

Hw7

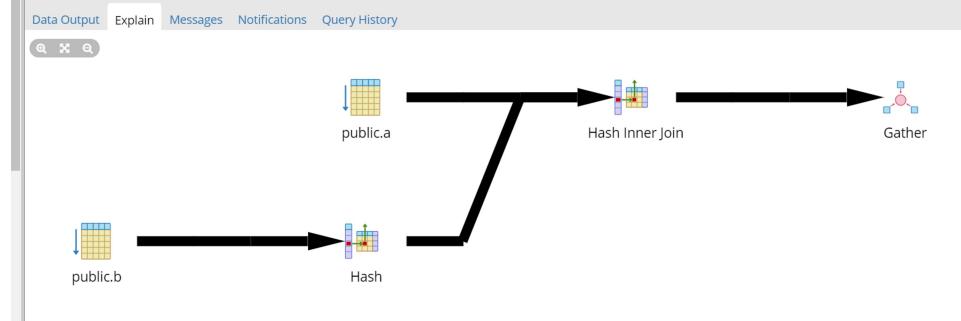
Sunday, November 4, 2018 16:19

I used Postgresql for this homework. I've included the jdbc code I used to generate the test data, and the SQL commands I used to generated the data, copy it to tables A,B,C,A',B', and C' and to create the secondary indexes.

Section 1: Explain Query Plans

1. A.pk = B.pk

```
22 CREATE INDEX c_ten_index ON t_prime(ten);  
23  
24 /* SECTION 1 */  
25 /* */  
26 SELECT * FROM A JOIN B ON A.pk = B.pk;  
27
```



System Chooses Hash Join. Hashes B and scans over A to do hash inner join

Expected Time:1338135.43

Expected Size:5000016 rows

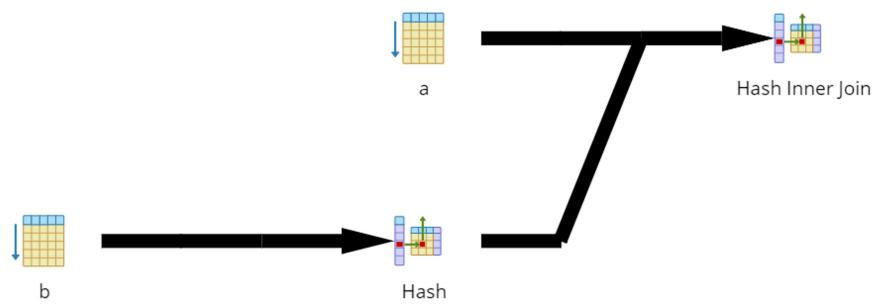
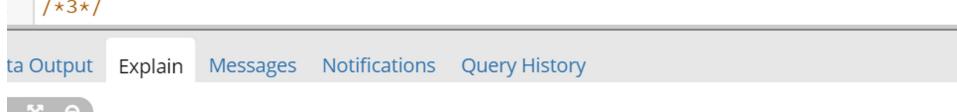
$T(A) = 5000000 V(A,pk) = 5000000$

$T(B) = 5000000 V(B,pk) = 5000000$

The selected plan is hash join.

2. A.ht = B.ht

```
/*2*/  
SELECT * FROM A JOIN B ON A.ht = B.ht;  
/*3*/
```



$$T(A) = 5000000 \quad V(A,ht) = 100000$$

$$T(B) = 5000000 \quad V(B,ht) = 100000$$

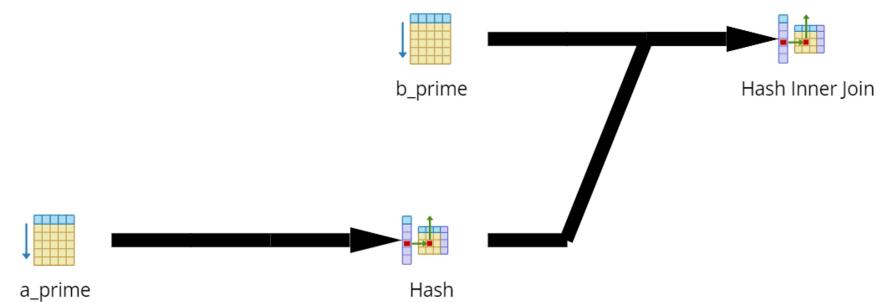
3. $A'.ht = B'.ht$

```

31 /*3*/
32 SELECT * FROM A_prime JOIN B_prime ON A_prime.ht =B_prime.ht;
33
34 /*4*/
35 SELECT * FROM A JOIN B ON A.ten =B.ten;

```

Data Output Explain Messages Notifications Query History



$$T(A') = 5000000 \quad V(A',ht) = 100000$$

$$T(B') = 5000000 \quad V(B',ht) = 100000$$

$V(R,attrib)$ did not change the query plan across 1,2,3. The secondary index has no impact on the selected query plan.

