Database Tuning – Assignment 5

Join Tuning

A2
Baumgartner Dominik, 0920177
Dafir Thomas Samy, 1331483
Schörgnhofer Kevin, 1421082

December 1, 2016

1 setup

All queries were sent to the database-server (biber) using *psql* on the computers of the RÜR.

2 Join Strategies Proposed by System

Response times

Indexes	Join Strategy Q1	Join Strategy Q2
no index	Hash Join	Hash Join
unique non-clustering on `Publ.pubID`	Hash Join	Nested Loop Join
clustering on `Publ.pubID` and `Auth.pubID`	Merge Join	Nested Loop Join

Discussion Discuss here your observations. Is the choice of the strategy expected? How does the system come to this choice?
no index:
For this scenario the proposed join strategies are the expected ones. A hash join is in this case the best choice. For a merge join the tables should be sorted and the nested loop join should be used when there is a small table.

unique non-clustering on `Publ.pubID`:
For the first query a hash join is the best choice, because of the big, unsorted table of both. For this query the index is of no use.
For the second query a Nested Loop Join is used because first the name from the author must be searched which reduces the table size from $3 * 10^6$ to $\sim$200. Thereforce the nested loop join is the better choice.

clustering on `Publ.pubID` and `Auth.pubID`:
For the first query a Merge Join is proposed as expected. The reason therefore is that both tables are sorted to this attribute which is ideal for the Merge Join.
For the second query we expected a Merge Join, but the system used a Nested Loop Join. The Reason therefore is again the table size. First the name is searched in Auth

which reduces the size and also "destroys" the ordering. This means that the system would have to sort the values again for the Merge Join although the values are still sorted. And therefore a Nested Loop Join is the best choice.

3 Nested Loop Join

Response times

Indexes	Response time Q1 [ms]	Response time Q2 [ms]
index on `Publ.pubID`	98635	504
index on `Auth.pubID`	46672	193800
index on `Publ.pubID` and `Auth.pubID`	56395	511

Query plans

Index on `Publ.pubID` (Q1/Q2):

```
Q1:
 Nested Loop  (cost=0.43..1589599.47 rows=3095201 width=82) (actual time=0.088..97373.109 rows=30
   ->  Seq Scan on auth  (cost=0.00..57761.01 rows=3095201 width=38) (actual time=0.012..1694.669
   ->  Index Scan using pubidp on publ_i  (cost=0.43..0.48 rows=1 width=89) (actual time=0.028..0
         Index Cond: ((pubid)::text = (auth.pubid)::text)
 Planning time: 0.647 ms
 Execution time: 98634.857 ms

Q2:
 Nested Loop  (cost=0.43..65701.99 rows=24 width=67) (actual time=309.784..503.737 rows=183 loops
   ->  Seq Scan on auth  (cost=0.00..65499.01 rows=24 width=23) (actual time=309.720..498.274 row
         Filter: ((name)::text = 'Divesh Srivastava'::text)
         Rows Removed by Filter: 3095018
   ->  Index Scan using pubidp on publ_i  (cost=0.43..8.45 rows=1 width=89) (actual time=0.027..0
         Index Cond: ((pubid)::text = (auth.pubid)::text)
 Planning time: 0.137 ms
 Execution time: 503.837 ms
```

Index on `Auth.pubID` (Q1/Q2):

```
Q1:
 Nested Loop  (cost=0.43..849985.42 rows=3095201 width=82) (actual time=0.066..45483.489 rows=309
   ->  Seq Scan on publ  (cost=0.00..34694.14 rows=1233214 width=89) (actual time=0.009..703.605
   ->  Index Scan using pubida on auth_i  (cost=0.43..0.63 rows=3 width=38) (actual time=0.026..0
         Index Cond: ((pubid)::text = (publ.pubid)::text)
 Planning time: 0.432 ms
 Execution time: 46672.318 ms

Q2:
 Nested Loop  (cost=0.00..562648.46 rows=25 width=67) (actual time=15405.220..193800.011 rows=183
   Join Filter: ((auth_i.pubid)::text = (publ.pubid)::text)
   Rows Removed by Join Filter: 225677979
   ->  Seq Scan on publ  (cost=0.00..34694.14 rows=1233214 width=89) (actual time=0.012..689.233
   ->  Materialize  (cost=0.00..65499.14 rows=25 width=23) (actual time=0.000..0.071 rows=183 loo
         ->  Seq Scan on auth_i  (cost=0.00..65499.01 rows=25 width=23) (actual time=104.495..507
               Filter: ((name)::text = 'Divesh Srivastava'::text)
               Rows Removed by Filter: 3095018
 Planning time: 0.135 ms
 Execution time: 193800.146 ms
```

