foreign keys

I then deleted the above indexes and created the following multi indexes which seemed to slow down the query dramatically to 363 msc:

create index multi\_index\_usrid\_vdid on cats.likes(user\_id,video\_id);

create index watch\_multi\_index\_usrid\_vdid on cats.watch(user\_id,video\_id);

Conclusion: no index recommendation for this query

2.

Time with no modifications: 239 msc

Modifications:

Added the following indexes:

CREATE INDEX likes\_userid on cats.likes(user\_id);

CREATE INDEX likes\_videoid on cats.likes(video\_id);

CREATE INDEX friend\_userid on cats.friend(user\_id);

CREATE INDEX friend\_friendid on cats.friend(friend\_id);

Obtained 235 msc runtime after modifications

I then deleted the two indexes on likes and kept the two indexes on friend and reran and obtained 321 msc (decrease in performance):

drop index cats.likes\_userid

drop index cats.likes\_videoid

I then created two multicolumn indexes and recreated the two indexes on the likes table:

CREATE INDEX likes\_userid on cats.likes(user\_id);

CREATE INDEX likes\_videoid on cats.likes(video\_id);

CREATE INDEX multi\_likes on cats.likes(user\_id, video\_id);

CREATE INDEX multi\_friend on cats.friend(user\_id, friend\_id);

This time the query decreased to 162 msc

Conclusion: create the above six indices (likes\_userid, likes\_videoid, friend\_userid, friend\_friendid, multi\_likes, multi\_friend)

3.

Time with no modifications: 233 msc

Modifications:

I recreated the following multi index on friend:

CREATE INDEX multi\_friend on cats.friend(user\_id, friend\_id);

The query time was 253 msc.

I kept the above index and also recreated the following multi index on likes:

CREATE INDEX multi\_likes on cats.likes(user\_id, video\_id);

The query time dropped to 233 msc.

I kept the above indexes and also recreated the following single column indexes:

CREATE INDEX likes\_userid on cats.likes(user\_id);

CREATE INDEX friend\_userid on cats.friend(user\_id);

The query time dropped to 182 msc.

Conclusion: create the four indices: multi\_friend, multi\_likes, likes\_userid, friend\_userid

4.

Time with no modifications: 283 msc

Modifications:

I recreated the indices from the above solution:

CREATE INDEX likes\_userid on cats.likes(user\_id);

CREATE INDEX friend\_userid on cats.friend(user\_id);

CREATE INDEX multi\_friend on cats.friend(user\_id, friend\_id);

CREATE INDEX multi\_likes on cats.likes(user\_id, video\_id);

Query time dropped to 228 msc.

Conclusion: create the above for indices. I don’t see any use for any additional indices

5.

Time with no modifications: 412 msc

Modifications:

First I created the following multi column index:

CREATE INDEX multi\_likes on cats.likes(user\_id, video\_id);

I obtained a decrease to 313 msc.

In a new trial, I recreated the above index and also added two single indices:

CREATE INDEX multi\_likes on cats.likes(user\_id, video\_id);

CREATE INDEX likes\_videoid on cats.likes(video\_id);

CREATE INDEX likes\_userid on cats.likes(user\_id);

The query time increased to 371 msc.

In a third trial, I only recreated the two single indices:

CREATE INDEX likes\_videoid on cats.likes(video\_id);

CREATE INDEX likes\_userid on cats.likes(user\_id);

Query time increased even further to 424 msc.

Conclusion: I don’t see the use of adding an index on user.user\_id as it is a primary key, nor other indices so I would just create the multi\_likes index as in my first test scenario above.

Sales Queries:

1.

Time with no modifications: 176 msc

Modifications:

I added an index on sale.customer\_id but that increased query time to 413 msc

create index sales\_customer\_id on sales.sale(customer\_id)

In a new scenario, I removed the above index and created a new index on customer.customer\_id:

create index cust\_customer\_id on sales.customer(customer\_id)

Query time was now 318 msc.

In a last scenario, I tried adding an index on sale.quantity (after removing the above index):

create index cust\_customer\_id on sales.sale(quantity)

The query time now was 179 msc

Conclusion: I would recommend not creating any indices

2.

Time with no modifications: 356 msc

Modifications:

In a first scenario I added the following multi column index on customer to help the join on state\_id obtaining query time of 309 msc:

create index multi\_cust on sales.customer(customer\_id, state\_id)

Because that helped, in a second case scenario I recreated the above index and also added a multi column index on the state table obtaining query time of 209 msc:

create index multi\_cust on sales.customer(customer\_id, state\_id)

create index multi\_state on sales.state(state\_id, state\_name)

Conclusion: use the multi\_cust and multi\_state indices

3.

Time before modifications: 233 msc.

Modifications:

I created the following index on sales.customer\_id obtaining a small decrease in query time to 231 msc:

create index sale\_cust\_id on sales.sale(customer\_id)

In a second scenario I dropped the index above and created the following multicolumn index on the sales table:

create index multi\_sales on sales.sale(customer\_id, product\_id)

Query time increased to 417 msc.

Conclusion: It serves to notice that the two columns in the index above are foreign keys so that might explain why they are not useful. The recommendation is not to implement any indices

4.

Time before modifications:

12 seconds

Modifications:

In my first scenario I created two multi column indices:

create index multi\_customer on sales.customer(customer\_id, customer\_name)

create index multi\_sale on sales.sale(product\_id, customer\_id)

Query time was again 12 seconds.

Then, I tested the two indices above individually:

create index multi\_sale on sales.sale(product\_id, customer\_id)

With Multi\_sale only, query time reached again 12 seconds.

create index multi\_customer on sales.customer(customer\_id, customer\_name)

With multi\_customer only, query time reached 13 seconds.

A new scenario with a single index on sale.customer\_id also gave 12 seconds:

create index sale\_cust\_id on sales.sale(customer\_id)

In a new scenario, I dropped all preceding indices and added a new multi column index on quantity and price in the sale table:

create index multi\_sale\_price on sales.sale(price, quantity)

Query time was again 12 seconds.

Conclusion: I ran out of ideas, maybe I don’t have enough rows but otherwise I would recommend no new indices

5.

Time before modifications: 586 msc

Modifications:

Conclusion: all attributes used are primary keys in their own tables (besides quantity sold and dollar value aggregate functions). No new indices are recommended

6.

Time before modifications: 1 second

Modifications:

Many portions of the query rely on dollar value being calculated in q1 so I created the following index:

create index multi\_sale\_price on sales.sale(price, quantity)

Query time decreased to 968 msc

Category ID and Customer ID are both primary keys respectively in the Category and Customer tables. Therefore, I can recommend no more indices.

Conclusion: create index multi\_sale\_price