Data base programming - Project

Creating indices to optimize the database schema in general:-

- **- actor table:-**
  CREATE INDEX idx_actor_id on actor USING BTREE (actor_id);
  CREATE INDEX idx_actor_last_name on actor USING BTREE (last_name);
  CREATE INDEX idx_actor_first_name on actor USING BTREE (first_name);
- - **payment:-**
  -- using those unique keys because they arent initialized as primary keys they dont have the red mark in this table.
  CREATE INDEX idx_payment_id on payment USING BTREE (payment_id);
  CREATE INDEX idx_pcustomer_id on payment USING BTREE (customer_id);
  CREATE INDEX idx_pstaff_id on payment USING BTREE (staff_id);
  CREATE INDEX idx_prental_id on payment USING BTREE (staff_id);
  CREATE INDEX idx_pay_amount_id on payment USING BTREE (amount);
- **- language table:-**
  CREATE INDEX idx_language_id on language USING BTREE (language_id);
  CREATE INDEX idx_language_name on language USING HASH (name);
- **- store table:-**
  CREATE INDEX idx_store_id on store USING BTREE (store_id);
  CREATE INDEX idx_f_store_address_id on store USING BTREE (address_id);
  CREATE INDEX idx_smanager_staff_id on store USING BTREE (manager_staff_id);
- **- address table:-**
  CREATE INDEX idx_address_id on address USING BTREE (address_id);
  CREATE INDEX idx_postal_code on address USING BRIN (postal_code) WITH (pages_per_range = 128);

CREATE INDEX idx_f_address_city_id on address USING BTREE (city_id);
CREATE INDEX idx_address_district on address USING BTREE (district);

- **- film_actor table:-**
  CREATE INDEX idx_f_actor_id on film_actor USING BTREE (actor_id);
  CREATE INDEX idx_f_filmactor_film_id on film_actor USING BTREE (film_id);
  CREATE INDEX idx_filmt_title on film USING BTREE (title);
  CREATE INDEX idx_film_year on film USING HASH (release_year);
  CREATE INDEX idx_film_rating on film USING BTREE (rating);
- **- country table:-**
  CREATE INDEX idx_country_id on country USING BTREE (country_id);
  CREATE INDEX idx_country_name on country USING HASH (country);
- **- city table:-**
  CREATE INDEX idx_city_id on city USING BTREE (city_id);
  CREATE INDEX idx_city_name on city USING HASH (city);
  CREATE INDEX idx_f_city_country_id on city USING BTREE (country_id);
- **- film table:-**
  CREATE INDEX idx_film_id on film USING BTREE (film_id);
  CREATE INDEX idx_f_film_language_id on film USING BTREE (language_id);
- **- film_category table:-**
  CREATE INDEX idx_f_fcat_film_id on film_category USING BTREE (film_id);
  CREATE INDEX idx_f_fcat_category_id on film_category USING BTREE (category_id);
- **- category tab:-**
  CREATE INDEX idx_category_id on category USING BTREE (category_id);
  CREATE INDEX idx_category_name on category USING HASH (name);
- **- inventory table:-**
  CREATE INDEX idx_inventory_id on inventory USING BTREE (inventory_id);
  CREATE INDEX idx_f_inv_film_id on inventory USING BTREE (film_id);
  CREATE INDEX idx_f_inv_store_id on inventory USING BTREE (store_id);
- **- staff table:-**
  CREATE INDEX idx_staff_id on staff USING BTREE (staff_id);
  CREATE INDEX idx_f_staff_address_id on staff USING BTREE (address_id);

CREATE INDEX idx_f_staff_store_id on staff USING BTREE (store_id);
CREATE INDEX idx_staff_last_name on staff USING BTREE (last_name);
CREATE INDEX idx_staff_first_name on staff USING BTREE (first_name);
CREATE INDEX idx_staff_username on staff USING BTREE (username);

- **- customer table:-**
  CREATE INDEX idx_customer_id on customer USING BTREE (customer_id);
  CREATE INDEX idx_f_cus_store_id on customer USING BTREE (store_id);
  CREATE INDEX idx_f_cus_address_id on customer USING BTREE (address_id);
  CREATE INDEX idx_customer_last_name on customer USING BTREE (last_name);
  CREATE INDEX idx_customer_first_name on customer USING BTREE (first_name);
- **- rental table:-**
  CREATE INDEX idx_rental_id on rental USING BTREE (rental_id);
  CREATE INDEX idx_f_ren_inventory_id on rental USING BTREE (inventory_id);
  CREATE INDEX idx_f_ren_customer_id on rental USING BTREE (customer_id);
  CREATE INDEX idx_f_ren_staff_id on rental USING BTREE (staff_id);

some were on keys i didn't know they were indexed by default so i added the indices in the adjustment file next to this one.