Index on `Auth.pubID` and `Auth.pubID` (Q1/Q2):

```
Q1:
 Nested Loop  (cost=0.43..850049.42 rows=3095201 width=82) (actual time=0.101..55181.800 rows=309
   ->  Seq Scan on publ_i  (cost=0.00..34758.14 rows=1233214 width=89) (actual time=0.037..721.90
   ->  Index Scan using pubida on auth_i  (cost=0.43..0.63 rows=3 width=38) (actual time=0.032..0
         Index Cond: ((pubid)::text = (publ_i.pubid)::text)
 Planning time: 0.148 ms
 Execution time: 56394.801 ms

Q2:
 Nested Loop  (cost=0.43..65710.45 rows=25 width=67) (actual time=103.244..511.228 rows=183 loops
   ->  Seq Scan on auth_i  (cost=0.00..65499.01 rows=25 width=23) (actual time=103.176..505.597 r
         Filter: ((name)::text = 'Divesh Srivastava'::text)
         Rows Removed by Filter: 3095018
   ->  Index Scan using pubidp on publ_i  (cost=0.43..8.45 rows=1 width=89) (actual time=0.028..0
         Index Cond: ((pubid)::text = (auth_i.pubid)::text)
 Planning time: 0.181 ms
 Execution time: 511.343 ms
```

Discussion Discuss here your observations. Are the response times expected? Why / why not?

4 Sort-Merge Join

Response times

Indexes	Response time Q1 [ms]	Response time Q2 [ms]
no index	too long	27886
two non-clustering indexes	37306	508
two clustering indexes	18891	515

Query plans

No index (Q1/Q2):

```
Q1:
Merge Join  (cost=846625.43..906956.29 rows=3095201 width=83)
   Merge Cond: ((publ.pubid)::text = (auth.pubid)::text)
   ->  Sort  (cost=285913.35..288996.38 rows=1233214 width=90)
         Sort Key: publ.pubid
         ->  Seq Scan on publ  (cost=0.00..34694.14 rows=1233214 width=90)
   ->  Materialize  (cost=560711.47..576187.47 rows=3095201 width=38)
         ->  Sort  (cost=560711.47..568449.47 rows=3095201 width=38)
               Sort Key: auth.pubid
               ->  Seq Scan on auth  (cost=0.00..57750.01 rows=3095201 width=38)

Q2:
Merge Join  (cost=351419.92..357590.96 rows=413 width=68)
    (actual time=24054.639..27857.060 rows=183 loops=1)
   Merge Cond: ((publ.pubid)::text = (auth.pubid)::text)
   ->  Sort  (cost=285913.35..288996.38 rows=1233214 width=90)
             (actual time=23183.951..26169.586 rows=1229958 loops=1)
         Sort Key: publ.pubid
         Sort Method: external merge  Disk: 121400kB
         ->  Seq Scan on publ  (cost=0.00..34694.14 rows=1233214 width=90)
```

```
                              (actual time=0.026..850.908 rows=1233214 loops=1)
   ->  Sort  (cost=65505.96..65506.99 rows=413 width=23)
     (actual time=519.464..519.550 rows=183 loops=1)
         Sort Key: auth.pubid
         Sort Method: quicksort  Memory: 39kB
         ->  Seq Scan on auth  (cost=0.00..65488.01 rows=413 width=23)
                               (actual time=8.420..518.464 rows=183 loops=1)
               Filter: ((name)::text = 'Divesh Srivastava'::text)
               Rows Removed by Filter: 3095018
 Planning time: 0.351 ms
 Execution time: 27886.590 ms
```

