Advanced Databases
INZ000109P
Project

Group: A2
Mustafa Tayyip BAYRAM 257639
Furkan ÖCALAN 257638

➔ CREATED PARTITIONS

- **PARTITION BY HASH(store_id) PARTITIONS 8;**

- **PARTITION BY HASH(brand_id) PARTITIONS 4;**

- **PARTITION BY RANGE (order_date)**

```
( PARTITION sales_q1_2017 VALUES LESS THAN (TO_DATE('1/1/2017' ,
'MM/DD/YYYY'))
 , PARTITION sales_q2_2017 VALUES LESS THAN (TO_DATE('2/2/2017',
'MM/DD/YYYY' ))
 , PARTITION sales_q3_2017 VALUES LESS THAN (TO_DATE('4/4/2017',
'MM/DD/YYYY'))
 , PARTITION sales_q4_2017 VALUES LESS THAN (TO_DATE('6/6/2017',
'MM/DD/YYYY'))
 , PARTITION sales_q5_2017 VALUES LESS THAN (TO_DATE('8/8/2017',
'MM/DD/YYYY'))
 , PARTITION sales_q6_2017 VALUES LESS THAN (TO_DATE('10/10/2017',
'MM/DD/YYYY'))
 , PARTITION sales_q7_2017 VALUES LESS THAN (TO_DATE('12/14/2017',
'MM/DD/YYYY'))
 );
```

- **PARTITION BY RANGE (salary)**

```
 ( PARTITION salary_q1 VALUES LESS THAN (6500.00)
 , PARTITION salary_q2 VALUES LESS THAN (13000.00)
 , PARTITION salary_q3 VALUES LESS THAN (18500.00)
 , PARTITION salary_q4 VALUES LESS THAN ( 25000.00 )
 );
```

- **PARTITION BY RANGE (list_price)(**

```
PARTITION list_price1 VALUES LESS THAN (9000)
 , PARTITION list_price2 VALUES LESS THAN (19000)
 , PARTITION list_price3 VALUES LESS THAN (29000)
 , PARTITION list_price4 VALUES LESS THAN (39000)
 , PARTITION list_price5 VALUES LESS THAN (49000)
 , PARTITION list_price6 VALUES LESS THAN (59000)
 , PARTITION list_price7 VALUES LESS THAN (69000)
 , PARTITION list_price8 VALUES LESS THAN (79000)
 , PARTITION list_price9 VALUES LESS THAN (89000)
 , PARTITION list_price10 VALUES LESS THAN ( MAXVALUE )
```

);

- **PARTITION BY HASH(customer_id) PARTITIONS 8;**

- FIRST QUERY

```
SELECT product_id,brand_name ,product_name, model_year, list_price, category_name FROM
(
    (SELECT * FROM
        (SELECT * FROM
          ( SELECT * from MUTABAY.products prod_outer
            where 1 = (
                SELECT COUNT(Distinct list_price)
                FROM MUTABAY.products prod_inner
                WHERE prod_outer.brand_id = prod_inner.brand_id
                AND prod_outer.list_price < prod_inner.list_price
                )
          ) prod
          FULL OUTER JOIN
          MUTABAY.brands brands on brands.brand_id = prod.brand_id
        ) prod_brand
        FULL OUTER JOIN
        MUTABAY.categories categories on categories.category_id = prod_brand.category_id
    )prod_brand_cat
)
where list_price > 980000 AND model_year < 2020

GROUP BY product_id,brand_name ,product_name, model_year, list_price, category_name
ORDER BY product_id ASC;
```

EXECUTION TIME COMPARISON

TRYING	BEFORE PARTITIONS	AFTER PARTITIONS
1	**1.779**	**1.318**
2	**1.73**	**0.918**
3	**1.822**	**0.944**
4	**1.751**	**0.921**
5	**1.73**	**0.918**
6	**1.74**	**0.942**
7	**1.724**	**0.946**
8	**1.744**	**0.912**
9	**1.742**	**0.893**
10	**1.743**	**0.94**
MAX	**1.822**	**1.318**
MIN	**1.73**	**0.893**

AVERAGE	1.7505	0.9652

WITHOUT ANY IMPROVEMENTS AVERAGE TIME = 1.7505
PARTITIONED AVG. TIME = 0.9652
INDEXED AVG. TIME = 0.410

Index improvements is more effective way to optimizing on this query like as with most.

