Index Hinting

-

Whitepaper

Mark Brummel
Apeldoorn, The Netherlands, September 2007
Version 1

Table of contents

Legal Notice

The information contained in this document represents the current view of Brummel Dynamics Services on the issues discussed as of the date of publication and is subject to change at any time without notice.

This document and its contents are provided AS IS without warranty of any kind, and should not be interpreted as an offer or commitment on the part of Brummel. Brummel cannot guarantee the accuracy of any information presented. BRUMMEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

The descriptions of other companies' products in this document, if any, are provided only as a convenience to you. Any such references should not be considered an endorsement or support by Brummel. Brummel cannot guarantee their accuracy, and the products may change over time. Also, the descriptions are intended as brief highlights to aid understanding, rather than as thorough coverage.

All trademarks are the property of their respective companies

Printed in the Netherlands

Introduction

Since update 6 for Dynamics NAV 4.0 SP3 Microsoft has chosen to perform index hinting by default.

This documents explains why this decision has been made and the disadvantages of this feature. It also gives a close insight into how and when to use it in specific scenario's.

What is an index hint

Index hinting means forcing SQL Server to use a particular index.

SQL Server has an advanced built in query optimizer that generates execution plans for queries. Those execution plans contain the index to be used to retrieve the data.

With index hinting you override the query optimizer.

Structure of the whitepaper

The whitepaper is divided into 3 parts. The first part gives a global overview of the problem definition and the available solutions. The second part takes a deep dive into the technical approach of the solution and the third part contains a personal note from the author and external information that is interesting to read.

Problem Definition

When using Microsoft Dynamics NAV 4.0 on a SQL Server 2005 platform sometimes queries take a long time to execute. These are often very simple queries and when executed directly on SQL Server using the Management Studio they are always fast.

The cause of this problem is the caching mechanism of the SQL Server query optimizer.

The query optimizer is a decision maker for SQL Server on how to read the data from the database. This is a highly intelligent mechanism that usually does a good job. The query optimizer decides which index is used to retrieve the data.

This is called an “execution plan” and gets compiled before it is used. This cost performance mainly on CPU.

For performance reasons SQL Server caches execution plans in memory so if the same query is executed again it can reuse it instead of compiling a new one.

Normally this is a good thing but in SQL Server 2005 Microsoft introduced a new feature called “parameter sniffing”. SQL 2000 also did this but in a less complex way and with less side effects.

This means more parameters are taken into consideration when compiling the execution plan.

Example 1 :

```
SELECT * FROM [G_L Entry] WHERE  [G_L Account No_] = '1000', [Posting Date] =
'01012006 00:00:00', and [Entry No_] = 10
```

The Entry No. 10 indicated that the entry is at the beginning of the table. SQL Server 2005 will recognise that and instead of using the index on G/L account No. and Posting Date it will use the clustered index and read the data. This is the fastest way. Good work SQL 2005!

This execution plan gets compiled into cache.

Example 2 :

Now this query will be run next

```
SELECT * FROM [G_L Entry] WHERE  [G_L Account No_] = '1000', [Posting Date] =
'01012006 00:00:00', and [Entry No_] = 1568753
```

But SQL 2005 will not compile a new execution plan. It will reuse the old plan and use the clustered index to read the data. Bad job SQL 2005.

Possible Solutions

The best solution would be to have the SQL team somehow find a fix for this.

The fact that it seems that with example 1 it can get the data faster is brilliant. Since the record is at the beginning of the table SQL reads the clustered index rather than the index which is faster. When you write new functionality it needs to be finished, not dropped half way. With the current solution the execution plan is reused for queries that read data that is in the middle of the clustered index where reading an index would be faster. The SQL team should be able to find a solution for that.

The other solution would be for the Dynamics NAV team to not use queries like this. Using filters like a '<=' parameter can be defined as "bad design" from a SQL Server perspective.

Why does Dynamics NAV use this anyway?

The biggest problem for the development team in Denmark is the backwards compatibility with the C/SIDE database and its behaviour. In Dynamics NAV basically all forms (UI) are flat views of tables that let users browse through them at any desired way. Ten or 15 years ago this was revolutionary. It gives the possibility to develop functionality very rapidly that always has a similar look and feel and is easy for new consultants and engineers to understand.