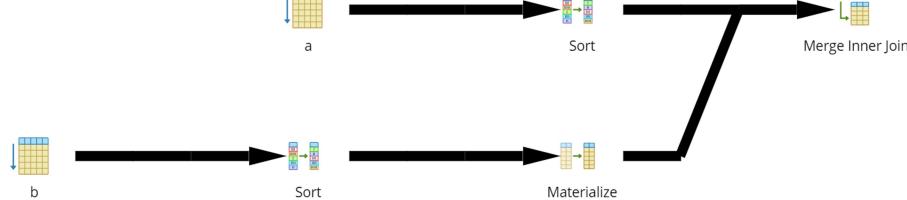
4. $A.ten = B.ten$

```

33 /*4*/
34 SELECT * FROM A JOIN B ON A.ten =B.ten;
35
36 /*5*/
37

```

Data Output Explain Messages Notifications Query History



$$T(A) = 5000000 \quad V(A,ten) = 10$$

$$T(B) = 5000000 \quad V(B,ten) = 10$$

Query plan selects Merge join

5. $A'.ten = B'.ten$

```

38 /*5*/
39 SELECT * FROM A_prime JOIN B_prime ON A_prime.ten =B_prime.ten;
40
41 /*6*/

```

Data Output Explain Messages Notifications Query History



```

36
37 /*5*/
38 SELECT * FROM A_prime JOIN B_prime ON A_prime.ten =B_prime.ten;
39
40 /*6*/

```

Data Output Explain Messages Notifications Query History

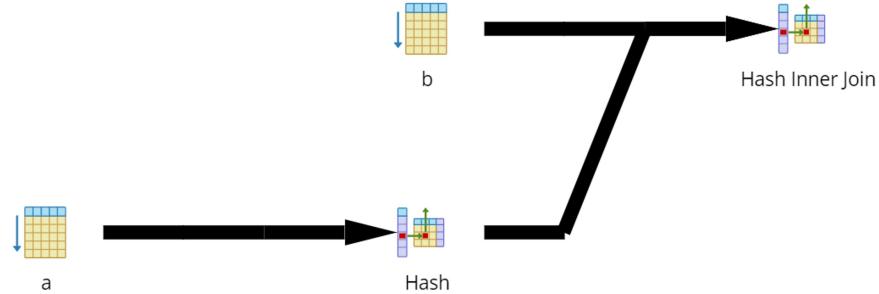


```

42
43 /*7*/
44 SELECT * FROM B JOIN A ON B.ten =A.ht;
45
46 /*8*/

```

Data Output Explain Messages Notifications Query History



$$T(A) = 5000000 \quad V(A, ht) = 100000$$

$$T(B) = 5000000 \quad V(B, ten) = 10$$

Selected plan is hash join again. The optimizer chose to hash a, then join b regardless of the order that a,b appear in the JOIN statement

8. A.ht = B'.ten

```

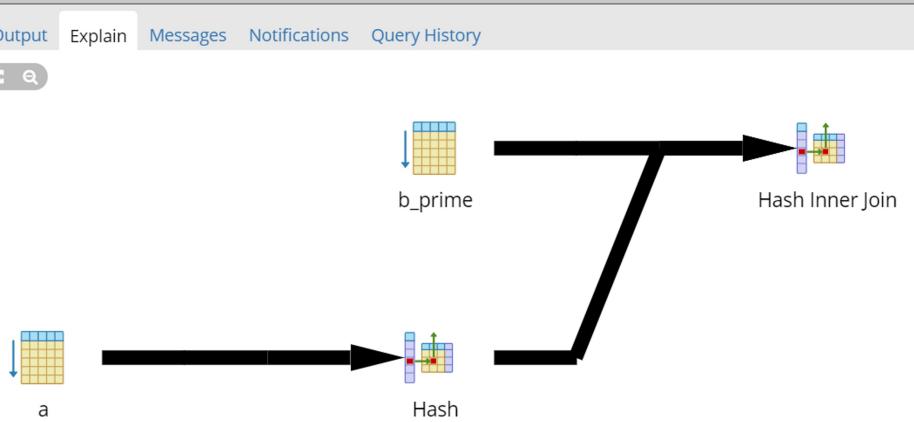
/*8*/
SELECT * FROM A JOIN B_prime ON A.ht =B_prime.ten;
/*9*/