- **SECOND QUERY**

```
SELECT first_name,last_name, active, salary, stores.store_name, stores.city, stores.state
FROM
(
   SELECT staffs.*,avg(salary) over (partition by store_id) as avgSalary
   from MUTABAY.staffs staffs
)staffs
FULL OUTER JOIN MUTABAY.stores stores
ON staffs.store_id=stores.store_id
FULL OUTER JOIN MUTABAY.orders orders
ON stores.store_id = orders.store_id
FULL OUTER JOIN MUTABAY.order_items order_items
ON orders.order_id = order_items.order_id
FULL OUTER JOIN MUTABAY.products products
ON order_items.product_id = products.product_id
WHERE staffs.salary < staffs.avgsalary or order_items.discount > 0.05 OR customer_id > 1500
GROUP BY store_name, first_name, salary, city, state, last_name, active
having (avg(staffs.salary) > 1000 OR state IS NOT NULL) OR (city = 'Aberdeen' AND active = 1)
ORDER BY store_name asc;
```

EXECUTION TIME COMPARISON

TRYING	BEFORE PARTITIONS	AFTER PARTITIONS
1	1.67	1.352
2	1.208	0.863
3	1.188	0.565
4	1.194	0.569
5	1.217	0.588
6	1.164	0.587
7	1.314	0.573
8	1.357	0.582
9	3.513	0.603
10	3.329	0.607
MAX	3.513	1.352
MIN	1.164	0.565
AVERAGE	1.7514	0.6889

WITHOUT ANY IMPROVEMENTS AVERAGE TIME = 1.7514
PARTITIONED AVG. TIME = 0.6889
INDEXED AVG. TIME = 1.447

Partition improvements is more effective way to optimizing on this query like as with most.

- **THIRD QUERY**

```
SELECT products.product_id, products.product_name, products.list_price,
      orders.order_date, orders.required_date, orders.order_status,
      categories.category_name, brands.brand_name, quantity_id,
      discount,COUNT(quantity_id) quantity_count, (quantity_id * discount * products.list_price) total
FROM MUTABAY.order_items order_items
   full outer join MUTABAY.orders orders on
      (orders.order_id = order_items.order_id)
   full outer join MUTABAY.products products on
      (products.product_id = order_items.product_id)
   full outer join MUTABAY.brands brands on
      (brands.brand_id = products.brand_id)
   full outer join MUTABAY.categories categories on
      (categories.category_id = products.category_id)
   full outer join MUTABAY.staffs staffs on
      (staffs.staff_id = orders.staff_id)
   WHERE (shipped_date - order_date ) > 2 OR
      (shipped_date - order_date ) = 0 OR
      (shipped_date - order_date ) < 0
GROUP BY products.product_id, products.product_name, products.list_price,
     orders.order_date, orders.required_date, orders.order_status,
     categories.category_name, brands.brand_name, quantity_id,
     discount
having AVG(list_price) > 10000
Order by order_status;
```

EXECUTION TIME COMPARISON

TRYING	BEFORE PARTITIONS	AFTER PARTITIONS
1	**0.713**	**1.083**
2	**0.707**	**0.722**
3	**0.722**	**0.661**
4	**0.952**	**0.738**
5	**0.912**	**0.672**
6	**0.772**	**0.769**
7	**0.822**	**0.639**
8	**0.795**	**0.735**
9	**0.714**	**0.74**
10	**0.926**	**0.695**
MAX	**0.952**	**1.083**
MIN	**0.707**	**0.639**
AVERAGE	**0.7985**	**0,7454**

WITHOUT ANY IMPROVEMENTS AVERAGE TIME = 0.7985
PARTITIONED AVG. TIME = 0.7454

INDEXED AVG. TIME = 0.549

Index improvements is more effective way to optimizing on this query like as with most.

- **FOURTH QUERY**

```
update MUTABAY.products products
set model_year =
                (
        select distinct(product_id) as total_product
        from MUTABAY.order_items order_items
          full outer join MUTABAY.orders orders on
            (orders.order_id = order_items.order_id)
          full outer join MUTABAY.stores stores on
            (orders.store_id = stores.store_id)
          where stores.store_id =
            (
            select store_id from MUTABAY.stocks
              full outer join MUTABAY.products products on
                (stocks.product_id = products.product_id)
                  where ((model_year between 2020 and 1958) or (ROUND(list_price) < 990.000))
                      or UPPER ( SUBSTR(product_name,2,3 ) )LIKE 'D%'
                  fetch first 1 rows only
            )
        fetch first 1 rows only
                );
```

EXECUTION TIME COMPARISON

TRYING	BEFORE PARTITIONS	AFTER PARTITIONS
1	1.255	0.63
2	0.641	0.76
3	0.628	0.394
4	0.731	0.945
5	0.809	0.593
6	0.618	0.598
7	1.118	0.595
8	0.62	0.757
9	0.873	0.407
10	1.064	0.618
MAX	1.255	0.945
MIN	0.62	0.394
AVERAGE	0.8357	0.6297

WITHOUT ANY IMPROVEMENTS AVERAGE TIME = 0.8357
PARTITIONED AVG. TIME = 0.6297
INDEXED AVG. TIME = 0726

Partition improvements is more effective way to optimizing on this query like as with most. But almost same.

