Σχεδίαση Βάσεων Δεδομένων και Κατανεμημένες ΒΔ

2 η Εργασία στη Σχεδίαση ΒΔ 2021-2022

Σιγάλας Σπυρίδων 21991
Παππάς Στέφανος 218131
Μαυροπουλος Ανδρεας 217129

Ερωτημα 1ο:

1:

OPERATION	COST	BYTES	CPU_COST	IO_COST
SELECT STATEMENT	322	62	29354285	321
HASH JOIN	322	62	29354285	321
MERGE JOIN	262	3321	21403184	261
TABLE ACCESS	3	15	63458	3
BUFFER	259	27040	21339726	258
TABLE ACCESS	259	27040	21339726	258
TABLE ACCESS	60	86184	6928551	60

2:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	322
HASH JOIN			1	322
Access Predicates				
AND				
O.CUSTOMER_ID=C.CUSTOMER_ID				
O.ID=P.IDENTIFIER				
MERGE JOIN		CARTESIAN	81	262
TABLE ACCESS	PRODUCTS	FULL	1	3
Filter Predicates				
AND				
P.SUBCATEGORY_REFERENCE=2042				
TO_NUMBER(P.LIST_PRICE)=41.24				
BUFFER		SORT	1040	259
TABLE ACCESS	CUSTOMERS	FULL	1040	259
Filter Predicates				
AND				
C.INCOME_LEVEL='high'				
C.MARITAL_STATUS='unknown'				
TABLE ACCESS	ORDERS	FULL	4104	60
Filter Predicates				
O.CHANNEL='Internet'				

Από ότι βλέπουμε η σειρά εκτελεσεις είναι πρώτον Select,Hash Join, Merge Join, Buffer και τέλος Table access

3: Με ευρετήριο βλέπουμε της εξής αλλαγές

OPERATION	COST	BYTES	CPU_COST	IO_COST
SELECT STATEMENT	64	62	8010731	64
NESTED LOOPS	64	62	8010731	64
HASH JOIN	63	36	8002559	63
TABLE ACCESS	3	15	63458	3
TABLE ACCESS	60	86184	6928551	60
INDEX	1	26	8171	1

Προσθετοντας ευρετηρια:

Έχοντας προσθέσει ευρετήρια παρατηρούμε αρκετά μεγάλη αλλαγή αρχικά στο select statement οπού βλέπουμε μια διαφορά στο κόστος σχεδόν 5 φορές μεγαλύτερη, όπως και στο cpu cost. Αυτό

όμως έρχεται με ένα μειονέκτημα, το μέγεθος παρατηρούμε ότι είναι σχεδόν 3 φορές μεγαλύτερο. Επίσης μεγάλη αλλαγή βλέπουμε και στα hash joins. Τέλος έχοντας βάλει ευρετήρια βλέπουμε ότι το κόστος του I/O όπως είναι λογικό να μικραίνει.

Ερωτημα 2ο:

1:Το εκτιμωμενο συνολικο κοστος ειναι 3 στην περιπτωση του BUFFER

Τα κόστοι των CPU_COST και IO_COST:

OPERATION	CPU_COST	IO_COST
SELECT STATEMENT	29291197	321
HASH JOIN	29291197	321
MERGE JOIN	21244196	261
TABLE ACCESS	21174029	258
BUFFER	70167	3
TABLE ACCESS	70167	3
TABLE ACCESS	6928551	60

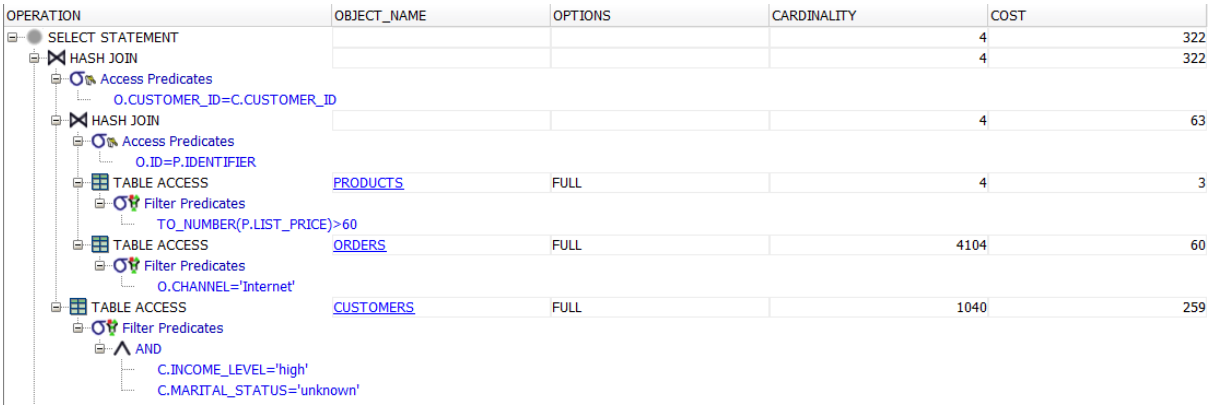
Η πιο χρονοβόρα ενέργεια είναι η SELECT STATEMENT και η HASH JOIN.

