STUDIA INFORMATICA 2007
Volume 28 Number 2 (71)

Marcin WOCH
Politechnika Śląska, Instytut Informatyki

MICROSOFT SQL SERVER OPTIMIZATION FOR MICROSOFT DYNAMICS NAV

Summary. This article presents some methods of SQL Server 2000 or SQL Server 2005 for Microsoft Dynamics NAV optimization. It focuses on database design, programming standards and a hardware configuration.

Keywords: SQL Server 2000, SQL Server 2005, Microsoft Dynamics NAV, Navision Attain, locking, optimization, deadlock

OPTYMALIZACJA MICROSOFT SQL SERVER PRZY WSPÓŁPRACY Z MICROSOFT DYNAMICS NAV

Streszczenie. Artykuł prezentuje wybrane metody optymalizacji pracy serwera baz danych SQL 2000 oraz 2005 przy współpracy z systemem klasy ERP Microsoft Dynamics NAV. Zwraca on szczególną uwagę na odpowiednie zaprojektowanie bazy danych, poprawny kod źródłowy aplikacji, a także podaje zalecaną konfigurację sprzętową.

Słowa kluczowe: SQL Server 2000, SQL Server 2005, Microsoft Dynamics NAV, Navision Attain, blokowanie, optymalizacja, deadlock

1. Introduction

Microsoft Dynamics NAV – called Navision, formerly Navision Attain and Navision Financials is an ERP (Enterprise Resource Planning) system which covers all areas of company's activities: accountancy, sales, receivables, purchase, payables, CRM, manufacturing, inventory, etc.

Navision system can run on two different servers: Microsoft Navision Database Server and Microsoft SQL Server. To a user these two servers work and look exactly the same. Some of the differences between these servers are:

- scalability
- SQL Server has no database size limit
- each database can have multiple filegroups on SQL Server
- large RAM support on SQL Server
- SQL Server is cluster aware
- performance monitoring
- SQL Server supports multi-processor, both 32 and 64 bit
- the way that SIFT works
- others.

In this article SQL Server option for Navision will be described.

2. Architecture Overview

Microsoft Dynamics NAV can run on Microsoft SQL Server, which is integrated with C/SIDE (Client/Server Integrated Development Engine). New versions of Navision support SQL 2000 and SQL 2005 server. However, much better performances are achieved on 2005 version.

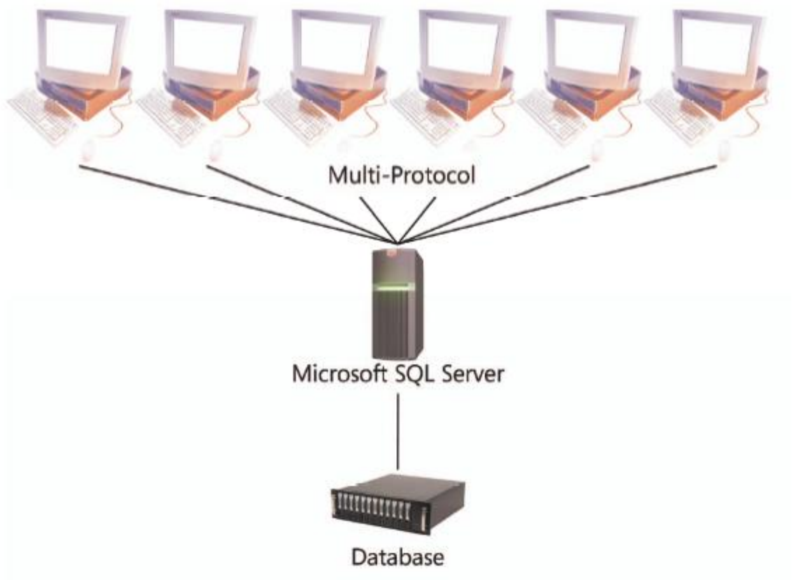


Fig. 1. MS SQL Server Option for MS Dynamics NAV architecture
Rys. 1. Architektura opcji MS SQL Server dla MS Dynamics NAV

3. Hardware recommendations

Minimum hardware recommendations depend on operational system and SQL Server version used. For example SQL Server 2005 requires minimum 512 MB RAM [8], but the better configuration the better results.

Hardware recommendations presented in that article are much higher then minimum one. Real company's expectations may be lowered due to available funds.

Dual Core processor provides equivalent or better computing power then two separate processors [2].

Table 1

MS Dynamics NAV option for MS SQL Server 2000 [2]

Windows Edition	SQL Server Edition	Concurrent No. of Users	SQL Db. Size [GB]	Proc.	RAM [GB]	Net. Card
MS Win 2003 Server St.	MS SQL Server 2000 St.	≤ 50	≤ 40	2 × 3 GHz 2 MB L3 Cache	3	1 GB Eth/Fibre
MS Win 2003 Server Ent.	MS SQL Server 2000 Ent.	51÷100	> 40	2 × 3 GHz 2 MB L3 Cache	8	2 GB Eth/Fibre

Both MS Windows 2003 Standard and Enterprise editions require SQL Server 2000 Service Pack 3 or later [2].

