>>

Week 8 Exercises

^ >>

❖ Exercise: SIMC Signatures

Consider a signature-indexed table with the following properties

- $n = 3$, $r = 10000$, $c = 20$, $pF = 0.01$

Compute m (bits per signature/codeword) and k (bits set in codewords)

Show examples of codewords and signatures produced for this scheme.

Reminder:

$$k = 1/\log_e 2 \,.\, \log_e (1/p_F)$$

$$m = (1/\log_e 2)^2 \,.\, n \,.\, \log_e (1/p_F)$$

<< ^ >>

❖ Exercise: Page-level SIMC Signatures

Consider a signature-indexed table with the following properties

- $n = 3$, $r = 10000$, $c = 20$, $pF = 0.01$

Compute m (bits per signature/codeword) and k (bits set in codewords)

Show examples of codewords and signatures produced for this scheme.

Reminder:

$$k = 1/\log_e 2 \,.\, \log_e (1/p_F)$$

$$m = (1/\log_e 2)^2 \,.\, c \,.\, n \,.\, \log_e (1/p_F)$$

<< ^ >>

❖ Exercise: CATC Signatures

Consider a signature-indexed table with the following properties

- $n = 3$, $r = 10000$, $c = 20$, $pF = 0.01$

Compute m (bits per signature), and m_i (bits per codeword) and k_i (bits set in codewords)

Show examples of codewords and signatures produced for this scheme.

Reminder:

$$m = (1/\log_e 2)^2 . n . \log_e (1/p_F)$$

<< ^ >>

❖ Exercise: Page-level CATC Signatures

Consider a signature-indexed table with the following properties

- $n = 3$, $r = 10000$, $c = 20$, $pF = 0.01$

Compute m (bits per signature), and m_i (bits per codeword) and k_i (bits set in codewords)

Show examples of codewords and signatures produced for this scheme.

Reminder:

$$m = (1/\log_e 2)^2 . c . n . \log_e (1/p_F)$$

<< ^ >>

❖ Exercise: SQL → RelAlg

Convert the following queries into (efficient?) RA expressions

```
select * from R where a > 5;

select * from R where id = 1234 and a > 5;

select R.a from R, S where R.i = S.j;

select R.a from R join S on R.i = S.j;

select * from R, S where R.i = S.j and R.a = 6

select R.a from R, S, T where R.i = S.j and S.k = T.y;
```

Assume **`R.id`** is a primary key and **`R`** is hashed on **`id`**

Assume that there is a B-tree index on **`R.a`**

COMP9315 21T1 ♢ Week 8 Exercises ♢ [5/22]

<< ^ >>

❖ Exercise: Alternative Join Plans

Consider the schema

```
Students(id,name,....)   Enrol(student,course,mark)
Staff(id,name,...)     Courses(id,code,term,lic,...)
```

the following query on this schema

```
select c.code, s.id, s.name
from   Students s, Enrol e, Courses c, Staff f
where  s.id=e.student and e.course=c.id
       and c.lic=f.id and c.term='19T2'
       and f.name='John Shepherd'
```

Show some possible evaluation orders for this query.

<< ^ >>

❖ Exercise: Selection Size Estimation

Assuming that

- all attributes have uniform distribution of data values
- attributes are independent of each other

Give formulae for the number of expected results for

1. `select * from R where not A=k`
2. `select * from R where A=k and B=j`
3. `select * from R where A in (k,l,m,n)`

where *j*, *k*, *l*, *m*, *n* are constants.

Assume: *V(A,R) = 10* and *V(B,R)=100* and *r=1000*

<< ^ >>

❖ Assignment 2 Q+A

Most common questions ...

- use of ADTs
- physical structure of bit-strings
- how many bits to set in codewords
- why doesn't x[123].c do something?
- is there a time-limit?
- where's the testing stuite?
- ...

<< ^ >>

❖ Exercise: Selection Size Estimation (ii)

Database stats for static table R(id,X,...)

- r = 5000, c = 50, tuples stored in X order, NULLs first
- V(X,R) = 5, **a,b,c,d,e,NULL** ~ 40:20:10:10:10:10

Estimate the number of result tuples and # pages read

1. `select * from R where X is not null`
2. `select * from R where X = 'a'`
3. `select * from R where X < 'a'`
4. `select * from R where X >= 'c'`
5. `select * from R where X between 'b' and 'd'`

Assume initial search is binary, if needed

<< ^ >>

❖ Exercise: Join Size Estimation

Assume **`S.id`** is a primary key, **`R.s`** is a FK referencing **`S.id`**

How many tuples are in the output from:

1. **`select * from R, S where R.s = S.id`**
2. **`select * from R, S where R.s <> S.id`**
3. **`select * from R, S where R.x = S.y`**

   where **`R.x`** and **`S.y`** have no connection except that *dom(R.x)=dom(S.y)*

Under what conditions will the first query have maximum/minimum size?