When you develop an application in Visual Basic for example you have to create a transaction to retrieve the data from and store the data back into the database. From a development standpoint that gives you much more control of what users do.

The C/SIDE Dynamics NAV database was designed to support the behaviour of the Forms in Dynamics NAV.

To support this behaviour in SQL Server the development team needs to be very creative when it comes to creating SQL statements. They have done a very good job at this and have figured out a way to do it that is robust and trustworthy.

One of the options users have in Dynamics NAV is to use List forms. This is a flat view of rows from a table with a certain filter. Users can browse through this set which can be multiple pages. When a user opens the G/L Entry form at the top of the table viewing only the first 10 records of the table the query shown in the example gets generated.

Redesigning SQL and Dynamics NAV is not a short term solution.

So are you stuck with this problem for customers who are already running Dynamics NAV on SQL 2005? No, there are workarounds for this problem. There are two supported ways to do this. One is hinting recompile, the other one is hinting a specific index.

Even though redesigning SQL is not a short term solution, the SQL Server Team should have started this months ago when the problem was first noticed.

A workaround can be to give users the opportunity to cancel long queries or to even abort them automatically with a readable error message explaining what has happened

Disadvantages of Index Hinting

One of the key features of the query optimizer is its ability to determine the best scenario of retrieving the data.

The C/SIDE database is loved by most developers and end users for its simplicity and predictability. It basically is a brilliant dumb thing that only does what it is told.

The advantage of SQL server experienced by most developers and end users is its ability to correct the mistakes they make.

In the C/SIDE client the performance of the entire system could be crashed by one single user forgetting to select the right index or a developer forgetting to put that into the C/AL code.

The query optimizer will correct this most of the time and use the correct index anyway.

On site experience learns that end users absolutely have no clue about indexes and databases. They just apply a filter and wait for the result. If that takes 10 seconds than it takes 10 seconds.

When applying index hinting to every query based on the order by Microsoft has sent us back to the old C/SIDE behavior of users having to be smart about filtering.

Unfortunately it just does not work in the real world.

In C/SIDE there were even add-on components which optimized indexes on forms based on the filters applied. That basically is writing a query optimizer for C/SIDE.

Index hinting can be a solution to the problem but it can also cause new issues. When the sorting and the filtering do not match it can result in large reads because the query optimizer will not correct this. Although this might not be happening that much in the standard NAV application where all C/AL statements are still optimised for C/SIDE there are a lot of customised databases and add-ons that do not have this. This might also happen when users want to filter their data differently from the sorting.

Another issue might be reporting where sorting might be needed on a non selective way. The query optimizer will correct this and read the data with a selective index and sort it before returning the set to the client.

Conclusion

Index hinting is not per se a bad thing but is only to be used as a last resort.

Hinting every index is just a huge overkill. It brings us back to C/SIDE behavior which most end users were so happy to get rid of.

The advise for when you apply hotfix 6 for Microsoft Dynamics NAV 4.0 SP3 is to disable index hinting. Only hint those indexes causing issues based on the information provided in this document.

Performance tuning is more than index hinting. So many variables are important for performance from hardware issues to application code to end user training.

If you have a large database and suffer from performance problems you are best off seeking expert assistance.

SQL Server Hinting

SQL Server uses the terminology hinting for giving the query optimizer directions on how to generate an execution plan. However with Index Hinting in specific, hinting is not the right term. It suggests that SQL can always ignore the hint and go its own way. Index Hinting means overriding the query optimizer and force it to use an index.

There are some options supported by Dynamics NAV to hint the query optimizer and some unsupported options. Recompile and Index Hinting are supported, plan guides and optimise for are supported by SQL Server but not documented for Dynamics NAV.

Recompile

From MSDN

> Instructs the SQL Server 2005 Database Engine to discard the plan generated for the query after it executes, forcing the query optimizer to recompile a query plan the next time the same query is executed. Without specifying RECOMPILE, the Database Engine caches query plans and reuses them. When compiling query plans, the RECOMPILE query hint uses the current values of any local variables in the query and, if the query is inside a stored procedure, the current values passed to any parameters.
>
> RECOMPILE is a useful alternative to creating a stored procedure that uses the WITH RECOMPILE clause when only a subset of queries inside the stored procedure, instead of the whole stored procedure, must be recompiled. For more information, see Recompiling Stored Procedures. RECOMPILE is also useful when you create plan guides. For more information, see Optimizing Queries in Deployed Applications by Using Plan Guides.