Table 2

MS Dynamics NAV option for MS SQL Server 2005 [2]

Windows Edition	SQL Server Edition	Concurrent No. of Users	SQL Db. Size [GB]	Proc.	RAM [GB]	Net. Card
MS Win SBS 2003 R2 St.	MS SQL Server 2005 Workgroup	≤ 5	≤ 20	1 × 3 GHz 2 MB Cache	2	1 GB Eth.
MS Win 2003 Server St.	MS SQL Server 2000 St.	6÷25	21÷40	2 × 3 GHz 2 MB Cache	3	1 GB Eth.
MS Win 2003 Server St.	MS SQL Server 2000 St.	26÷50	21÷40	2 × 3 GHz 2 MB L3 Cache	3	1 GB Eth/Fibre
MS Win 2003 Server St.	MS SQL Server 2000 St.	51÷100	41÷80	4 × 3 GHz 2 MB L3 Cache	4	1 GB Eth/Fibre
MS Win 2003 Server Ent.	MS SQL Server 2000 Ent.	101÷150	41÷80	4 × 3 GHz 2 MB L3 Cache	6	2 GB Eth/Fibre
MS Win 2003 Server Ent.	MS SQL Server 2000 Ent.	151÷200	> 80	6 × 3 GHz 2 MB L3 Cache	8	2 GB Eth/Fibre
MS Win 2003 Server Ent.	MS SQL Server 2000 Ent.	201 – 250	> 80	8 × 3 GHz 2 MB L3 Cache	8	2 GB Eth/Fibre

Recommendation for hard disk size depicts minimum value only. When choosing a hard drive factors such as RPM, Seek Time, Latency and number of spindles should be taken into consideration.

Table 3

Storage for Microsoft SQL Server [2]

Storage Solution	Technology	SQL Db. Size [GB]	Concurrent No. of Users	No. of Disks for Data	No. of Disks for Transaction Log
DAS	SATA	≤ 20	≤ 5	2 × 36 GB 7,2K RPM RAID 1	2 × 72 GB 7,2K RPM RAID 1
DAS	SATA	21÷40	6÷25	4 × 36 GB 7,2K RPM RAID 0+1/RAID 10	4 × 72 GB 15K RPM, U320 RAID 1
DAS	SCSI	21÷40	26÷50	4 × 36 GB 15K RPM, U320 RAID 0+1/RAID 10	4 × 72 GB 15K RPM, U320 RAID 1
SAN	Fiber/SCSI	41÷80	51÷150	6 × 36 GB 15K RPM RAID 0+1/RAID 10	8 × 72 GB 15K RPM, U320 RAID 1
SAN	Fiber/SCSI	> 80	151÷250	14 × 36 GB 15K RPM RAID 0+1/RAID 10	10 × 72 GB 15K RPM, U320 RAID 1

Files distribution depends on number of disks. A suggested approach is to put the Navision standard tables in the Primary Data File and SIFT tables in the Additional Data File.

Table 4

Recommended Data Distribution [2]

No. of Disks	Disk 1	Disk 2	Disk 3	Disk 4	Disk 5
3	OS Program Files TempDB	NAV Data File	SQL Trans. Log	–	–
4	OS Program Files TempDB	NAV Primary Data File	NAV Add. Data File	SQL Trans. Log	–
5	OS Program Files	TempDB	NAV Primary Data File	NAV Add. Data File	SQL Trans. Log

4. SQL Server administering

Database server maintenance is a day-to-day task for every administrator. There are many server and database parameters and tools which can speed up a system. In that article only that are described which directly affect SQL with Navision cooperation:

- Auto shrink – in both SQL Server 2000 and SQL Server 2005 auto shrink parameter should be set as false. If necessary it is recommended to set up a database shrinking in a scheduler to be run overnight.
- Update statistics – statistics affect an optimizer and speed up queries. Some administrators switch off auto update statistics parameter and set it up as an overnight routine.

