Does the join order matter in SQL?

Asked 12 years, 9 months ago Modified 2 years ago Viewed 255k times

278

Disregarding performance, will I get the same result from query A and B below? How about C and D?

```
----- Scenario 1:
-- A (left join)
select *
from   a left join b
           on <blahblah>
       left join c
           on <blahblan>


-- B (left join)
select *
from   a left join c
           on <blahblah>
       left join b
           on <blahblan>

----- Scenario 2:
-- C (inner join)
select *
from   a join b
           on <blahblah>
       join c
           on <blahblan>


-- D (inner join)
select *
from   a join c
           on <blahblah>
       join b
           on <blahblan>
```

sql join relational-database

Share

Improve this question

Follow

edited Nov 26, 2022 at 8:29

hagrawal7777
14.6k ●5 ●45 ●78

asked Mar 8, 2012 at 8:48

Just a learner
28.4k ●53 ●163 ●246

12 What's `<blahblah>` ? are you joining A to B and A to C, or are you joining A to B and B to C? – beny23 Mar 8, 2012 at 8:57

8 Hi Beny, the code in my question is an abstraction. I'm not concerned on joining A to B or A to C, I just want to know will the syntax like that will provide identical results. – Just a learner Mar 8, 2012 at 9:22

4 Answers

Sorted by: Highest score (default)

324

For `INNER` joins, no, the order doesn't matter. The queries will return same results, as long as you change your selects from `SELECT *` to `SELECT a.*, b.*, c.*`.

For (`LEFT`, `RIGHT` or `FULL`) `OUTER` joins, yes, the order matters - and (***updated***) things are much more complicated.

First, outer joins are not commutative, so `a LEFT JOIN b` is not the same as `b LEFT JOIN a`

Outer joins are not associative either, so in your examples which involve both (commutativity and associativity) properties:

```
a LEFT JOIN b
    ON b.ab_id = a.ab_id
  LEFT JOIN c
    ON c.ac_id = a.ac_id
```

is equivalent to:

```
a LEFT JOIN c
    ON c.ac_id = a.ac_id
  LEFT JOIN b
    ON b.ab_id = a.ab_id
```

but:

```
a LEFT JOIN b
    ON  b.ab_id = a.ab_id
  LEFT JOIN c
    ON  c.ac_id = a.ac_id
    AND c.bc_id = b.bc_id
```

is not equivalent to:

```
a LEFT JOIN c
    ON  c.ac_id = a.ac_id
  LEFT JOIN b
    ON  b.ab_id = a.ab_id
    AND b.bc_id = c.bc_id
```

Another (hopefully simpler) associativity example. Think of this as `(a LEFT JOIN b) LEFT JOIN c`:

```
a LEFT JOIN b
    ON b.ab_id = a.ab_id          -- AB condition
 LEFT JOIN c
    ON c.bc_id = b.bc_id          -- BC condition
```

This **is equivalent** to `a LEFT JOIN (b LEFT JOIN c)`:

```
a LEFT JOIN
    b LEFT JOIN c
        ON c.bc_id = b.bc_id          -- BC condition
    ON b.ab_id = a.ab_id          -- AB condition
```

only because we have "nice" `ON` conditions. Both `ON b.ab_id = a.ab_id` and `c.bc_id = b.bc_id` are equality checks and do not involve `NULL` comparisons.

You can even have conditions with other operators or more complex ones like: `ON a.x <= b.x` or `ON a.x = 7` or `ON a.x LIKE b.x` or `ON (a.x, a.y) = (b.x, b.y)` and the two queries would still be equivalent.

If however, any of these involved `IS NULL` or a function that is related to nulls like `COALESCE()`, for example if the condition was `b.ab_id IS NULL`, then the two queries would not be equivalent.

