Week 7 Exercises

∧ >>

❖ Exercise: SIMC Query Cost

Consider a SIMC-indexed database with the following properties

- all pages are B = 8192 bytes
- tuple descriptors have m = 64 bits (= 8 bytes)
- total records r = 102,400,   records/page c = 100
- false match probability p_F = 1/1000
- answer set has 1000 tuples from 100 pages
- 90% of false matches occur on data pages with true match
- 10% of false matches are distributed 1 per page

Calculate the total number of pages read in answering the query.

❖ Exercise: Page-level SIMC Query Cost

Consider a SIMC-indexed database with the following properties

- all pages are B = 8192 bytes
- page descriptors have m = 4096 bits (= 512 bytes)
- total records r = 102,400,   records/page c = 100
- false match probability p_F = 1/1000
- answer set has 1000 tuples from 100 pages
- 90% of false matches occur on data pages with true match
- 10% of false matches are distributed 1 per page

Calculate the total number of pages read in answering the query.

❖ Exercise: Bit-sliced SIMC Query Cost

Consider a SIMC-indexed database with the following properties

- all pages are B = 8192 bytes
- r = 102,400, c = 100, b = 1024
- page descriptors have m = 4096 bits (= 512 bytes)
- bit-slices have b = 1024 bits (= 128 bytes)
- false match probability p_F = 1/1000
- query descriptor has k = 10 bits set to 1
- answer set has 1000 tuples from 100 pages
- 90% of false matches occur on data pages with true match
- 10% of false matches are distributed 1 per page

Calculate the total number of pages read in answering the query.

❖ Exercise: CATC Query Evaluation

Consider a SIMC-indexed database with the following properties

- all pages are B = 8192 bytes
- tuple descriptors have m = 64 bits (= 8 bytes)
- #attributes n = 4, so 4 × 16-bit codewords
- total records r = 102,400, records/page c = 100
- false match probability p_F = 1/1000
- answer set has 1000 tuples from 100 pages
- 90% of false matches occur on data pages with true match
- 10% of false matches are distributed 1 per page

Calculate the total number of pages read in answering the query.

<< ∧ >>

❖ Exercise: Nested Loop Join Cost

Compute the cost (# pages fetched) of $(S \bowtie E)$, where

- $r_S = 20{,}000$, $c_S = 20$, $b_S = 1000$
- $r_E = 160{,}000$, $c_S = 40$, $b_S = 4000$

for $N = 22, 202, 2002$ and different inner/outer combinations

❖ Exercise: Join Example Variation

If the query in the above example was:

```
select j.code, j.title, s.name
from   Student s
       join Enrolled e on (s.id=e.student)
       join Subject j on (e.subj=j.code)
```

how would this change the previous analysis?

What join combinations are there?

Assume 2000 subjects, with $c_J = 10$

How large would the intermediate tuples be? What assumptions?

Compute the cost (# pages fetched, # pages written) for $N = 202$

<< ^ >>

❖ Exercise: Index Nested Loop Join Cost

Consider executing *Join[i=j](S,T)* with the following parameters:

- $r_S = 1000$, $b_S = 50$, $r_T = 3000$, $b_T = 600$
- *S.i* is primary key, and *T* has index on *T.j*
- *T* is sorted on *T.j*, each *S* tuple joins with 2 *T* tuples
- DBMS has $N = 12$ buffers available for the join

Calculate the costs for evaluating the above join

- using block nested loop join
- using index nested loop join

$Cost_r$ = # pages read and $Cost_j$ = # join-condition checks

❖ Exercise: Sort-merge Join Cost

Consider executing *Join[i=j](S,T)* with the following parameters:

- $r_S = 1000$, $b_S = 50$, $r_T = 3000$, $b_T = 150$
- *S.i* is primary key, and *T* has index on *T.j*
- *T* is sorted on *T.j*, each *S* tuple joins with 2 *T* tuples
- DBMS has $N = 42$ buffers available for the join

Calculate the cost for evaluating the above join

- using sort-merge join
- compute #pages read/written
- compute #join-condition checks performed

<< ^ >>

❖ Exercise: Simple Hash Join Cost

Consider executing *Join[i=j](R,S)* with the following parameters:

- $r_R = 1000,\ b_R = 50,\ r_S = 3000,\ b_S = 150,\ c_{Res} = 30$
- *R.i* is primary key, each *R* tuple joins with 2 *S* tuples
- DBMS has *N = 43* buffers available for the join
- data + hash have uniform distribution

Calculate the cost for evaluating the above join

- using simple hash join
- compute #pages read/written
- compute #join-condition checks performed
- assume that hash table has *L=0.75* for each partition

❖ Exercise: Grace Hash Join Cost

Consider executing *Join[i=j](R,S)* with the following parameters:

- $r_R = 1000$, $b_R = 50$, $r_S = 3000$, $b_S = 150$, $c_{Res} = 30$
- *R.i* is primary key, each *R* tuple joins with 2 *S* tuples
- DBMS has $N = 43$ buffers available for the join
- data + hash have reasonably uniform distribution

Calculate the cost for evaluating the above join

- using Grace hash join
- compute #pages read/written
- compute #join-condition checks performed
- assume that no *R* partition is larger than 40 pages

❖ Exercise: Hybrid Hash Join Cost

Consider executing *Join[i=j](R,S)* with the following parameters:

- $r_R = 1000,\ b_R = 50,\ r_S = 3000,\ b_S = 150,\ c_{Res} = 30$
- *R.i* is primary key, each *R* tuple joins with 2 *S* tuples
- DBMS has $N = 42$ buffers available for the join
- data + hash have reasonably uniform distribution

Calculate the cost for evaluating the above join

- using hybrid hash join with various *k*
- compute #pages read/written
- compute #join-condition checks performed
- assume that no *R* partition is larger than 40 pages

❖ Exercise: Join Cost Comparison

Consider the cost of each of

- block nested loop join
- index nested loop join
- sort-merge join
- hash join
- grace hash join
- hybrid hash join

on *Join[i=j](R,S)* from the previous exercises.

Is any one algorithm overall better than the others?

❖ Exercise: Outer Join?

Join discussion was all in terms of theta inner-join.

How would the algorithms adapt to outer join?

Consider the following ...

```
select *
from   R left outer join S on (R.i = S.j)

select *
from   R right outer join S on (R.i = S.j)

select *
from   R full outer join S on (R.i = S.j)
```

Produced: 30 Mar 2021