Microsoft SQL Server performance- or in on clause [duplicate]

Asked 9 years, 1 month ago Modified 9 years, 1 month ago Viewed 135 times

0

This question already has answers here:

UNION ALL vs OR condition in sql server query (3 answers)

Closed 9 years ago.

I need to join 2 tables using a field 'cdi' from the 2nd table with cdi or cd_cliente from the 1st table. I mean that it might match the same field or cd_cliente from the 1st table.

My original query was

```
select
    a.cd_cliente, a.cdi as cdi_cli,b.*
from
    clientes a
left join
    rightTable b on a.cdi = b.cdi or a.cd_cliente = b.cdi
```

But since it took too much time, I changed it to:

```
Select a.cd_cliente, a.cdi, b.*
from clientes a
left join
    (select
         a.cd_cliente, a.cdi as cdi_cli, b.*
     from
         clientes a
     inner join
         rightTable  b on a.cdi = b.cdi
     union
     select
         a.cd_cliente, a.cdi as cdi_cli, b.*
     from
         clientes a
     inner join
         rightTable  b on a.cd_cliente = b.cdi) b
      on a.cd_cliente=b.cd_cliente
```

And it took less time. I'm not sure if the results would be the same. And if so, why the time taken by the 2nd query is considerably less?

sql sql-server performance join

Share
Improve this question
Follow

edited Nov 12, 2015 at 19:37

asked Nov 12, 2015 at 19:01
GabyLP
3,759 ● 8 ● 48 ● 69

You need to post the execution plans & index for us to help you. My guess is that the first one use a table scan and one of the 2 is a huge table so that takes a lot of time – Code Different Nov 12, 2015 at 19:11

2 Possible duplicate of UNION ALL vs OR condition in sql server query also stackoverflow.com/questions/15361972/… – hatchet - done with SOverflow Nov 12, 2015 at 19:18

@hatchet, not a duplicate. The linked question has two statements that mean the same thing. In this question, the two statements are not equivalent. – Shannon Severance Nov 12, 2015 at 19:21

First compare both result using `COUNT()` to see if return same number of rows. That way we can know for sure if return same result or not. Also please include your execution plan. stackoverflow.com/questions/7359702/…, – Juan Carlos Oropeza Nov 12, 2015 at 19:32

@JuanCarlosOropeza, I can't run the first query to count the number of rows. It takes more tan 30 minutes to run. – GabyLP Nov 12, 2015 at 19:46

3 Answers

Sorted by: Highest score (default)

1

I'm not sure if the results would be the same. Most likely not.

Consider a row in `clientes` that matched a row in `rightTable` on `cdi` but did not match any row on `cd_cliente`. The first query will return one row for the match. The second query will return two rows. Once for the match, and once for the not match, but with nulls filled in the `rightTable` columns because of the `left outer join`.

Also, if the first query returns any legitimate duplicates those will be removed by the `union` operator in the second query.

Share Improve this answer Follow

answered Nov 12, 2015 at 19:20
Shannon Severance
18.4k ● 4 ● 48 ● 67

Thanks, please see the edition. If I change it to inner joins inside and then I do a left join using any of the fields, do I get the same result? (Regarding duplicates, I shouldn't have them). – GabyLP Nov 12, 2015 at 19:44

Is `clientes.cd_cliente` the primary key or a unique key with a not null constraint?
– Shannon Severance Nov 12, 2015 at 20:17

1

SQL Server isn't good with `OR` and indexes. Not sure why. Your second query is getting around that by (most likely) seeking via indexes twice and then merging them somehow.

There are simpler queries you could try, such as this one:

```
SELECT
   a.cd_cliente,
   cdi_cli = a.cdi,
   b.*
FROM
   dbo.clientes a
   OUTER APPLY (
      SELECT *
      FROM dbo.rightTable b
      WHERE a.cdi = b.cdi
      UNION
      SELECT *
      FROM dbo.rightTable b
      WHERE a.cd_cliente = b.cdi
   ) b
;
```

And here's a weird one that could actually work, though I'm not sure:

```
SELECT
   a.cd_cliente,
   cdi_cli = a.cdi,
   b.*
FROM
   dbo.clientes a
   OUTER APPLY (
      SELECT *
      FROM dbo.rightTable b
      WHERE EXISTS (
         SELECT 1 WHERE a.cdi = c.cdi
         UNION
         SELECT 1 WHERE a.cd_cliente = b. cd_cliente
      )
   ) b
;
```

Told you it was weird! And here's an even weirder (and probably inadvisable) one.

```
SELECT
   a.cd_cliente,
   cdi_cli = a.cdi,
   BColumn1 = Max(BColumn1),
   BColumn2 = Max(BColumn2),
   BColumn3 = Max(BColumn3),
   BColumn4 = Max(BColumn4)
   -- all columns of B
FROM
   dbo.clientes a
   CROSS APPLY (VALUES
      (a.cdi),
      (a.cd_cliente)
   ) c (cdi)
   LEFT JOIN dbo.rightTable b
      ON c.cdi = b.cdi
GROUP BY
   a.cd_cliente,
   a.cdi,
   -- all columns of A
;
```

Given some time to play with your data and indexes and work with execution plans, I'm sure we could come up with something that would really sizzle.

Share Improve this answer Follow

answered Nov 12, 2015 at 22:50

ErikE
50.1k ● 23 ● 154 ● 200

0

This is your original query:

```
select a.cd_cliente, a.cdi as cdi_cli,b.*
from clientes a left join
     rightTable b
     on a.cdi = b.cdi or a.cd_cliente = b.cdi;
```

The performance problem is due to the `or` in the `on` condition. This generally interferes with using indexes.

If you only cared about one column from `b`, you could do:

```
select a.cd_cliente, a.cdi as cdi_cli, coalesce(b1.col, b2.col)
from clientes a left join
     rightTable b1
     on a.cdi = b1 left join
     rightTable b2
     on a.cd_cliente = b2.cdi;
```

These easily generalizes to a small handful of columns, but is cumbersome if `b` is wide.

Another way of writing the query would be much more cumbersome. It would start with the `b` table, double left join to `a` and then union in the remaining values from a:

```
select coalesce(a1.cd_cliente, a2.cd_cliente) as cd_cliente,
       coalesce(a1.cdi, a2.cd) as cdi_cli,
       b.*
from rightTable b left join
     clientes a1
     on a1.cdi = b.cdi left join
     clientes a2
     on a2.cd_cliente = b.cdi
where a1.cdi is not null or c2.cdi is not null
union all
select a.cd_cliente, a.cdi, b.*
from clientes a left join
     righttable b
     on 1 = 0
where not exists (select 1 from righttable b where a.cdi = b.cdi) and
      not exists (select 1 from righttable b where a.cd_cliente = b.cdi)
```

The first part of the query gets all the matching rows, to one or the other tables. The second adds in the unmatching rows. Note the strange use of `left join` with a condition that always evaluates to FALSE. That makes it easier to bring in the tables from `b`.

Although this looks complicated, the joins and `not exists` subqueries can all take advantage of appropriate indexes on the tables. That means that it should have more reasonable performance.

Share

Improve this answer

Follow

edited Nov 12, 2015 at 22:26

answered Nov 12, 2015 at 19:26

Gordon Linoff

1.3m ● 60 ● 686 ● 832