Two non-clustering indexes (Q1/Q2):

```
Q1:
Merge Join  (cost=0.86..263625.34 rows=3095201 width=83)
            (actual time=0.038..36048.698 rows=3095201 loops=1)
   Merge Cond: ((publ.pubid)::text = (auth.pubid)::text)
   ->  Index Scan using pubidpubl on publ  (cost=0.43..73055.43 rows=1233214 width=90)
   (actual time=0.006..6608.030 rows=1233208 loops=1)
   ->  Index Scan using pubidauth on auth  (cost=0.43..148974.61 rows=3095201 width=38)
          (actual time=0.005..13439.998 rows=3095201 loops=1)
 Planning time: 0.768 ms
 Execution time: 37306.638 ms

Q2:
Merge Join  (cost=65499.99..141460.64 rows=24 width=68)
    (actual time=508.011..508.011 rows=0 loops=1)
   Merge Cond: ((publ.pubid)::text = (auth.pubid)::text)
   ->  Index Scan using pubidpubl on publ  (cost=0.43..73055.43 rows=1233214 width=90)
   (actual time=0.006..0.006 rows=1 loops=1)
   ->  Sort  (cost=65499.56..65499.62 rows=24 width=23)
     (actual time=508.002..508.002 rows=0 loops=1)
         Sort Key: auth.pubid
         Sort Method: quicksort  Memory: 25kB
         ->  Seq Scan on auth  (cost=0.00..65499.01 rows=24 width=23)
       (actual time=507.995..507.995 rows=0 loops=1)
               Filter: ((name)::text = 'Divesh Srivastav'::text)
               Rows Removed by Filter: 3095201
 Planning time: 0.528 ms
 Execution time: 508.045 ms
```

Two clustering indexes (Q1/Q2):

```
Q1:
Merge Join  (cost=0.86..263629.21 rows=3095201 width=83)
    (actual time=0.022..17687.042 rows=3095201 loops=1)
   Merge Cond: ((publ.pubid)::text = (auth.pubid)::text)
   ->  Index Scan using pubidpubl on publ  (cost=0.43..73057.97 rows=1233214 width=90)
   (actual time=0.005..917.866 rows=1233208 loops=1)
   ->  Index Scan using pubidauth on auth  (cost=0.43..148975.94 rows=3095201 width=38)
   (actual time=0.004..2033.002 rows=3095201 loops=1)
 Planning time: 0.606 ms
 Execution time: 18891.404 ms

Q2:
Merge Join  (cost=65500.99..141464.18 rows=24 width=68)
    (actual time=515.935..515.935 rows=0 loops=1)
```

```
  Merge Cond: ((publ.pubid)::text = (auth.pubid)::text)
  ->  Index Scan using pubidpubl on publ  (cost=0.43..73057.97 rows=1233214 width=90)
              (actual time=0.006..0.006 rows=1 loops=1)
  ->  Sort  (cost=65500.56..65500.62 rows=24 width=23)
    (actual time=515.926..515.926 rows=0 loops=1)
        Sort Key: auth.pubid
        Sort Method: quicksort  Memory: 25kB
        ->  Seq Scan on auth  (cost=0.00..65500.01 rows=24 width=23)
                              (actual time=515.919..515.919 rows=0 loops=1)
              Filter: ((name)::text = 'Divesh Srivastav'::text)
              Rows Removed by Filter: 3095201
Planning time: 0.535 ms
Execution time: 515.970 ms
```

Discussion Discuss here your observations. Are the response times expected? Why / why not?

5 Hash Join

Response times

Indexes	Response time Q1 [ms]	Response time [ms] Q2
no index	9427	1668

Query plans

No Index (Q1/Q2):

```
Q1:
 Hash Join  (cost=68174.32..250399.34 rows=3095201 width=82) (actual time=1834.$
   Hash Cond: ((auth.pubid)::text = (publ.pubid)::text)
   ->  Seq Scan on auth  (cost=0.00..57761.01 rows=3095201 width=38) (actual ti$
   ->  Hash  (cost=34694.14..34694.14 rows=1233214 width=89) (actual time=1833.$
         Buckets: 4096  Batches: 64  Memory Usage: 2344kB
         ->  Seq Scan on publ  (cost=0.00..34694.14 rows=1233214 width=89) (act$
 Planning time: 0.445 ms
 Execution time: 9426.895 ms

Q2:
 Hash Join  (cost=65499.31..140273.15 rows=24 width=67) (actual time=607.564..1$
   Hash Cond: ((publ.pubid)::text = (auth.pubid)::text)
   ->  Seq Scan on publ  (cost=0.00..34694.14 rows=1233214 width=89) (actual ti$
   ->  Hash  (cost=65499.01..65499.01 rows=24 width=23) (actual time=516.867..5$
         Buckets: 1024  Batches: 1  Memory Usage: 11kB
         ->  Seq Scan on auth  (cost=0.00..65499.01 rows=24 width=23) (actual t$
               Filter: ((name)::text = 'Divesh Srivastava'::text)
               Rows Removed by Filter: 3095018
 Planning time: 0.111 ms
 Execution time: 1667.943 ms
```

Discussion What do you think about the response time of the hash index vs. the response times of sort-merge and index nested loop join for each of the queries? Explain.

Time in hours per person: **XXX**

Important: Reference your information sources!