```

Data Output Explain Messages Notifications Query History



1



$$T(A) = 5000000 \quad V(A, ht) = 100000$$

$$T(B') = 5000000 \quad V(B', ten) = 10$$

Selected plan is hash join

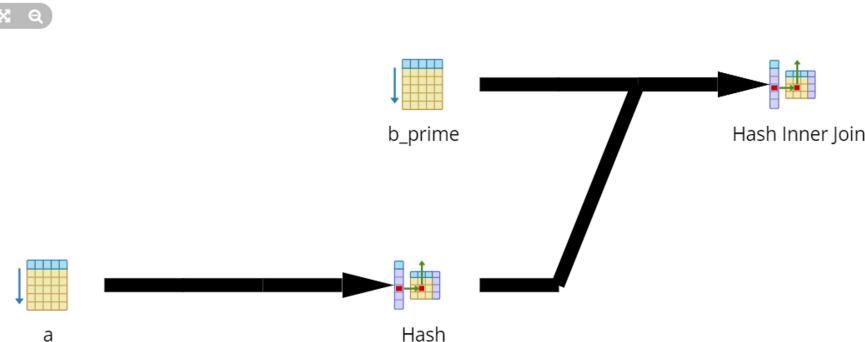
9. B'.ten = A.ht

```

47
48
49 /*9*/
50 SELECT * FROM B_prime JOIN A ON B_prime.ten = A.ht;
51
52 /* 3 way joins */
53 /*10*/
54 SELECT * FROM A JOIN B ON A.ht = B.ht JOIN C ON B.ht = C.ht;

```

Data Output Explain Messages Notifications Query History



$$T(A) = 5000000 \quad V(A,ht) = 100000$$

$$T(B') = 5000000 \quad V(B',ten) = 10$$

Selected plan is hash join. The optimizer again chose its own order regardless of how a,b appear in the JOIN statement

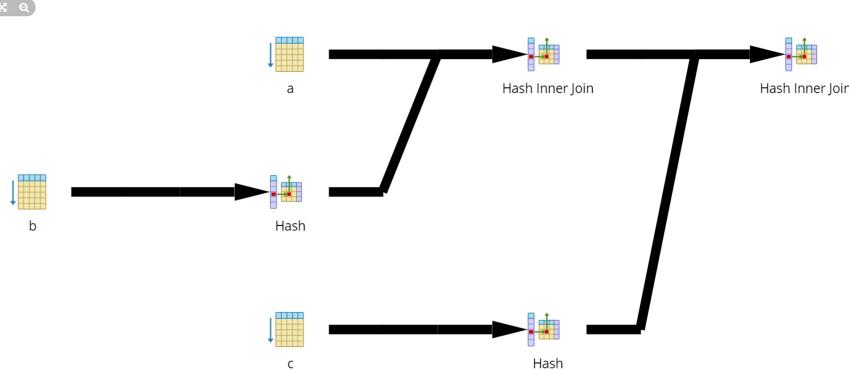
10. $A.ht = B.ht = C.ht$

```

51
52 /* 3 way joins */
53 /*10*/
54 SELECT * FROM A JOIN B ON A.ht = B.ht JOIN C ON B.ht = C.ht;
55
56 /*11*/
57 SELECT * FROM A JOIN B ON A.ten = B.ten JOIN C ON B.ten = C.ten;