Using the recompile option you will generate more CPU overhead since it does not use the caching mechanism of the execution plans anymore.

Example

```
SELECT * FROM [G_L Entry] WHERE  [G_L Account No_] = '1000', [Posting Date] =
'01012006 00:00:00', and [Entry No_] = 10 WITH RECOMPILE
```

Index Hint

From MSDN

> Specifies the name or ID of the indexes to be used by the query optimizer when it processes the statement. Only one index hint per table can be specified.
> If a clustered index exists, INDEX(0) forces a clustered index scan and INDEX(1) forces a clustered index scan or seek. If no clustered index exists, INDEX(0) forces a table scan and INDEX(1) is interpreted as an error.
>
> **The alternative INDEX = syntax specifies a single index hint. This is supported only for backward compatibility.**
>
> If multiple indexes are used in the single hint list, the duplicates are ignored and the rest of the listed indexes are used to retrieve the rows of the table. The order of the indexes in the

index hint is significant. A multiple index hint also enforces index ANDing and the query optimizer applies as many conditions as possible on each index accessed. If the collection of hinted indexes is not covering, a fetch is performed after the SQL Server 2005 Database Engine retrieves all the indexed columns.

Example

```
SELECT * FROM [G_L Entry] WITH (INDEX([G_L Entry$0])) WHERE  [G_L Account
No_] = '1000', [Posting Date] = '01012006 00:00:00', and [Entry No_] = 10
```

The benefit of index hinting for the parameter sniffing issue is not to have the CPU overhead of the recompile option and to have predictability on the query.

Examples of tables where you can use index hinting depending on the table size:

- G/L Entries
- Customer Ledger Entries
- Warehouse Entries
- Some custom tables with the same structure

Plan Guides

With a plan guide it is possible to hint on a even more detailed level.

Example :

```
use [NavisionDB]
exec sp_create_plan_guide
@name = N'CustLedgerEntry_Guide1',
@stmt = N'SELECT * FROM "[NavisionDB]"."dbo"."[CompanyName]$Cust_ Ledger
Entry"
  WHERE (("Customer No_"=@P1)) AND "Customer No_"=@P2
     AND "Posting Date"=@P3 AND "Currency Code"=@P4
        AND "Entry No_">@P7 ORDER BY "Customer No_","Posting Date","Currency
Code","Entry No_"
           OPTION (FAST 10)',
@type = N'SQL',
@module_or_batch = NULL,
@params = N'@P1 varchar(20),@P2 varchar(20),@P3 datetime,@P4 varchar(10),@P5
varchar(20)',
@hints = N'OPTION (RECOMPILE)'
```

Optimise for

This is kind of a long shot and difficult to implement for Dynamics NAV but is should be mentioned.

Example :

```
SELECT * FROM sales WHERE country
OPTION (OPTIMIZE FOR (@country='USA')
```

When to use

How to find out if you are experiencing this issue in SQL 2005.

Usually this starts with users complaining about performance, but not all performance issues are related to this issue.

Generally performance issues start to come when hitting the 10GB database size. This is for NAV 4.0 only. There are no figures yet about 5.0.

Out of the box, Dynamics NAV 4.0 is not optimised for SQL. The indexes in the database are designed to perform on C/SIDE.

Microsoft has released this SQL Statement to find out if you have this issue in your database

```
SELECT TOP 30
st.text,
    SUBSTRING(st.text, (qs.statement_start_offset/2) + 1,
    ((CASE statement_end_offset
        WHEN -1 THEN DATALENGTH(st.text)
        ELSE qs.statement_end_offset END
            - qs.statement_start_offset)/2) + 1) as statement_text,
    execution_count,
         case
              when execution_count = 0 then null
              else total_logical_reads/execution_count
         end as avg_logical_reads,
      last_logical_reads,
      min_logical_reads,
      max_logical_reads,
      max_elapsed_time,
         case
              when min_logical_reads = 0 then null
              else max_logical_reads / min_logical_reads
         end as diff_quota
FROM sys.dm_exec_query_stats as qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) as st
--WHERE max_logical_reads > 50000 and max_logical_reads > 5*
min_logical_reads
ORDER BY max_logical_reads DESC
```

The result of the query shows one line for each query plan that currently exists in the cache. Additionally, the result shows which query will trigger the query plan. You can compare the value in the **min_logical_reads** column with the value in the **max_logical_reads** column.

