DBMS Tutorial 13.02.2019

Topics

Index Join and Cost of Selections

Index Join

Assumptions : R has an index on the join attribute A. Index is in memory. **R and S are contiguous.**

Algorithm:

1. Read S
    a. For each tuple in S probe index to find if R.A = S.A exists.
        i. If it does, fetch it from disk, and Join
        ii. Each tuple from S will join with T(R)/V(R,A) tuples of R (expected result of a selection).
        iii. Assuming the index to be non-clustering, T(R)/V(R,A) records can be fetched in as many IOs
        iv. Assuming the index to be clustering: T(R)/V(R,A) records can be fetched in B(R)/V(R,A) IOs

Cost = Reading S + retrieving tuples of R whenever an index entry exists

= B(S) + T(S) x T(R)/V(R,A) for non-clustering index

= B(S) + T(S) x B(R)/V(R,A) for clustering index

Index Join

Assumptions : R has an index on the join attribute A. Index is in memory. **R and S are contiguous.**

Algorithm:

1. Read S
    a. For each tuple in S probe index to find if R.A = S.A exists.
        i. If it does, fetch it from disk, and Join
        ii. Each tuple from S will join with T(R)/V(R,A) tuples of R (expected result of a selection).
        iii. Assuming the index to be non-clustering, T(R)/V(R,A) records can be fetched in as many IOs
        iv. Assuming the index to be clustering: T(R)/V(R,A) records can be fetched in B(R)/V(R,A) IOs

Cost = Reading S + retrieving tuples of R whenever an index entry exists

= B(S) + T(S) x T(R)/V(R,A) for non-clustering index

= B(S) + T(S) x B(R)/V(R,A) for clustering index

In case of a join, when one of the relations has an index on join attribute (here R), we read the other relation completely(here S) and for each value of join attribute in S, we probe the index on R to find if such a value exists in R. Expected result from R is similar to a selection query on R.

Index Join Example

Assumptions : R has an index on the join attribute A. Index is in memory. S is contiguous.

Examples:

Case 1. R.A is primary key and S.A is foreign key

Case 2. V(R,A) = 100 with non-clustering index

Case 3. V(R,A) = 100 with clustering index

Remember:
Relations R and S
T(R) = 60000
T(S) = 2000
Block size = 4000 bytes

S(R) = 100 bytes
S(S) = 500 bytes

Main Memory = 126 Buffers

Index Join Example

Assumptions : R has an index on the join attribute A. Index is in memory. S is contiguous. Cost in terms of IOs

Examples:

Case 1. R.A is primary key and R.S is foreign key, ie V(R,A) = T(R)

Number of possible matches for each tuple of S =T(R)/V(R,A) = 1

Cost = 250 + 2000x1

Case 2. V(R,A) = 100 with non-clustering index

Number of possible matches for each S.A = T(R)/V(R,A) = 60000/100

Cost = 250 + 2000x600

Case 3. V(R,A) = 100 with clustering index

Number of possible matches for each S.A = T(R)/V(R,A) = 60000/100
but now they can be found in B(R)/V(R,A) = 1500/100 blocks!

Cost = 250 + 2000x15

Notes

Probing Index for R.A = ‘10’

1. Probing Index means checking the index for a given key. For this the index needs to be in memory. If it’s not kept in memory probing the index will be an IO cost.
2. Expected number of tuples for such a query is T(R)/V(R,A)
   a. If the index is clustering we can get all the tuples (using the pointers from the index) in B(R)/V(R,A) IOs.
   b. If the index is non-clustering we can get all the tuples (using the pointers from the index) in T(R)/V(R,A) IOs.

Example from DSCB

Suppose B(R) = 1000, and T(R) = 20,000. (This means R has 20,000 tuples, packed at most 20 to a block). Let A be one of the attributes of R, suppose there is an index on A, and consider the operation $\sigma_{A=0}(R)$. Here are some possible situations and the worst-case number of disk I/O 's required.