```

Data Output Explain Messages Notifications Query History



$$T(A) = 5000000 \quad V(A,ht) = 100000$$

$$T(B) = 5000000 \quad V(B,ht) = 100000$$

$$T(c) = 5000000 \quad V(C,ht) = 100000$$

Selected plan: hash join A,b, hash join c

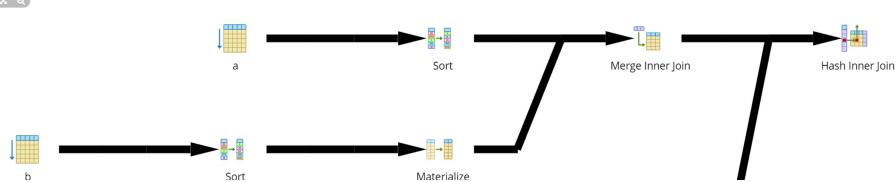
11. $A.ten = B.ten = C.ten$

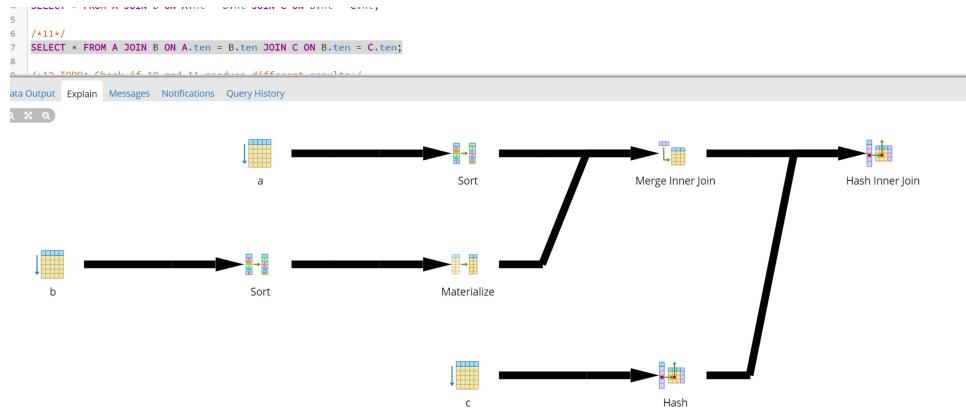
```

5
6 /*11*/
7 SELECT * FROM A JOIN B ON A.ten = B.ten JOIN C ON B.ten = C.ten;
8
9 /*12. YODA: Check if 24, 25 and 26 produce different answers*/

```

Data Output Explain Messages Notifications Query History





$$T(A) = 5000000 \quad V(A,ht) = 10$$

$$T(B) = 5000000 \quad V(B,ht) = 10$$

$$T(C) = 5000000 \quad V(C,ht) = 10$$

Selected plan: merge join A,B, hash join c

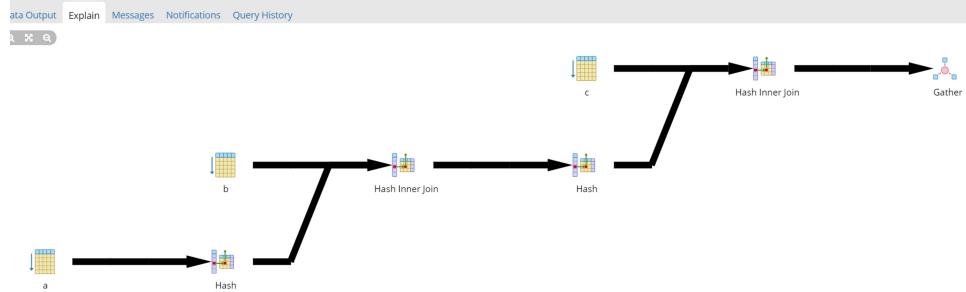
12. Do they produce different plans?

Yes, the plans are different. I found the plan changes on the ot attribute when $V(R,ot) = 1000$

$$A.ot = B.ot = C.ot$$

13. $A.pk = B.ht = C.ht$

```
/*12*/
SELECT * FROM A JOIN B ON A.ot = B.ot JOIN C ON B.ot = C.ot; */
/*13*/
SELECT * FROM A JOIN B ON A.pk = B.ht JOIN C ON B.ht = C.ht
/*14*/
SELECT * FROM A JOIN B ON A.pk = B.ht JOIN C ON B.ht = C.ht
```



Selected plan is to hash join a,b then hash join c

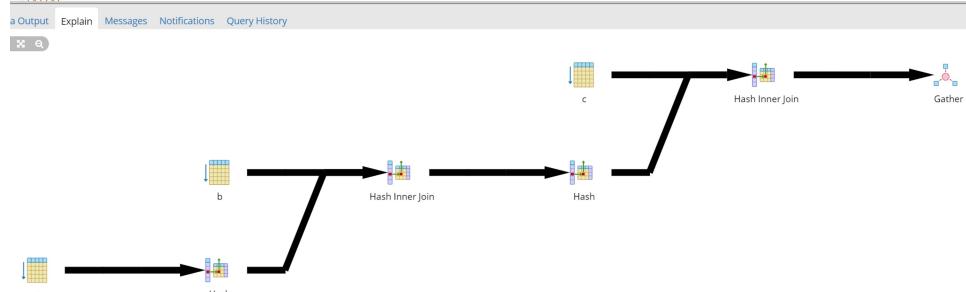
$$T(A) = 5000000 \quad V(A,pk) = 5000000$$

$$T(B) = 5000000 \quad V(B,ht) = 1000000$$

$$T(C) = 5000000 \quad V(C,ht) = 1000000$$

14. $A.pk = B.hund = C.hund$

```
/*14*/
SELECT * FROM A JOIN B ON A.pk = B.hund JOIN C ON B.hund = C.hund
/*15 TODO: Check if 13 and 14 produce different results */
/*16*/
SELECT * FROM A_prime JOIN B_prime ON A_prime.ht = B_prime.ht JOIN C_prime ON B_prime.ht = C_prime.ht
/*17*/
```