<< ^ >>

❖ Exercise: EXPLAIN examples

Consider this database ...

```
CourseEnrolments(student, course, mark, grade, ...)
Courses(id, subject, semester, homepage)
People(id, family, given, title, name, ..., birthday)
ProgramEnrolments(id, student, semester, program, wam, ...)
Students(id, stype)
Subjects(id, code, name, longname, uoc, offeredby, ...)
-- plus many other table
```

with this view

```
create view EnrolmentCounts as
 select s.code, c.semester, count(e.student) as nstudes
   from Courses c join Subjects s on c.subject=s.id
        join Course_enrolments e on e.course = c.id
  group by s.code, c.semester;
```

<< ^ >>

❖ Exercise: EXPLAIN examples (cont)

Some database statistics:

```
         tab_name          | n_records
---------------------------+-----------
 courseenrolments          |    503120
 courses                   |     71288
 people                    |     36497
 programenrolments         |    161110
 students                  |     31048
 subjects                  |     18799
```

<< ^ >>

❖ Exercise: EXPLAIN examples (cont)

Predict how each of the following queries will be executed ...

1. `select max(birthday) from People`
2. `select max(id) from People`
3. `select family from People order by family;`
4.
```
select distinct p.id, p.name
  from People p, CourseEnrolments e
  where p.id=e.student and e.grade='FL';
```
5.
```
select * from EnrolmentCounts where
  code='COMP9315';
```

Check your prediction using the **`EXPLAIN ANALYZE`** command.

Examine the effect of adding **`ORDER BY`** and **`DISTINCT`**.

Add indexes to improve the speed of slow queries.

COMP9315 21T1 ♢ Week 8 Exercises ♢ [13/22]

<< ^ >>

❖ Exercise: EXPLAIN examples (cont)

Example: Select on non-indexed attribute

```
uni=# explain
uni=# select * from Students where stype='local';
                        QUERY PLAN
-----------------------------------------------------
 Seq Scan on students
              (cost=0.00..562.01 rows=23544 width=9)
   Filter: ((stype)::text = 'local'::text)
```

<< ^ >>

❖ Exercise: EXPLAIN examples (cont)

Example: Select on non-indexed attribute with actual costs

```
uni=# explain analyze
uni=# select * from Students where stype='local';
                          QUERY PLAN
-----------------------------------------------------------
 Seq Scan on students
             (cost=0.00..562.01 rows=23544 width=9)
             (actual time=0.052..5.792 rows=23551 loops=1)
   Filter: ((stype)::text = 'local'::text)
   Rows Removed by Filter: 7810
 Planning time: 0.075 ms
 Execution time: 6.978 ms
```

<< ^ >>

❖ Exercise: EXPLAIN examples (cont)

Example: Select on indexed, unique attribute

```
uni=# explain analyze
uni-# select * from Students where id=100250;
                          QUERY PLAN
-----------------------------------------------------------
 Index Scan using student_pkey on student
             (cost=0.00..8.27 rows=1 width=9)
             (actual time=0.049..0.049 rows=0 loops=1)
   Index Cond: (id = 100250)
 Planning Time: 0.274 ms
 Execution Time: 0.109 ms
```

<< ^ >>

❖ Exercise: EXPLAIN examples (cont)

Example: Select on indexed, unique attribute

```
uni=# explain analyze
uni-# select * from Students where id=1216988;
                          QUERY PLAN
-----------------------------------------------------------
 Index Scan using students_pkey on students
             (cost=0.29..8.30 rows=1 width=9)
             (actual time=0.011..0.012 rows=1 loops=1)
   Index Cond: (id = 1216988)
 Planning time: 0.273 ms
 Execution time: 0.115 ms
```

❖ Exercise: EXPLAIN examples (cont)

Example: Join on a primary key (indexed) attribute (2016)

```
uni=# explain analyze
uni-# select s.id,p.name
uni-# from Students s, People p where s.id=p.id;
                        QUERY PLAN
-----------------------------------------------------------
Hash Join (cost=988.58..3112.76 rows=31048 width=19)
          (actual time=11.504..39.478 rows=31048 loops=1)
  Hash Cond: (p.id = s.id)
  -> Seq Scan on people p
        (cost=0.00..989.97 rows=36497 width=19)
        (actual time=0.016..8.312 rows=36497 loops=1)
  -> Hash (cost=478.48..478.48 rows=31048 width=4)
          (actual time=10.532..10.532 rows=31048 loops=1)
          Buckets: 4096  Batches: 2  Memory Usage: 548kB
       ->  Seq Scan on students s
              (cost=0.00..478.48 rows=31048 width=4)
              (actual time=0.005..4.630 rows=31048 loops=1)
 Planning Time: 0.691 ms
 Execution Time: 44.842 ms
```

COMP9315 21T1 ♢ Week 8 Exercises ♢ [18/22]

<< ^ >>

❖ Exercise: EXPLAIN examples (cont)