If a query plan generates a few reads sometimes, and then it generates many reads other times, you might be suffering the issue described.

How to use

You can activate both Recompile and Index Hint using an internal Dynamics NAV configuration table.

This table has to be created on SQL Server in your Dynamics NAV Database

```
CREATE TABLE [$ndo$dbconfig] (config VARCHAR(512) NOT NULL)

GRANT SELECT ON [$ndo$dbconfig] TO public
```

To activate an index hint for a paricular index use

```
INSERT INTO [$ndo$dbconfig]

VALUES ('IndexHint=Yes;Company="DEFAULT COMPANY";Table="Vendor";Key="Search
Name";Search Method="-+$";Index=1')
```

To activate a recompile for a particular statement use

```
INSERT INTO [$ndo$dbconfig]

VALUES ('UseRecompileForTable="G/L Entry";Company="CRONUS International
Ltd.";RecompileMode=1;')
```

NOTE : to use recompile you need to install SQL Server SP 2 and cumulative hotfix 933097 (build 3152)

When you use update 6 for Dynamics NAV 4.0 SP3 Index hinting is activated by default for all queries.

You can turn this off by creating this entry in the configuration table.

```
INSERT INTO [$ndo$dbconfig]

VALUES ('IndexHint=No')
```

More information about the configuration table on

http://www.waldo.be

https://mbs.microsoft.com/knowledgebase/KBDisplay.aspx?scid=kb;en-us;935395

http://blogs.msdn.com/microsoft_dynamics_nav_sustained_engineering/archive/2007/09/19/index-hinting-in-platform-update-for-microsoft-dynamics-nav-4-0-sp3-kb940718.aspx

Parameter Sniffing

On the internet there are some interesting posts about avoiding the parameter sniffing issue.

A good explanation is using the example from

> http://blogs.msdn.com/khen1234/archive/2005/06/02/424228.aspx

In his example he is using an unbalanced clustered index in SQL 2000 but the example is good in explaining the issue.

The idea is to give the query a default value which will cause the compiler to always use the same plan and then change the value for the actual value.

This will of course be difficult for Dynamics NAV since every customer as another value for using a correct execution plan.

Example:

```
CREATE PROC prSalesByCountry

@country char(3),

@template_country char(3)="USA"

AS

SET @template_country=@country

SELECT * FROM sales WHERE country=@template_country

GO
```

However the web contains tons of information about parameter sniffing and very detailed whitepapers like http://www.microsoft.com/technet/prodtechnol/sql/2005/recomp.mspx

This is another blog about the same issue but using SQL 2005

http://glennberrysqlperformance.spaces.live.com/Blog/cns!45041418ECCAA960!541.entry

Some expert recommendations for Index hints

www.sql-server-performance.com

"Index hints should only be used when the Query Optimizer, for whatever reason, does not select the available index, and if using the index provides a performance enhancement over not using an index."

Link to the document

Microsoft MSDN

"Caution

Because the SQL Server 2005 query optimizer typically selects the best execution plan for a query, we recommend that <join_hint>, <query_hint>, and <table_hint> be used **only as a last resort** by experienced developers and database administrators."

"INDEX **(** *index_val* [**,**... *n*] **)**

Specifies the name or ID of the indexes to be used by the query optimizer when it processes the statement. Only one index hint per table can be specified.

If a clustered index exists, INDEX(0) forces a clustered index scan and INDEX(1) forces a clustered index scan or seek. If no clustered index exists, INDEX(0) forces a table scan and INDEX(1) is interpreted as an error.

The alternative INDEX = syntax specifies a single index hint. This is supported only for backward compatibility.

If multiple indexes are used in the single hint list, the duplicates are ignored and the rest of the listed indexes are used to retrieve the rows of the table. The order of the indexes in the index hint is significant. A multiple index hint also enforces index ANDing and the query optimizer applies as many conditions as possible on each index accessed. If the collection of hinted indexes is not covering, a fetch is performed after the SQL Server 2005 Database Engine retrieves all the indexed columns. "

Link to the document

SqlSkills (Kimberly Tripp)

"Q: How well are Microsoft's standard products that use SQL optimized? Are there published studies on optimizing Microsoft CRM, Sharepoint Services, Project Server, and the various Microsoft Business Products?