Selected plan is to hash join a,b then hash join c

$$T(A) = 5000000 \quad V(A,pk) = 5000000$$

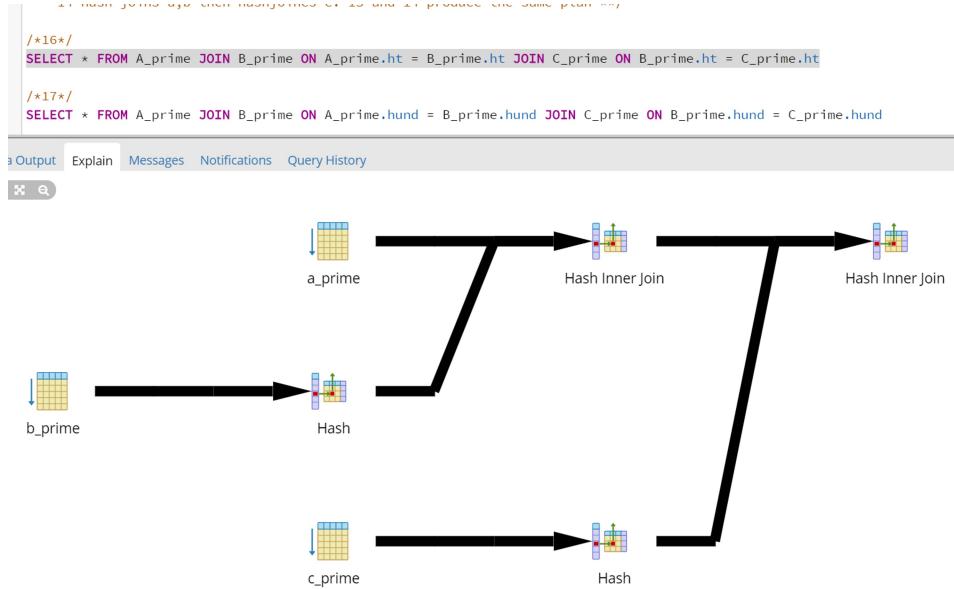
$$T(B) = 5000000 \quad V(B,hund) = 100$$

$$T(C) = 5000000 \quad V(C,hund) = 100$$

15. Do they produce different plans?

No, 13 and 14 produce the same plan.

16. $A'.ht = B'.ht = C'.ht$



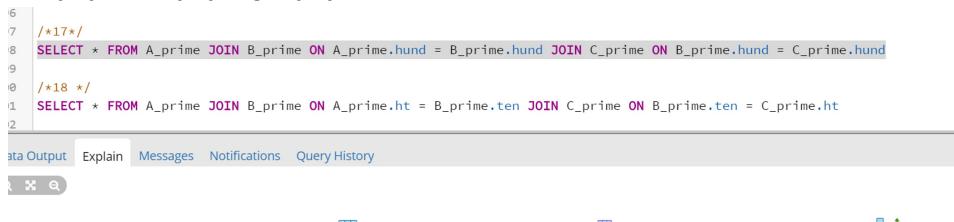
$$T(A) = 5000000 \quad V(A,ht) = 100000$$

$$T(B) = 5000000 \quad V(B,ht) = 100000$$

$$T(C) = 5000000 \quad V(C,ht) = 100000$$

Selected plan: hash join A,b, hash join c. The query plan does not change from 10, so the secondary index has no effect on the plan

17. $A'.hund = B'.hund = C'.hund$



$$T(A) = 5000000 \quad V(A,hund) = 100$$

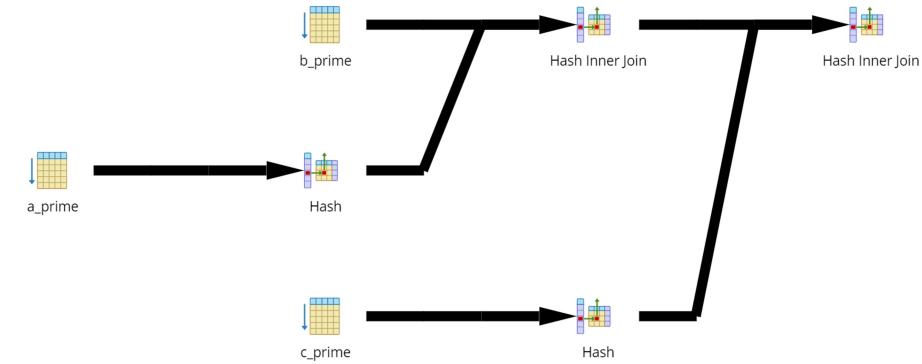
$$T(B) = 5000000 \quad V(B,hund) = 100$$

$$T(C) = 5000000 \quad V(C,hund) = 100$$

Selected plan: merge join A,b, hash join c. The query plan changes from 16.