- **performance:-**
  - **Before optimization:-**
    ---> latency average = 0.144 ms
    ---> tps = 6936.343956 (including connection establishing)
    tps = 6946.625067 (excluding connections establishing)
  - **ADJUSTED: second run after adding more indices.**
    ---> latency average = 0.148 ms
    ---> tps = 6771.151580 (including connection establishing)
    tps = 6781.134771 (excluding connections establishing)

  - **After optimization:-**

    --> latency average = 0.095 ms
    ---> tps = 10476.788964 (including connection establishing)
    tps = 10496.874421 (excluding connections establishing)

- **Adjustment optimization:-**

  --> latency average = 0.144 ms - another run 0.097 ms
  ---> tps = 6921.587833 (including connection establishing) - another run 10294.311483
  tps = 6933.332931 (excluding connections establishing) - another run 10313.515440

- The observation on indexing the general database schema in both tries i did the first one was with using keys as they are unique keys but didnt recognize that they are by default indexed so i did add some other attributes could be used for searching on indices queries the conclusion that the latency average it less than before the optimization and the tps are higher more than before the optimization.

Queries:-

- Some attributes could be used but didn’t use it anymore because they were already indexed in the schema so i was afraid that creating many indices that do similar jobs could affect the performance because i read that sometimes much indices would affect in negative way not positively.
- Most of the reasons of choosing the index type will be the same as btree and hash are the most used for optimizing the performance, also most of the observations will be the same.

- **Query_1: -**
  - **The attributes:**
    - category_id - using the BTREE indexing because handling this attribute will be flexible and easy to handle for fast searching and for better performance
    - film_id - using the BTREE indexing because handling this attribute will be flexible and easy to handle for fast searching and for better performance
    - title - here i used the hash index type as i could search the title or the name by hashing them alphabetically or by ids i could use the btree but as hash

indexing would be directly used and its o(1) so it would be faster that btree indexing

- name - same as the title.

- **Statistics of plan-time, execution-time, & index usage:-**
  - **Before:**
    - query plan time = 2454.39084000001
    - query execution time = 4568.097105000011
  - **After:**
    - query plan time = 5138.898921000019
    - query execution time = 8923.48153700003
    - index usage:-
      - how many times index used =
        idx_cat = 0
        idx_film_cat = 20000
        idx_film = 20000
        idx_film_title = 0
        idx_cat_name = 0
      - how many rows were fetched =
        idx_cat = 0
        idx_film_cat = 20000
        idx_film = 20000
        idx_film_title = 0
        idx_cat_name = 0
- **TPS & Average latency:-**
  - **Before:**
    - latency average = 0.905 ms
    - tps = 1105.457515(including connection establishing)
      tps = 1105.690366(excluding connections establishing)

        - **After:**
            - latency average = 0.859 ms
            - tps = 1164.113663(including connection establishing)
              tps = 1164.437230(excluding connections establishing)
        - comparison observations: the latency average is less than before optimization that means the indices made the performance of the query better than before, the execution time and the plan time are more than the numbers before the optimization, because of the previous queries took part in the execution time and it's a relation with the latency if the latency is getting less the execution and the total plan will get higher and the opposite.

- **Query_2:-**
    - **The attributes:-**
        - actor_id - using the BTREE indexing because handling this attribute will be flexible and easy to handle for fast searching and for better performance
        - first_name - using the BTREE indexing because handling this attribute will be flexible and easy to handle for fast searching and for better performance
        - last_name - same as the first_name
    - **Statistics of plan-time, execution-time, & index usage:-**
        - **Before:**
            - query plan time = 1659.9974409999975
            - query execution time = 4227.637881999994
        - **After:**
            - query plan time = 3444.6592329999926
            - query execution time = 8512.890651999987
            - index usage:-
                - how many times index used =
                  idx_ac_id = 20000
                  idx_fiac_id = 230000

idx_ac_fi = 0
idx_ac_la = 0

- how many rows were fetched =
  idx_ac_id = 20000
  idx_fiac_id = 0
  idx_ac_fi = 0
  idx_ac_la = 0