This one I kind of answered during the webcast…First, I will start by mentioning that I'm NOT an expert on any of these products specifically. However, you are correct in that they are based on SQL Server and use SQL Server as their data store. What that means is that they too must be optimized and maintained; however, it is impossible for anyone to predict perfectly the exact usage of each of these products and as a result they might have a good selection of indexes but they are unlikely to have maintenance operations defined. However – again, I

cannot say that I am an expert and I could be wrong! What I will say however, is that regardless of what is defined and what is maintained – you can create additional indexes, possibly drop indexes (more on this in a moment) and maintain indexes with NO application changes. The reason why I say "possibly" drop indexes is because queries/procedures, etc. will FAIL if index hints are used and the index does not exist at runtime. **And while I'm fairly certain that NO Microsoft product hardcodes index hints**, I have seen other third party products which do and unfortunately they become VERY hard to optimize and manage. In general, it is harder to drop indexes than add indexes. Nevertheless, it is always possible to defrag/rebuild indexes – regardless of the application whether Microsoft of not. So, while I can't speak to each of these applications directly (although my partner works on Microsoft Project and he's pretty sharp! ☺), you can always improve upon an application with better/more useful indexes (meaning more specific indexes for YOUR usage patterns and requirements) and proper/consistent/automated index maintenance."

Link to the document

Word of the author

It is hard to argue with a decision that can do the miracles which index hinting can do. On the other hand there is a fantastic query optimizer which can do even bigger miracles correcting users and developers mistakes whether on purpose or not.

Fact is that the problem is in SQL Server, not in Dynamics NAV. Try searching the web for “Parameter Sniffing” and you’ll find that we are not alone in this subject. So bottom line is that the SQL Server team should take care of it and fix this undocumented feature.

There is nothing wrong with the behavior of Dynamics NAV out of the box.

Hynek Muhlbacher and SQL Perform have done a very good job in the past improving the base product and it is a shame that this road to improvement is not being used anymore.

Please let me say that I do not see myself as the master on SQL. I just do my work in by region and help partners and customers solve issues. I have only seen a couple of examples where this issue happens. Microsoft must have seen hundreds and therefore I am very curious how many of these sites were tested with this update and how they measured the increase of performance.

I would like Microsoft to release their test documentation where they tell us how large the sites were, how many users there were on that system, what the problems were, etc. I also would like to hear how the SQL Server development team participated in this.

Reviewers

I would like to thank the reviewers of this document because without their input it was not possible to generate is.

Hynek Muhlbacher – SQL Perform

Daniel Rimmelzwaan – Ris +

David Studebaker – Liberty Grove Software USA
Author of Programming Microsoft® Dynamics™ NAV

Eric Wauters – iFacto

Peter Wijntjes – LEAD

Sources of information

SQL Perform - Hynek Muhlbacher
http://www.sqlperform.com
iFacto - Eric Wauters
http://www.waldo.be
Microsoft Dynamics NAV Sustained Engineering Team Weblog
http://blogs.msdn.com/microsoft_dynamics_nav_sustained_engineering/archive/2007/09/20/is-sql2000-more-clever-than-sql2005.aspx
http://blogs.msdn.com/microsoft_dynamics_nav_sustained_engineering/archive/2007/09/19/index-hinting-in-platform-update-for-microsoft-dynamics-nav-4-0-sp3-kb940718.aspx
Microsoft MSDN
http://msdn2.microsoft.com/en-us/library/ms187373.aspx
http://blogs.msdn.com/khen1234/archive/2005/06/02/424228.aspx
SQL Skills – Kimberly Tripp
http://www.sqlskills.com/blogs/kimberly/2004/07/24/MSDNWebcastQAIndexDefragBestPracticesGeneralQuestions.aspx
SQL Server Performance .com
http://www.sql-server-performance.com/tips/hints_table_p1.aspx
Microsoft Dynamics Partner Source
https://mbs.microsoft.com/knowledgebase/KBDisplay.aspx?scid=kb;en-us;935395
Other useful websites
www.mibuso.com
www.dynamicsusers.com
www.brummelds.com