18. a) $B'.ht = A'.ht = C'.ht$

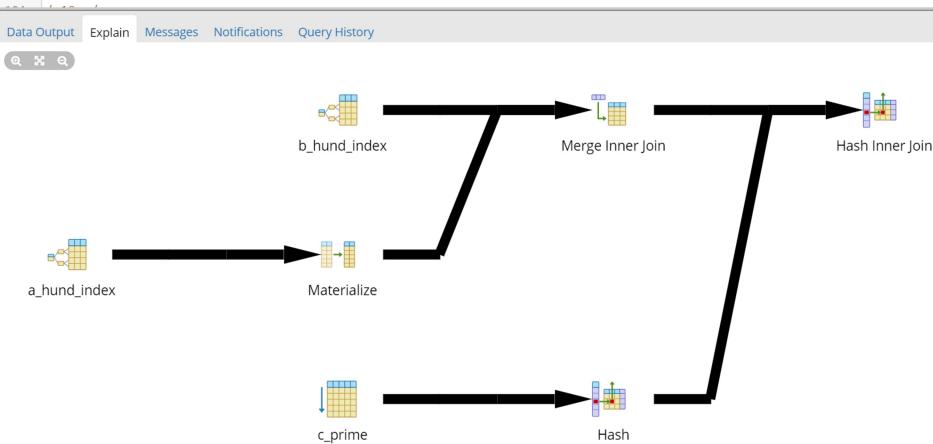
```
9  
0 /*18 */  
1 SELECT * FROM B_prime JOIN A_prime ON B_prime.ht = A_prime.ht JOIN C_prime ON C_prime.ht = B_prime.ht  
2 SELECT * FROM B_prime JOIN A_prime ON B_prime.hund = A_prime.hund JOIN C_prime ON C_prime.hund = B_prime.hund  
3
```



The query optimizer does not choose its own order. The query plan is dependent on the order in the statement

b) $B'.hund = A'.hund = C'.hund$

```
100 /*18 */  
101 SELECT * FROM B_prime JOIN A_prime ON B_prime.ht = A_prime.ht JOIN C_prime ON C_prime.ht = B_prime.ht  
102 SELECT * FROM B_prime JOIN A_prime ON B_prime.hund = A_prime.hund JOIN C_prime ON C_prime.hund = B_prime.hund  
103
```



This time the optimizer chooses its own order regardless of the order in the statement.

Section 2:

19. $A'.ht = C'.ot \text{ AND }$

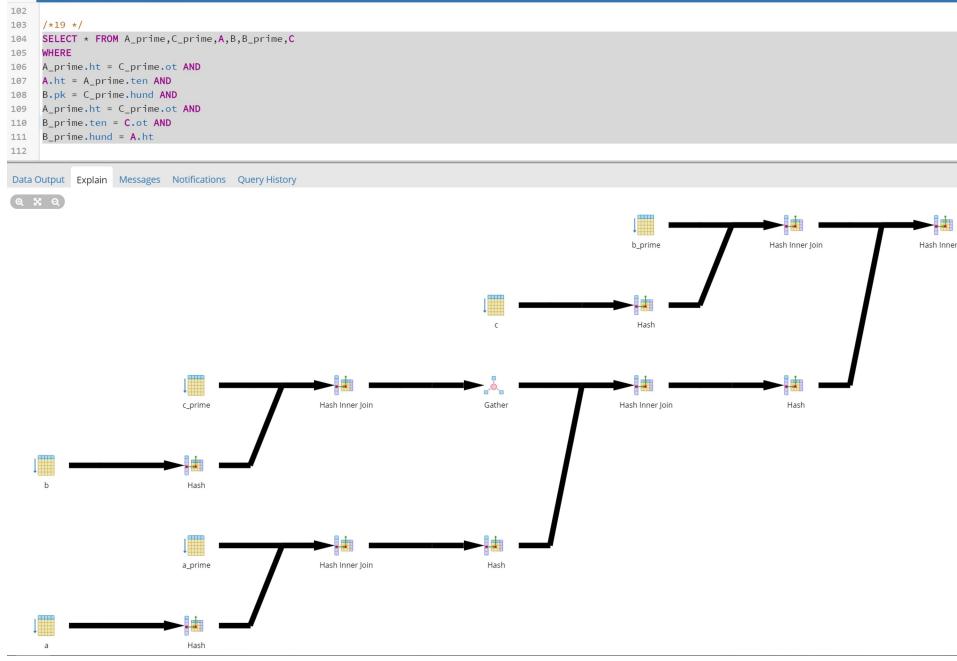
$A'.ht = A'.ten \text{ AND }$

$B'.pk = C'.hund \text{ AND }$

$A'.ht = C'.ot \text{ AND }$

$B'.ten = C'.ot \text{ AND }$

$B'.hund = A'.ht$



The estimated size in the last hash inner join is over 3 quadrillion rows (3,172,245,629,574,755)

20. $A'.ht = C'.ot \text{ AND}$

$A'.ten = 5 \text{ AND}$

$A.ht < A'.ten \text{ AND}$

$B.pk = C.hund \text{ AND}$

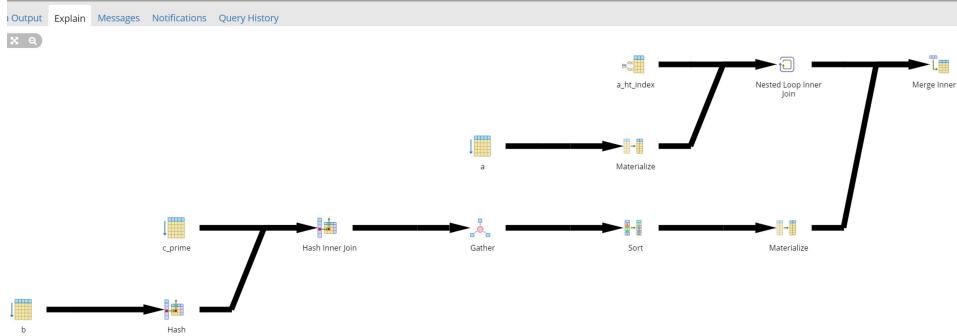
$B.ot < 500 \text{ AND}$

$A'.ht = C'.ot$

```

/*+28*/
SELECT * FROM A_prime,C_prime,A,B
WHERE
A_prime.ht = C_prime.ot AND
A_prime.ten = 5 AND
A.ht < A_prime.ten AND
B_pk = C_prime.hund AND
B.ot < 500 AND
A_prime.ht = C_prime.ot

```



The estimated size of the last merge inner join is: over 21 trillion rows(21,554,197,796,542)!

The compiler did not notice the equivalent predicate.

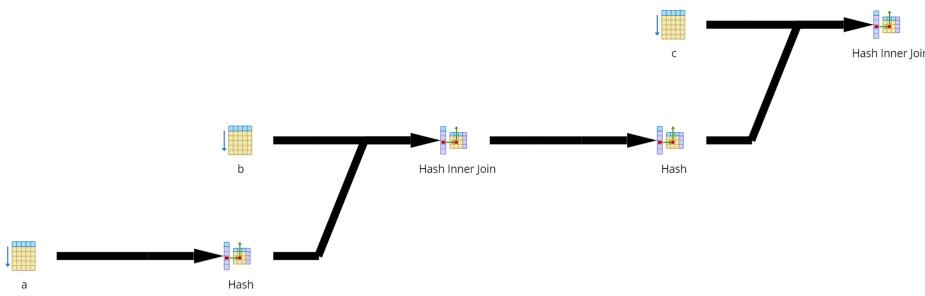
21. $A.ht = B.tt = C.ot$

```

16 /*+21*/
17
18 SELECT * FROM A
19 JOIN B ON A.ht = B.tt
20 JOIN C ON B.tt = C.ot
21
22 /*+22*/
23 SELECT * FROM C
24 JOIN B ON C.ot = B.tt
25 JOIN A ON B.tt = A.ht
26