- **TPS & Average latency:-**
  - **Before:**
    - latency average = 0.767 ms
    - tps = 1302.961050(including connection establishing)
      tps = 1303.355980(excluding connections establishing)
  - **After:**
    - latency average = 0.732 ms
    - tps = 1365.325416(including connection establishing)
      tps = 1365.702384(excluding connections establishing)
    - comparison observations: the latency average is less than before optimization that means the indices made the performance of the query better than before, the execution time and the plan time are more than the numbers before the optimization, because of the previous queries took part in the execution time and it's a relation with the latency if the latency is getting less the execution and the total plan will get higher and the opposite.

- **Query_3:-**
- from this query some queries will run it again so they will have another runs next to the previous ones as it was giving not the best result because of the vm and laptop problem to get best results.
  - **The attributes:-**

- first_name - using the BTREE indexing because handling this attribute will be flexible and easy to handle for fast searching and for better performance
- last_name - using the BTREE indexing because handling this attribute will be flexible and easy to handle for fast searching and for better performance

- **Statistics of plan-time, execution-time, & index usage:-**
  - **Before:**
    - query plan time = 4893.430174000009 - 38000.079174000115
    - query execution time = 9004.378807999969 - 18878.450572000118
  - **After:**
    - query plan time = 36217.047111000196 - 39259.14364600018
    - query execution time = 18381.54681300006 - 19221.549454000196
    - index usage:-
      - how many times index used =
        idx_ac3_fi = 0
        idx_ac3_la = 0
      - how many rows were fetched =
        idx_ac3_fi = 0
        idx_ac3_la = 0
- **TPS & Average latency:-**
  - **Before:**
    - latency average = 0.276 ms - 0.334
    - tps = 3621.779552(including connection establishing) - 2993.679993
      tps = 3626.851392(excluding connections establishing) - 2995.291401
  - **After:**
    - latency average = 0.372 ms - 0.301
    - tps = 2691.373547(including connection establishing) - 3316.851890
      tps = 2693.525979(excluding connections establishing) - 3319.609409

- As you see the first records weren’t the best i think because of the vm and latptop problems or the background was occupied by the laptop a lot but after the second run the results were like the normal ones less latency and high tps even if they are less in difference but because they are small queries.

- **Query_4:-**
  - **The attributes:-**
    - active - using btree for this query was a better option more than the hash index because i received more latency with the hash index while my laptop and my fan getting bad and having many processes in the background so having higher latency and low pts. Choosing the active attribute because its not indexed before and searching into the names with active true values will short the time.
  - **Statistics of plan-time, execution-time, & index usage:-**
    - **Before:**
      - query plan time = 569.4600340000003
      - query execution time = 8061.647313999997
    - **After:**
      - query plan time = 3242.740714999989
      - query execution time = 61516.89225299978
      - index usage:-
        - how many times index used = idx_cus_active = 0
        - how many rows were fetched = idx_cus_active = 0
  - **TPS & Average latency:-**
    - **Before:**
      - latency average = 1.039 ms -

        - tps = 962.240188(including connection establishing)
          tps = 962.532030(excluding connections establishing)
    - **After:**
        - latency average = 1.320 ms
        - tps = 757.781803(including connection establishing)
          tps = 757.894266(excluding connections establishing)
        - latency and the pts aren’t the best results but thats the best results i could having because of the laptop background performance although i tried to close all the background processes but this is the best records i could get latency is higher than before optimization and the pts is less.

- **Query_5:-**
    - **The attributes:-**
        - active - using btree here was better than hash index as hashing made the latency higher than how much it is higher.
    - **Statistics of plan-time, execution-time, & index usage:-**
        - **Before:**
            - query plan time = 659.7282129999995 - 2545.841180999984
            - query execution time = 1470.0787199999977 - 6272.851345999954
        - **After:**
            - query plan time = 1914.52360599999 - 3200.1621309999796
            - query execution time = 4649.964339999971 - 7865.332640999911
            - index usage:-
                - how many times index used =
                  idx_customer_act = 0
                - how many rows were fetched =
                  idx_customer_act = 0
    - **TPS & Average latency:-**

        - **Before:**
            - latency average = 0.348 ms - 0.381
            - tps = 2875.517652(including connection establishing) - 2625.512280
              tps = 2877.138060(excluding connections establishing) - 2626.859880
        - **After:**
            - latency average = 0.387 ms - 0.383
            - tps = 2585.763721(including connection establishing) - 2610.820562
              tps = 2587.085707(excluding connections establishing) - 2612.263790
            - latency is higher a bit and the tps are less because of the vm and the laptop background working. The less differences because of the use of the indexing and the query is small it doesn't need that much of indexing.