We shall ignore the cost of accessing the index blocks in all cases.

a) R is clustered but we do not use the index. Cost = 1000 disk I/O 's.
b) If R is not clustered and we do not use the index. Cost = 20000 disk I/O 's
c) If V(R, A) = 100 and the index is clustering. Cost = 1000/100 = 10 disk I/O 's, (plus whatever is needed to access the index).
d) If V(R,A) = 10 and the index is non-clustering.Cost = 20,000/10 = 2000 disk I/O 's. (**Notice that this cost is higher than scanning the entire relation R, if R is clustered but the index is not**.)
e) If V(R, A) = 20,000, i.e., A is a key, then the index-based algorithm takes 1 disk I/O (plus whatever is needed to access the index, regardless of whether the index is clustering or not.)

Exercise 1

T(R) = 10000, B(R) = 500, V(R,A) =30. Calculate

a) the number of tuples expected in the result, and

b) the IOs it will take to retrieve the result, for the query $\sigma_{A=10}(R)$ in the following cases :

1. R has a clustering index on A.
2. R has a non-clustering index on A
3. R has no index on A but is a clustered/contiguous relation.
4. R is non-contiguous.

Exercise 1 Ans

T(R) = 10000, B(R) = 500, V(R,A) =30. Calculate

a) the number of tuples expected in the result, and

b) the IOs it will take to retrieve the result, for the query $\sigma_{A=10}(R)$ in the following cases:

1. R has a clustering index on A.
   a. Tuples: T(R)/V(R,A) = 10000/30
   b. IOs: B(R)/V(R,A) = 500/30
2. R has a non-clustering index on A
   a. Tuples: T(R)/V(R,A) = 10000/30
   b. IOs: T(R)/V(R,A) = 10000/30
3. R has no index on A but is a clustered/contiguous relation
   a. Tuples: T(R)/V(R,A) = 10000/30
   b. IOs: B(R) = 500
4. R is non-contiguous.
   a. Tuples: T(R)/V(R,A) =10000/30
   b. IOs: T(R) = 10000

Exercise 2

T(R) = 10000, B(R) = 400, V(R,A) =50, V(R,B)=30, V(R,C)=100. R has a clustering index on A and non-clustering index on B.

Calculate the cost in terms of a) the number of tuples expected in the result and b) the IOs it will take to retrieve the result in the following cases (assume index to be in memory) :

1. $\sigma_{A<10}(R)$
2. $\sigma_{B=20}(R)$
3. $\sigma_{C=40}(R)$

What would be a cost effective plan to execute this query:

4. $\sigma_{A=10 \text{ AND } B=4 \text{ AND } C=6}(R)$

Exercise 2 Ans

T(R) = 10000, B(R) = 400, V(R,A) =500, V(R,B)=300, V(R,C)=100. R has a clustering index on A and non-clustering index on B.

Calculate the cost in terms of a) the number of tuples expected in the result and b) the IOs it will take to retrieve the result in the following cases (assume index to be in memory):

1. $\sigma_{A<10}(R)$
   a. Tuples: T(R)/3 = 10000/3
   b. IOs: B(R)/3 = 400/3
2. $\sigma_{B=20}(R)$
   a. Tuples: T(R)/V(R,B) = 10000/300
   b. IOs: T(R)/V(R,B) = 10000/300
3. $\sigma_{C=40}(R)$
   a. Tuples: T(R)/V(R,C) = 10000/100
   b. IOs: B(R) = 400

What would be a cost effective plan to execute this query:

4. $\sigma_{A=10 \text{ AND } B=4 \text{ AND } C=6}(R)$

4.

All the tuples satisfying A=10 (T(R)/500) can be brought memory in B(R)/V(R,A) = 400/500 IOs, ie in one IO. Therefore its better to use the index on A to get these tuples satisfying A=10 to memory. Once in memory we can look for the ones among these which satisfy the other two conditions, ie B=4 and C=6.