OPERATIONS	COST	BYTES	CPU_COST	IO_COST
SELECT STATEMENT	322	64	29291197	321
HASH JOIN	322	64	29291197	321

MERGE JOIN	262	43	21244196	261
TABLE ACCESS	259	32	21174029	258
BUFFER	3	44	70167	3
TABLE ACCESS	3	44	70167	3
TABLE ACCESS	60	108843	6928551	60

OPERATION	COST	BYTES	CPU_COST	IO_COST
SELECT STATEMENT	322	64	29291197	321
HASH JOIN	322	64	29291197	321
MERGE JOIN	262	43	21244196	261
TABLE ACCESS	259	32	21174029	258
BUFFER	3	44	70167	3
TABLE ACCESS	3	44	70167	3
TABLE ACCESS	60	108843	6928551	60

2:Η σειρα εκτελεσης απο οτι βλεπουμε στην φωτογραφια απο κατω ειναι SELECT. HASH. HASH_JOIN. TABLE_ACCESS



Ερωτημα 3ο:

i)

```
-- Update order items , add column days to process
ALTER TABLE ORDER_ITEMS
ADD DAYS_TO_PROCESS NUMBER(3);
ALTER TABLE ORDER_ITEMS
ADD ORDER_FINISHED DATE;
-- Insert order finished data from orders table
INSERT INTO ORDER_ITEMS(ORDER_FINISHED)
SELECT ORDER_FINISHED FROM ORDERS
    WHERE ORDERS.ROWNUM = ORDER_ITEMS.ROWNUM;
-- Finally update Days to process column
UPDATE ORDER_ITEMS
SET DAYS_TO_PROCESS = ROUND(ORDER_FINISHED - ORDER_DATE);
```

ii)

```
--  2
-- Insert Into Order Items 2 more columns
ALTER TABLE ORDER_ITEMS
ADD TOTAL_EARNED FLOAT;
-- TOTAL_EARNED = (LIST_PRICE - COST) - ((0.0001 * LIST_PRICE) * (DAYS_TO_PROCESS - 10))
UPDATE ORDER_ITEMS
SET ORDER_ITEMS.TOTAL_EARNED =
    (ORDER_ITEMS.LIST_PRICE - ORDER_ITEMS.COST) - ((0.0001 * ORDER_ITEMS.LIST_PRICE) * (ORDER_ITEMS.DAYS_TO_PROCESS - 10))
WHERE ORDER_ITEMS.ROWNUM = LIST_PRICE.ROWNUM;
```

iii)

```
--3
--  Create a new profit table
CREATE TABLE PROFIT (
  ORDER_ID NUMBER(10),
  CUSTOMER_ID NUMBER(10),
  CHANNEL VARCHAR2(10),
  AMMOUNT FLOAT
);
-- Create new deficit table
CREATE TABLE DEFICIT (
  ORDER_ID NUMBER(10),
  CUSTOMER_ID NUMBER(10),
  CHANNEL VARCHAR2(10),
  AMMOUNT FLOAT
);
```

```
-- Declare the cursor
DECLARE CURSOR pay_cursor IS select * from ORDER_ITEMS;
BEGIN
    -- OPEN CURSOR
    OPEN pay_cursor;
    -- FETCH CURSOR
LOOP
    FETCH pay_cursor INTO :TOTAL_EARNED;
    EXIT WHEN pay_cursor%NOTFOUND;
    IF pay_cursor > 0 THEN
        -- INSERT INTO PROFIT
        INSERT INTO PROFIT (ORDER_ID,CUSTOMER_ID,CHANNEL,AMMOUNT)
        VALUES (ORDER_ITEMS.ORDER_ID,ORDER_ITEMS.CUSTOMER_ID,ORDER_ITEMS.CHANNEL,ORDER_ITEMS.TOTAL_EARNED);
    ELSE
        -- INSERT INTO DEFICIT
        INSERT INTO DEFICIT (ORDER_ID,CUSTOMER_ID,CHANNEL,AMMOUNT)
        VALUES (ORDER_ITEMS.ORDER_ID,ORDER_ITEMS.CUSTOMER_ID,ORDER_ITEMS.CHANNEL,ORDER_ITEMS.TOTAL_EARNED);
    END IF;
END LOOP;
    -- Close the cursor
    CLOSE pay_cursor;
end;
```

vi)

```
-- 4
SELECT CUSTOMERS.CUSTOMER_ID,PROFIT.CUSTOMER_ID,DEFICIT.CUSTOMER_ID,CUSTOMERS.GENDER,PROFIT.AMMOUNT,DEFICIT.AMMOUNT
      FROM CUSTOMERS CUSTOMERS
CROSS JOIN PROFIT PROFIT,DEFICIT DEFICIT
ON CUSTOMERS.CUSTOMER_ID = PROFIT.PROFIT_ID = DEFICIT.DEFICIT_ID
WHERE CUSTOMERS.GENDER = 'MALE';

SELECT CUSTOMERS.CUSTOMER_ID,PROFIT.CUSTOMER_ID,DEFICIT.CUSTOMER_ID,CUSTOMERS.GENDER,PROFIT.AMMOUNT,DEFICIT.AMMOUNT
      FROM CUSTOMERS CUSTOMERS
CROSS JOIN PROFIT PROFIT,DEFICIT DEFICIT
ON CUSTOMERS.CUSTOMER_ID = PROFIT.PROFIT_ID = DEFICIT.DEFICIT_ID
WHERE CUSTOMERS.GENDER = 'FEMALE';
```

v)

```
-- 5
SELECT ORDERS.ID , PROFIT.ORDER_ID,DEFICIT.ORDER_ID,PROFIT.AMMOUNT,DEFICIT.AMMOUNT
FROM ORDERS ORDERS
CROSS JOIN PROFIT PROFIT ,DEFICIT DEFICIT
WHERE ORDERS.ID = PROFIT.ORDER_ID = DEFICIT.ORDER_ID;
```