- **FIFTH QUERY**

```
update MUTABAY.order_items set quantity_id =
(
  select quantity_id from MUTABAY.products products
  full outer join MUTABAY.order_items order_items on order_items.product_id = products.product_id
  full outer join MUTABAY.stocks stocks on stocks.product_id = products.product_id
  full outer join MUTABAY.orders orders on order_items.order_id=orders.order_id
  full outer join MUTABAY.customers customers on orders.customer_id=customers.customer_id
  full outer join MUTABAY.stores stores on orders.store_id=stores.store_id
  full outer join MUTABAY.staffs staffs on orders.staff_id=staffs.staff_id
  where products.product_id in
    (
      Select product_id from MUTABAY.order_items where order_id in
      (
        Select order_id from MUTABAY.orders
        WHERE
        (order_status = 1 AND ( shipped_date - required_date = 1 ))
        OR
        (order_status = 2 AND (shipped_date - required_date = 0 ))
      )
    )fetch next 1 rows only
);
```

EXECUTION TIME COMPARISON

TRYING	BEFORE PARTITIONS	AFTER PARTITIONS
1	**1.825**	**1.504**
2	**2.068**	**1.372**
3	**2.059**	**1.404**
4	**1.3**	**1.443**
5	**1.447**	**1.612**
6	**1.289**	**1.022**
7	**1.443**	**1.356**
8	**1.45**	**1.331**
9	**1.723**	**1.341**
10	**1.318**	**1.534**
MAX	**2.068**	**1.612**
MIN	**1.3**	**1.022**
AVERAGE	**1.5922**	**1.3919**

WITHOUT ANY IMPROVEMENTS AVERAGE TIME = 1.5922

PARTITIONED AVG. TIME = 1.3919
INDEXED AVG. TIME = 1.413

Partition improvements is more effective way to optimizing on this query like as with most. But almost same

- **SIXTH QUERY**

```
UPDATE MUTABAY.stores
SET MUTABAY.stores.store_name = (
   SELECT store_name FROM MUTABAY.stores stores
   INNER JOIN
   (
      SELECT order_i.staff_id, first_name, last_name, phone, email, order_i.store_id, manager_id, active,
salary ,
      order_i.item_id ,order_i.product_id ,order_i.quantity_id ,order_i.discount ,order_i.customer_id
,order_i.order_status ,
      order_i.order_date ,order_i.required_date ,order_i.shipped_date
      FROM MUTABAY.staffs staffs
      FULL OUTER JOIN
      (
         SELECT orders.order_id, item_id, product_id, quantity_id, discount, orders.customer_id,
orders.order_status,
               orders.order_date, orders.required_date, orders.shipped_date, orders.store_id, orders.staff_id
         FROM MUTABAY.order_items order_items
         FULL OUTER JOIN MUTABAY.orders orders
         ON orders.order_id = order_items.order_id
         WHERE ORDERS.ORDER_ID IN (SELECT ORDER_ID FROM MUTABAY.ORDER_ITEMS WHERE
DISCOUNT > (SELECT AVG(DISCOUNT) FROM MUTABAY.ORDER_ITEMS))
         OR
         (order_status = 2)
      ) order_i
      ON order_i.staff_id = staffs.staff_id
      WHERE (discount > 0.48 AND discount < 0.05) AND salary > 5000
      OR
      (active = 1 AND discount = 0.4)
   )order_i_staff
   ON order_i_staff.store_id = stores.store_id
   WHERE street='1 Fremont Point' or STATE IS NOT NULL
   fetch first 1 rows only
);
```

EXECUTION TIME COMPARISON

TRYING	BEFORE PARTITIONS	AFTER PARTITIONS
1	5.015	4.889
2	5.204	4.975
3	4.97	4.844
4	5.4	5.359
5	4.924	4.942
6	4.922	4.85
7	4.919	4.449
8	4.917	4.873
9	5.131	4.921
10	4.824	4.725
MAX	5.204	5.359
MIN	4.824	4.449
AVERAGE	5.0226	4.8827

WITHOUT ANY IMPROVEMENTS AVERAGE TIME = 5.0226
PARTITIONED AVG. TIME = 4.8827
INDEXED AVG. TIME = 20.1

Partition improvements is more effective way to optimizing on this query like as with most.