Exercise 3

This question is created from the DSCB example mentioned earlier.

Suppose B(R) = 1000, and T(R) = 20,000.

R is clustered and A is one of the attributes of R, and there is a non-clustering-index on A. What would be a good way to execute the query $\sigma_{A=0}(R)$.

1. Index probe costs 2 IO per probe
2. Index is in memory

Exercise 3 Ans

This question is created from the DSCB example mentioned earlier.

Suppose B(R) = 1000, and T(R) = 20,000.

R is clustered and A and B are the attributes of R with V(R,A) =10, and there is a non-clustering-index on A. What would be a good way to execute the query $\sigma_{A=0}(R)$.

There can be two ways of executing this query, simply reading the whole Relation ie B(R) IOs or using the index.

1. Index probe costs 2 IO per probe
   a. Reading the relation = B(R) = 1000 IOs
   b. Using Index : 2+ T(R)/V(R,A) = 2 + 2000 = 2002 IOs
2. Index is in memory
   a. Reading the relation = B(R) = 1000 IOs
   b. Using Index : T(R)/V(R,A) = 2000 IOs

⇒ Cheaper to read the whole relation than to use index in both scenarios.

Please repeat this question for attribute B, with V(R,B) = 100 and a clustering index on B and query: $\sigma_{B<10}(R)$.

Exercise 4

Relation R : B(R) = 10,000

Relation S : B(S) = 4000

Given that R and S are sorted and contain no duplicates for join attribute, what Join would you suggest. What would be the

Exercise 4 Ans

Relation R : B(R) = 10,000

Relation S : B(S) = 4000

Given that R and S are sorted and contain no duplicates for join attribute, what Join would you suggest. What would be the minimum memory requirement for such a join.

Since the question specifically states there are no duplicates we can do the join with even two buffers using merge join. (The number of duplicates can place a constraint on the number of buffers required for merge join).

Exercise 4

Estimate the number of tuples for:

$$R(a,b,c) \bowtie_{R.b=S.d \text{ AND } R.c=S.e} S(d,e,f)$$

with

$R(a,b,c)$	$S(d,e,f)$
$T(R) = 1000$	$T(S) = 2000$
$V(R,b) = 20$	$V(S,d) = 50$
$V(R,c) = 100$	$V(S,e) = 50$

We can think of this join as a natural join if we regard R.b and S.d as the same attribute and also regard R.c and S.e as the same attribute. Then using the same rules we for a natural join:

$$\text{No. of Tuples} = \frac{T(R)*T(S)}{\max\{V(R,b),V(S,d)\}*\max\{V(R,c),V(S,e)\}}$$

$$= \quad 1000*2000/(50*100) \quad = 400$$

Note: Size of each resultant tuple = S(R) + S(S)

Exercise 5

Consider the join $R(a,b,c) \bowtie S(b,c,d) \bowtie U(b,e)$,

$R(a,b,c)$	$S(b,c,d)$	$U(b,e)$
$T(R) = 1000$	$T(S) = 2000$	$T(U) = 5000$
$V(R,a) = 100$		
$V(R,b) = 20$	$V(S,b) = 50$	$V(U,b) = 200$
$V(R,c) = 200$	$V(S,c) = 100$	
	$V(S,d) = 400$	
		$V(U,e) = 500$

And the following sizes:
R.a = 10 bytes
R.b = S.b = U.b = 5 bytes
R.c = S.c = 20 bytes
S.d = 50 bytes
U.e = 22 bytes

What will be the size of the join result?

Exercise 5 Ans

1. The number of tuples = $\frac{\left(\frac{1000*2000}{50*200}\right) \times 5000}{200}$

   = 5000

2. Size of each tuple = S(R) + S(S) + S(U) - S(R.b) - S(R.c) - S(U.b)

   = 35 + 75 + 27 - 5 - 20 - 5

   = 107 bytes

Therefore size of result = 5000x107 bytes.