

## Re: Select Statement: Join vs Inner Select

**Source:**

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2004-04/0164.html>

---

**From:** Anith Sen (*anith\_at\_bizdatasolutions.com*)

**Date:** 03/31/04

Date: Wed, 31 Mar 2004 10:16:09 -0600

Srinivas,

>> *The JOIN will definitely perform much better. In (2), the SELECT statement will be executed for each row, which is huge overhead.* <<

In general, there is no guarantee that the query with JOIN will perform better. Also, it is not correct that correlated subqueries are executed for each row. One major component of query optimization involves query analysis & if SQL Server cannot find an efficient plan in the first stage (trivial optimization), it may perform further simplifications like syntactical transformations, rearrangement of operations etc. This could result in better plans for subqueries over joins (and vice versa).

Here is a quick example where subquery beats a join :

```
--#1
SELECT a1.au_lname, a1.au_fname, SUM( a2.royaltyper)
  FROM Authors a1
 LEFT OUTER JOIN TitleAuthor a2
   ON a1.au_id = a2.au_id
 WHERE a1.State = 'CA'
 GROUP BY a1.au_lname, a1.au_fname ;
```

```
--#2
SELECT a1.au_lname, a1.au_fname,
  (SELECT SUM(a2.royaltyper)
   FROM TitleAuthor a2
   WHERE a1.au_id = a2.au_id )
  FROM Authors a1
 WHERE a1.State = 'CA';
```

One cannot assure that the execution plans will be simpler and optimal when using a join relative to a subquery. The query optimizer estimates a cost for each combination of join/subquery strategy, join order, and indexes and that is why in some cases, changing the order of table references in a join, especially outer joins, changes the plan & cost

Since efficiency depends on physical models, the only way Jason can find out if one construct performs better than the other is by testing both of them against his tables and comparing the results.

>> *Also, if you are executing this query from your VB app each time, SQL Server will have to prepare an execution plan for each call.* <<

While this may happen, in certain cases auto-parameterization can occur with ad hoc statements where the statements are parameterized and plan is cached. Turn on a Profiler trace with "SP:CacheHit" event, send an EXEC or an ad-hoc SQL statements from client and you may see the cache usage. In such cases there is no need for plan preparation & the efficiency may be comparable to a stored procedure. However in general, I do agree that stored procedures are definitely a better option than ad hoc queries.

--  
Anith