- **Query_6:-**
    - **The attributes:-**
        - customer_id - choosing only the customer_id from rental as a reference key for more optimization of the searching the other attributes already indexed within the general schema, using the btree for better performance and flexiblity in searching.
    - **Statistics of plan-time, execution-time, & index usage:-**
        - **Before:**
            - query plan time = 1822.3133199999993
            - query execution time = 4868.538707000013
        - **After:**
            - query plan time = 3648.320509000002 - 9040.086571000036
            - query execution time = 9776.473508999987 - 570492.5342019974
            - index usage:-
                - how many times index used =
                  idx_rental_cusid = 220000

        - how many rows were fetched =
          idx_rental_cusid = 0

    - **TPS & Average latency:-**
        - **Before:**
            - latency average = 0.805 ms - 0.620
            - tps = 1242.481334(including connection establishing) - 1613.001282
              tps = 1242.786615(excluding connections establishing) - 1613.510287
        - **After:**
            - latency average = 0.816 ms - 0.859
            - tps = 1225.711872(including connection establishing) - 1163.660691
              tps = 1226.128241(excluding connections establishing) - 1163.921212
            - so appearntly that the new run of the workload is still affected by the background processes of the laptop and the previous reads are better from the average latency and the tps. Latency is higher and the tps is less.

- **Query_7:-**
    - **The attributes:-**
        - last_name - using btree for faster search , choosing this attribute to enhace the performance of searching of a last name doesnt start with A.
    - **Statistics of plan-time, execution-time, & index usage:-**
        - **Before:**
            - query plan time = 7185.00985600002
            - query execution time = 439503.9423779982
        - **After:**
            - query plan time = 9040.086571000036
            - query execution time = 570492.5342019974
            - index usage:-

        - how many times index used = idx_cus_lastname = 0
        - how many rows were fetched = idx_cus_lastname = 0
    - **TPS & Average latency:-**
        - **Before:**
            - latency average = 13.460 ms
            - tps = 74.291488(including connection establishing)
              tps = 74.292835(excluding connections establishing)
        - **After:**
            - latency average = 13.451 ms
            - tps = 74.345534(including connection establishing)
              tps = 74.346582(excluding connections establishing)
            - latency is less and the pts is higher means the performance is optimized.

- **Query_8:-**
    - **The attributes:-**
        - amount - choosing it because it’s the only attribute not used could be used to be indexed using the btree for better performance.
    - **Statistics of plan-time, execution-time, & index usage:-**
        - **Before:**
            - query plan time = 8156.499124999994
            - query execution time = 346762.5801789981
        - **After:**
            - query plan time = 14324.95464799998
            - query execution time = 624896.9144539982
            - index usage:-

        - how many times index used = payment_p2022_07_amount_idx1 = 0
        - how many rows were fetched = payment_p2022_07_amount_idx1 = 0
    - **TPS & Average latency:-**
        - **Before:**
            - latency average = 28.661 ms
            - tps = 34.890960(including connection establishing)
              tps = 34.891294(excluding connections establishing)
        - **After:**
            - latency average = 28.656 ms
            - tps = 34.896281(including connection establishing)
              tps = 34.896651(excluding connections establishing)
            - latency is less and the pts is higher means the performance is optimized.

