CSC553

Raymond Elward

Assignment 4

2/19/2012

2A)

Q1… no indexes:

Estimated Cost = 71598.429688

with suggested index:

Estimated Cost = 26986.078125

Q2 no indexes:

Estimated Cost = 71670.421875

with suggested index:

Estimated Cost = 27631.779297

Q3 no indexes:

Estimated Cost = 71670.640625

with suggested index:

Estimated Cost = 27669.808594

Yes, they all use the index and save a lot on estimated cost. See above for exact numbers of saved cost on indexes.

2b)

(d\_year=1993) = 0.14

(lo\_discount between 1 and 3) = 0.27

(lo\_quantity < 25) = 0.48

yes, It suggests that we should reorder the index values to get the less selections first. So we should change the order to those entries with the lowest selectivity first.

The new index declaration is : CREATE INDEX Q123\_INDEX1 ON LINEORDER (lo\_discount, lo\_quantity, lo\_orderdate, lo\_extendedprice)

The new Estimated Cost = 2318.881592. 10 times lower than the original index.

2c)

I created the index compressed with: CREATE INDEX Q123\_INDEX1 ON LINEORDER (lo\_discount, lo\_quantity, lo\_orderdate, lo\_extendedprice) COMPRESS YES

The new Estimated Cost = 3314.249023. A little bit more expensive than the non-compressed version. I wouldn’t have guessed that. The main downside to compressing indexes is on updates you have uncompress the block change it then recompress it to get anything changed, and this can get very expensive.

2d)

Indexed with CREATE INDEX Q123\_INDEX1 ON LINEORDER (lo\_extendedprice, lo\_orderdate, lo\_discount, lo\_quantity) CLUSTER

Q1 new estimated cost = 8920.618164

Q2 new estimated cost = 9586.207031

Q3 new estimated cost = 9606.917969

Yes it did improve performance on the queries from the original index. Clustering indexes just have more overhead at insert time to reorganize the clusters.

now with my index we create: CREATE INDEX Q123\_INDEX1 ON LINEORDER (lo\_discount, lo\_quantity, lo\_orderdate, lo\_extendedprice) CLUSTER

Q1 new estimated cost = 3429.600586

Q2 new estimated cost = 3310.259766

Q3 new estimated cost = 3271.632080

Yes it did improve performance over the given clustering index. But still the oringinal non-clustered version of my index creates the best performance for these 3 queries out of all the ones I’ve tried in question 2.

3)

a)

T1 T2 T3

R(X)

R(X)

W(X)

R(X)

W(X)

Conflict Serializable? No. You can’t swap to achieve serializability. The circle of dependency is T3 –R/W-> T1 –R/W -> T3.

equivalent serial schedule? n/a

B)

T1 T2 T3

R(X)

R(X)

W(X)

R(X)

R(X)

Conflict Serializable? No. You can’t swap to achieve serializability. The circle of dependency is T1 –R/W-> T3 –R/W -> T1.

equivalent serial schedule? n/a

C)

T1 T2 T3

R(X)

R(X)

W(X)

R(X)

W(X)

Conflict Serializable? Yes. You can swap to achieve conflict serializability. Just need to swap the R(X) in T2 with the R(X) in T3.

equivalent serial schedule? <T2, T3, T1>

D)

T1 T2 T3

R(X)

R(X)

R(X)

W(X)

W(X)

Conflict Serializable? No. You can’t swap to achieve serializability. The circle of dependency is T3 –R/W-> T1 –R/W -> T3. But it is view serializable that can be achieved through blind writes.

equivalent serial schedule? n/a