Share
Improve this answer
Follow

edited Nov 16, 2013 at 21:36

answered Mar 8, 2012 at 9:24
ypercubeᵀᴹ
115k ● 19 ● 179 ● 245

4 It's more correct to say that the outer join is associative as long as neither predicate can be satisfied by a row in which all columns from one table are NULL, than to say that it's associative as long as the predicates don't involve IS NULL or 'a function that is related to nulls'. One can easily imagine a predicate that satisfies the former description but not the latter, like `a.somecol > 0 OR b.someothercol > 0`; associativity could fail for that condition. – Mark Amery Nov 17, 2013 at 11:03

1 But yeah, I think that it's technically true to say that OUTER JOIN is associative as long as the predicate doesn't satisfy either of the conditions I describe here: stackoverflow.com/questions/20022196/… (the first of which also breaks associativity for INNER JOINs, but is such a cheap and obvious approach to breaking it that perhaps it's not worth mentioning.) It's also worth pointing out that the most common kind of JOIN - JOINing on a foreign key - doesn't satisfy either of those conditions and thus is nice and associative. – Mark Amery Nov 17, 2013 at 11:06

2 @MarkAmery Thank you, I was having a hard time structuring my sentences on that point (and I have already upvoted that answer of yours ;) – ypercube™ Nov 17, 2013 at 11:10

ypercube i have a `INNER JOIN` and a following `LEFT JOIN`. Does it work like that first the query will `Filter` the records on the base of `INNER JOIN` and then will apply `LEFT JOIN` to the `Filtered` records? – Muhammad Babar Feb 12, 2015 at 13:09

1 In fact, all join types *are* associative, as specified by the SQL standard and according to mathematical definitions of associativity, but they don't *appear* associative because rearranging the parentheses requires moving the `ON` clause (i.e. the "join specification") to a new location. This is only syntax, though. If you use relational algebra notation (where the join specification is placed below the join operator), then associativity becomes more evident. Your argument only displays that outer joins are not *commutative*, which is correct – Lukas Eder May 9, 2015 at 15:51

14

If you try joining C on a field from B before joining B, i.e.:

```
SELECT A.x,
       A.y,
       A.z
FROM A
   INNER JOIN C
       on B.x = C.x
   INNER JOIN B
       on A.x = B.x
```

your query will fail, so in this case the order matters.

Share
Improve this answer
Follow

edited Dec 1, 2020 at 11:10

answered May 9, 2018 at 12:22
Teo J.
560 ● 7 ● 11

Yes this is right, the correct answer should be amended. – Nir Pengas Dec 12, 2019 at 18:12

8

for regular Joins, it doesn't. `TableA join TableB` will produce the same execution plan as `TableB join TableA` (so your C and D examples would be the same)

for left and right joins it does. `TableA left Join TableB` is different than `TableB left Join TableA`, BUT its the same than `TableB right Join TableA`

Share Improve this answer Follow

answered Mar 8, 2012 at 9:52
Diego
36.1k ● 21 ● 93 ● 138

6 This only addresses commutativity, but the examples in the question show that the asker is interested in associativity. ypercube's answer addresses both. – Mark Amery Dec 22, 2014 at

22:50

4

Oracle optimizer chooses join order of tables for inner join. Optimizer chooses the join order of tables only in simple FROM clauses . U can check the oracle documentation in their website. And for the left, right outer join the most voted answer is right. The optimizer chooses the optimal join order as well as the optimal index for each table. The join order can affect which index is the best choice. The optimizer can choose an index as the access path for a table if it is the inner table, but not if it is the outer table (and there are no further qualifications).

The optimizer chooses the join order of tables only in simple FROM clauses. Most joins using the JOIN keyword are flattened into simple joins, so the optimizer chooses their join order.

The optimizer does not choose the join order for outer joins; it uses the order specified in the statement.

When selecting a join order, the optimizer takes into account: The size of each table The indexes available on each table Whether an index on a table is useful in a particular join order The number of rows and pages to be scanned for each table in each join order

Share Improve this answer Follow

answered Apr 26, 2019 at 7:26

S Saumyojit Das
131 ● 1 ● 7

1 I believe this is a good answer. It could be reworded to explain that the Oracle optimizer does it because it is indeed an equivalence in certain cases. There is another person who wrote something similar: mastel.org/blog/when-join-order-matters which explains that ordering is taken as hints for the optimizer but the optimizer may also reorder joins to improve performance. – Pierre-Luc Bertrand Jun 16, 2021 at 2:20