```

Data Output Explain Messages Notifications Query History



$$T(A) = 5000000 \quad V(A,ht) = 100000$$

$$T(B) = 5000000 \quad V(B,tt) = 10000$$

$$T(C) = 5000000 \quad V(C,ot) = 1000$$

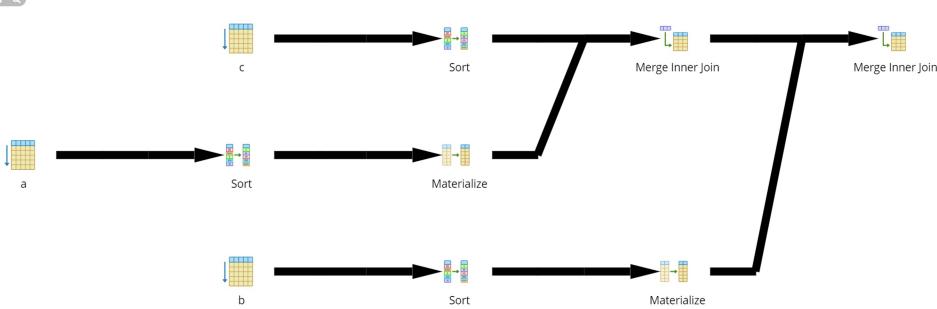
22. C.ot = B.tt = A.ht

```

27 /*+22*/
28
29 SELECT * FROM C
30 JOIN B ON C.ot = B.tt
31 JOIN A ON B.tt = A.ht
32

```

Data Output Explain Messages Notifications Query History



The query plan changes between 21 and 22 even though the only difference is the order of the tables in the query.

$$T(A) = 5000000 \quad V(A,ht) = 100000$$

$$T(B) = 5000000 \quad V(B,tt) = 10000$$

$$T(C) = 5000000 \quad V(C,ot) = 1000$$

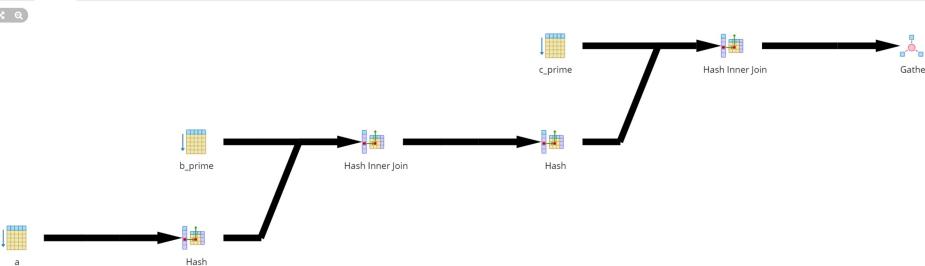
23. A.pk = B'.tt = C'.ot

```

32 /*+23*/
33
34 SELECT * FROM A
35 JOIN B_prime ON A.pk = B_prime.tt
36 JOIN C_prime ON B_prime.tt = C_prime.ot
37
38 /*+24*/
39 SELECT * FROM C_prime
40 JOIN B_prime ON C_prime.ot = B_prime.tt
41 JOIN A ON B_prime.tt = A.pk
42

```

Data Output Explain Messages Notifications Query History

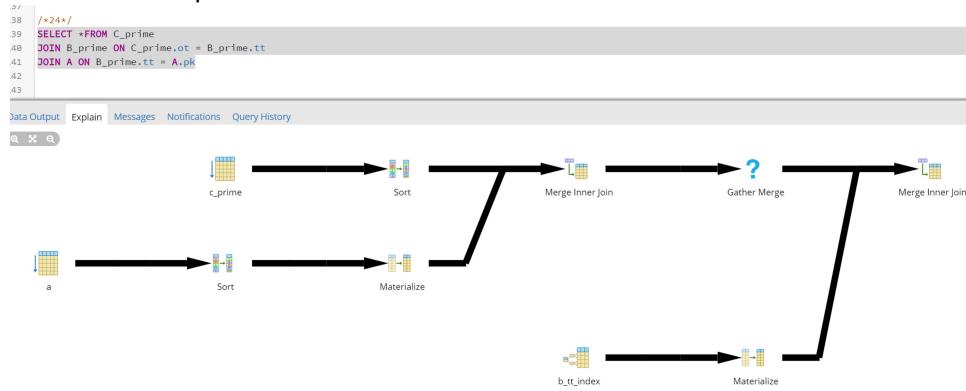


$$T(A) = 5000000 \quad T(A,pk) = 5000000$$

$$T(B) = 5000000 \quad T(B,tt) = 10000$$

$$T(C) = 5000000 \quad T(C,ot) = 1000$$

24. $C'.ot = B'.tt = A.pk$



$$T(A) = 5000000 \quad T(A,pk) = 5000000$$

$$T(B) = 5000000 \quad T(B,tt) = 10000$$

$$T(C) = 5000000 \quad T(C,ot) = 1000$$

Again, the query plan changes between 23 and 24 even though the only difference is the order of the tables in the query