- **Query_9:-**
    - **The attributes:-**
        - name - choosing the name for fast searching and for fast grouping by them using the btree indexing
        - title - choosing the title for fast searching and for fast grouping by them using the btree indexing
    - **Statistics of plan-time, execution-time, & index usage:-**
        - **Before:**
            - query plan time = 10660.233310000007
            - query execution time = 70158.55351999965
        - **After:**
            - query plan time = 21168.437660999974

- query execution time = 139393.59151199932
- index usage:-
  - how many times index used =
    idx_cat_nname = 0
    idx_f_t = 0
  - how many rows were fetched =
    idx_cat_nname = 0
    idx_f_t = 0

- **TPS & Average latency:-**
  - **Before:**
    - latency average = 8.392 ms
    - tps = 119.163313(including connection establishing)
      tps = 119.166371(excluding connections establishing)
  - **After:**
    - latency average = 8.260 ms
    - tps = 121.059553(including connection establishing)
      tps = 121.062706(excluding connections establishing)
    - latency is less and the pts is higher means the performance is optimized.

- **Query_10:-**
  - **The attributes:-**
    - description - choosing the description because its the attribute that is used and it should be indexed for a better search performance using the btree indexing for flexibility .
  - **Statistics of plan-time, execution-time, & index usage:-**
    - **Before:**
      - query plan time = 791.8515389999978
      - query execution time = 422624.6532269988

        - **After:**
            - query plan time = 1442.7413679999993
            - query execution time = 842299.994754997
            - index usage:-
                - how many times index used =
                  idx_film_des =0
                - how many rows were fetched =
                  idx_film_des = 0
    - **TPS & Average latency:-**
        - **Before:**
            - latency average = 42.498 ms
            - tps = 23.530621(including connection establishing)
              tps = 23.530737(excluding connections establishing)
        - **After:**
            - latency average = 42.175 ms
            - tps = 23.710695(including connection establishing)
              tps = 23.710839(excluding connections establishing)
            - latency is less and the pts is higher means the performance is optimized.

- **Query_11:-**
    - **The attributes:-**
        - name - choosing the name because its the attribute that is used and it should be indexed for a better search performance using the btree indexing for flexibility
    - **Statistics of plan-time, execution-time, & index usage:-**
        - **Before:**
            - query plan time = 9291.371429000043

        - query execution time = 172146.57924500003
    - **After:**
        - query plan time = 18658.233831000034
        - query execution time = 342951.0742019988
        - index usage:-
            - how many times index used =
              cat_name = 0
            - how many rows were fetched =
              cat_name = 0
- **TPS & Average latency:-**
    - **Before:**
        - latency average = 18.473 ms
        - tps = 54.134487(including connection establishing)
          tps = 54.135152(excluding connections establishing)
    - **After:**
        - latency average = 18.326 ms
        - tps = 54.567714(including connection establishing)
          tps = 54.568343(excluding connections establishing)
        - latency is less and the pts is higher means the performance is optimized.

- **Query_12:-**
    - **The attributes:-**
        - last_name - Choosing the last_name attribute for better performance using the btree indexing for better performance
    - **Statistics of plan-time, execution-time, & index usage:-**
        - **Before:**
            - query plan time = 522.520316000001

        - query execution time = 1302.234326
    - **After:**
        - query plan time = 891.524494999995
        - query execution time = 2179.0754710000065
        - index usage:-
            - how many times index used =
              act_lastname = 0
            - how many rows were fetched =
              act_lastname = 0
- **TPS & Average latency:-**
    - **Before:**
        - latency average = 0.322 ms
        - tps = 3101.563002(including connection establishing)
          tps = 3103.752601(excluding connections establishing)
    - **After:**
        - latency average = 0.269 ms
        - tps = 3711.426540(including connection establishing)
          tps = 3715.107035(excluding connections establishing)
        - the latency is less than the previous records now with the enhancing of the laptop and the less background work also the tps is high and the query plan time and the execution plan time is more higher than how it was before the optimization so its a relation when the latency less the tps high and the opposite same with the execution time and the total plan when lateny less they are high also with the tps and the execution time and the total plan are the same in progress high or low.

- Query_13:-
    - **The attributes:-**

        - title - choosing the film title for enhacing the performance of searching.using btree for indexing.
    - **Statistics of plan-time, execution-time, & index usage:-**
        - **Before:**
            - query plan time = 1614.8444290000043
            - query execution time = 160.5452349999996
        - **After:**
            - query plan time = 3071.5853139999967
            - query execution time = 295.31406100000095
            - index usage:-
                - how many times index used =
                  f_t = 10000
                - how many rows were fetched =
                  f_t = 10000
    - **TPS & Average latency:-**
        - **Before:**
            - latency average = 0.292 ms
            - tps = 3428.495137(including connection establishing)
              tps = 3431.198995(excluding connections establishing)
        - **After:**
            - latency average = 0.310 ms
            - tps = 3223.036820(including connection establishing)
              tps = 3225.533245(excluding connections establishing)
            - latency here is higher a bit and the tps is less for the same previous reasons.