Example: Join on a primary key (indexed) attribute (2018)

```
uni=# explain analyze
uni-# select s.id,p.name
uni-# from Students s, People p where s.id=p.id;
                          QUERY PLAN
-----------------------------------------------------------
Merge Join  (cost=0.58..2829.25 rows=31361 width=18)
            (actual time=0.044..25.883 rows=31361 loops=1)
  Merge Cond: (s.id = p.id)
  ->  Index Only Scan using students_pkey on students s
            (cost=0.29..995.70 rows=31361 width=4)
            (actual time=0.033..6.195 rows=31361 loops=1)
        Heap Fetches: 31361
  ->  Index Scan using people_pkey on people p
            (cost=0.29..2434.49 rows=55767 width=18)
            (actual time=0.006..6.662 rows=31361 loops=1)
Planning time: 0.259 ms
Execution time: 27.327 ms
```

COMP9315 21T1 ♢ Week 8 Exercises ♢ [19/22]

❖ Exercise: EXPLAIN examples (cont)

Example: Join on a non-indexed attribute (2016)

```
uni=# explain analyze
uni=# select s1.code, s2.code
uni-# from Subjects s1, Subjects s2
uni=# where s1.offeredBy=s2.offeredBy;
                          QUERY PLAN
-------------------------------------------------------------
Merge Join (cost=4449.13..121322.06 rows=7785262 width=18)
           (actual time=29.787..2377.707 rows=8039979 loops=1)
 Merge Cond: (s1.offeredby = s2.offeredby)
 ->  Sort (cost=2224.57..2271.56 rows=18799 width=13)
          (actual time=14.251..18.703 rows=18570 loops=1)
     Sort Key: s1.offeredby
     Sort Method: external merge  Disk: 472kB
     ->  Seq Scan on subjects s1
               (cost=0.00..889.99 rows=18799 width=13)
               (actual time=0.005..4.542 rows=18799 loops=1)
 ->  Sort (cost=2224.57..2271.56 rows=18799 width=13)
          (actual time=15.532..1100.396 rows=8039980 loops=1)
     Sort Key: s2.offeredby
     Sort Method: external sort  Disk: 552kB
     ->  Seq Scan on subjects s2
```

```
                (cost=0.00..889.99 rows=18799 width=13)
                (actual time=0.002..3.579 rows=18799 loops=1)
Total runtime: 2767.1 ms
```

❖ Exercise: EXPLAIN examples (cont)

Example: Join on a non-indexed attribute (2018)

```
uni=# explain analyze
uni=# select s1.code, s2.code
uni-# from Subjects s1, Subjects s2
uni-# where s1.offeredBy = s2.offeredBy;
                          QUERY PLAN
-----------------------------------------------------------------
Hash Join  (cost=1286.03..108351.87 rows=7113299 width=18)
           (actual time=8.966..903.441 rows=7328594 loops=1)
  Hash Cond: (s1.offeredby = s2.offeredby)
  ->  Seq Scan on subjects s1
          (cost=0.00..1063.79 rows=17779 width=13)
          (actual time=0.013..2.861 rows=17779 loops=1)
  ->  Hash  (cost=1063.79..1063.79 rows=17779 width=13)
            (actual time=8.667..8.667 rows=17720 loops=1)
        Buckets: 32768  Batches: 1  Memory Usage: 1087kB
        ->  Seq Scan on subjects s2
                (cost=0.00..1063.79 rows=17779 width=13)
                (actual time=0.009..4.677 rows=17779 loops=1)
Planning time: 0.255 ms
Execution time: 1191.023 ms
```

COMP9315 21T1 ♢ Week 8 Exercises ♢ [21/22]

❖ Exercise: EXPLAIN examples (cont)

Example: Join on a non-indexed attribute (2018)

```
uni=# explain analyze
uni=# select s1.code, s2.code
uni-# from Subjects s1, Subjects s2
uni-# where s1.offeredBy = s2.offeredBy and s1.code < s2.code;
                          QUERY PLAN
-----------------------------------------------------------------
Hash Join  (cost=1286.03..126135.12 rows=2371100 width=18)
           (actual time=7.356..6806.042 rows=3655437 loops=1)
  Hash Cond: (s1.offeredby = s2.offeredby)
  Join Filter: (s1.code < s2.code)
  Rows Removed by Join Filter: 3673157
  ->  Seq Scan on subjects s1
          (cost=0.00..1063.79 rows=17779 width=13)
          (actual time=0.009..4.602 rows=17779 loops=1)
  ->  Hash  (cost=1063.79..1063.79 rows=17779 width=13)
            (actual time=7.301..7.301 rows=17720 loops=1)
        Buckets: 32768  Batches: 1  Memory Usage: 1087kB
        ->  Seq Scan on subjects s2
                (cost=0.00..1063.79 rows=17779 width=13)
                (actual time=0.005..4.452 rows=17779 loops=1)
```

```
Planning time: 0.159 ms
Execution time: 6949.167 ms
```

Produced: 8 Apr 2021