- **Query_14:-**
    - **The attributes:-**

        - first_name - the query doesn't need an index but using an index on the first_name and last_name will optimize the performance.
        - last_name - he query doesn't need an index but using an index on the first_name and last_name will optimize the performance.
    - **Statistics of plan-time, execution-time, & index usage:-**
        - **Before:**
            - query plan time = 213.1614969999997
            - query execution time = 735.7608269999961
        - **After:**
            - query plan time = 702.378650000001
            - query execution time = 2193.592946999986
            - index usage:-
                - how many times index used =
                  act_first = 0
                  act_last = 0
                - how many rows were fetched =
                  act_first = 0
                  act_last = 0
    - **TPS & Average latency:-**
        - **Before:**
            - latency average = 0.237 ms
            - tps = 4218.210181(including connection establishing)
              tps = 4221.825558(excluding connections establishing)
        - **After:**
            - latency average = 0.209 ms
            - tps = 4788.645308(including connection establishing)
              tps = 4793.928648(excluding connections establishing)

        - latency is less and the pts is higher means the performance is optimized.

- **Query_15:-**
    - **The attributes:-**
        - active - the only attribute to be indexed using the hash for direct fast search and better performance.
    - **Statistics of plan-time, execution-time, & index usage:-**
        - **Before:**
            - query plan time = 391.00061599999935
            - query execution time = 181.5933709999991
        - **After:**
            - query plan time = 811.0736539999978
            - query execution time = 349.0668889999965
            - index usage:-
                - how many times index used =
                  cuss_acti = 10000
                - how many rows were fetched =
                  cuss_acti = 0
    - **TPS & Average latency:-**
        - **Before:**
            - latency average = 0.191 ms
            - tps = 5223.050712(including connection establishing)
              tps = 5228.828853(excluding connections establishing)
        - **After:**
            - latency average = 0.186 ms
            - tps = 5372.393612(including connection establishing)
              tps = 5380.301130(excluding connections establishing).

        - latency is less and the pts is higher means the performance is optimized.

- **Query_16:-**
    - **The attributes:-**
        - district - the only attribute that can be used and indexed by a btree indexing type for better performance.
    - **Statistics of plan-time, execution-time, & index usage:-**
        - **Before:**
            - query plan time = 1682.0101279999976
            - query execution time = 1487.0767580000038
        - **After:**
            - query plan time = 3270.108076999999
            - query execution time = 2980.051892999998
            - index usage:-
                - how many times index used =
                  add_dis = 10000
                - how many rows were fetched =
                  add_dis = 0
    - **TPS & Average latency:-**
        - **Before:**
            - latency average = 0.491 ms
            - tps = 2035.071158(including connection establishing)
              tps = 2035.819817(excluding connections establishing)
        - **After:**
            - latency average = 0.405 ms
            - tps = 2466.931075(including connection establishing)
              tps = 2468.466151(excluding connections establishing)

        - latency is less and the pts is higher means the performance is optimized.

- **Query_17:-**
    - **The attributes:-**
        - language_id - using btree for better performance and choosing the id for fast searching.
        - title - for finding the title faster using the btree indexing type.
    - **Statistics of plan-time, execution-time, & index usage:-**
        - **Before:**
            - query plan time = 410.07169299999885
            - query execution time = 2126.4497130000054
        - **After:**
            - query plan time = 1898.7749969999911
            - query execution time = 11295.563929999978
            - index usage:-
                - how many times index used =
                  fi_ti = 0
                  lan_id = 0
                - how many rows were fetched =
                  fi_ti = 0
                  lan_id = 0
    - **TPS & Average latency:-**
        - **Before:**
            - latency average = 0.411 ms
            - tps = 2434.035790(including connection establishing)
              tps = 2435.146060(excluding connections establishing)
        - **After:**

- latency average = 0.546 ms
- tps = 1831.142199(including connection establishing)
  tps = 1831.828058(excluding connections establishing)
- last_name - he query doesn't need an index but using an index on the first_name and last_name will optimize the performance.