- Files distribution – detailed description in point 3.
- Index rebuild – indices should be rebuilt periodically. Parameters like FILLFACTOR and PAD_INDEX can be used when an index is being rebuilt. FILLFACTOR determines the percentage of space on each leaf-level page filled in data by a server [10]. PAD_INDEX specifies the space left on each page in the intermediate levels on the index [10].
- Index hinting – index hinting can help to avoid situations when query optimizer chooses an index that requires many page reads and generates time consuming queries. It helps to hint a query optimizer to choose a "better" index. When index hinting is used Navision adds commands to the SQL queries that are sent to the server. These additional commands bypass the normal decision to the Queries Optimizer and force the server to choose a particular index [1]. Index hint syntax is:

  ```
  IndexHint=<Yes, No>; Company=<Company_name>; Table=<Table_name>;
  Key=<Key_field1, …, KeyFieldn>; Search Method=<search_method_list>;
  Index=<index_id>
  ```

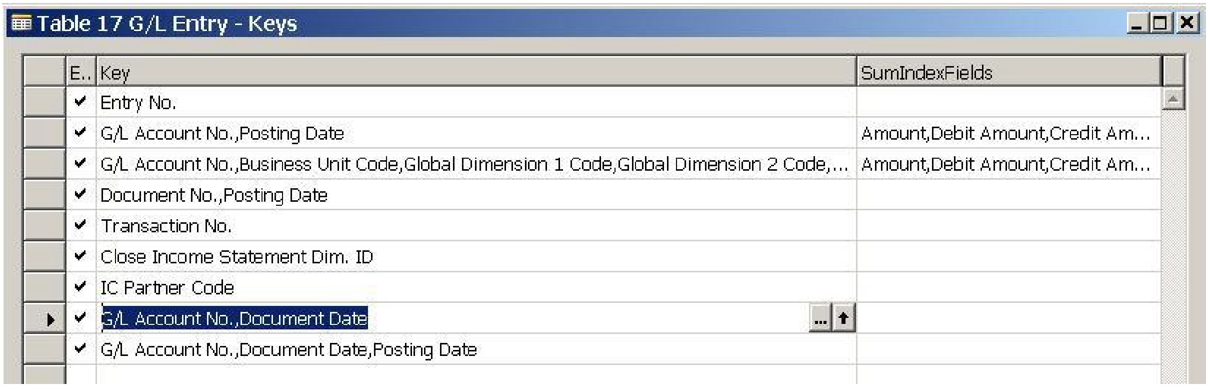
  Each parameter keyword can be found in stx file in Navision directory in "Drive Configuration Parameters" section. Index hint will be ignored when IndexHint=No and/or a keyword in a query is missing or incorrect. Search method contains a list of methods used in C/AL FIND/GET statement: "-", "+", "=", "!". Index ID corresponds to a SQL server index for a table: 0 is for a primary key. Index ID can be found by sp_helpindex stored procedure.
- DefaultLockGranulity – if this parameter is false then SQL Server places rowlocks, when is set as true SQL Server will choose its own granulity (page, table, row) [11].
- Clustered and nonclustered index – clustered index sorts and stores the data rows in the table or view based on their key values. Only one clustered index can exist in a table. By default a primary key is set as clustered index.

5. Programming recommendations

Carefully planning the details of an application will help ensure that a database has the best possible design. Properly designed application is much easier to build and maintain. This section does not focus on well-known methodology of analysis, design and implementation, but gives many hints and guidelines regarding best practises in C/AL programming and Navision administering.

Proper keys and indices definition belongs to the features with high influence on the system behaviour. Developers should remember not to define too many keys, because it slows down records manipulation. System has to update a record itself and all related indices during modification, insertion or deletion. All unused keys should be deleted.

Some keys are unnecessarily duplicated.



E..	Key	SumIndexFields
✓	Entry No.	
✓	G/L Account No.,Posting Date	Amount,Debit Amount,Credit Am...
✓	G/L Account No.,Business Unit Code,Global Dimension 1 Code,Global Dimension 2 Code,...	Amount,Debit Amount,Credit Am...
✓	Document No.,Posting Date	
✓	Transaction No.	
✓	Close Income Statement Dim. ID	
✓	IC Partner Code	
✓	G/L Account No.,Document Date	
✓	G/L Account No.,Document Date,Posting Date	

Fig. 2. G/L Entry key definition
Rys. 2. Definicja kluczy w tabeli G/L Entry

In the example shown in fig. 2 the key marked blue is unnecessary and should be deleted. The key composed of the fields "G/L Account No." and "Document Date" is included in the key "G/L Account No.", "Document Date", "Posting Date". SETCURRENTKEY statement will work correctly even if the key is deleted.

```
SETCURRENTKEY("G/L Account No.", "Document Date");
```

For queries using filters a correct key should be used. In below example the query works much faster when a programmer uses SETCURRENTKEY() statement. Filters (SETRANGE) in that case will be applied on index due to the chosen key. C/AL query:

```
SETCURRENTKEY(Field1, Field2);
SETRANGE(Field1, Code1);
SETRANGE(Field2, Code2);
IF FIND('-') THEN
   ….
```

results with SQL query:

```
SELECT * FROM Table
WHERE Field1 = Code1 AND Field2 = Code2
ORDER BY FIELD1, FIELD2
```

where:

```
SETRANGE(Field1, Code1);
SETRANGE(Field2, Code2);
IF FIND('-') THEN
   ….
```

results with:

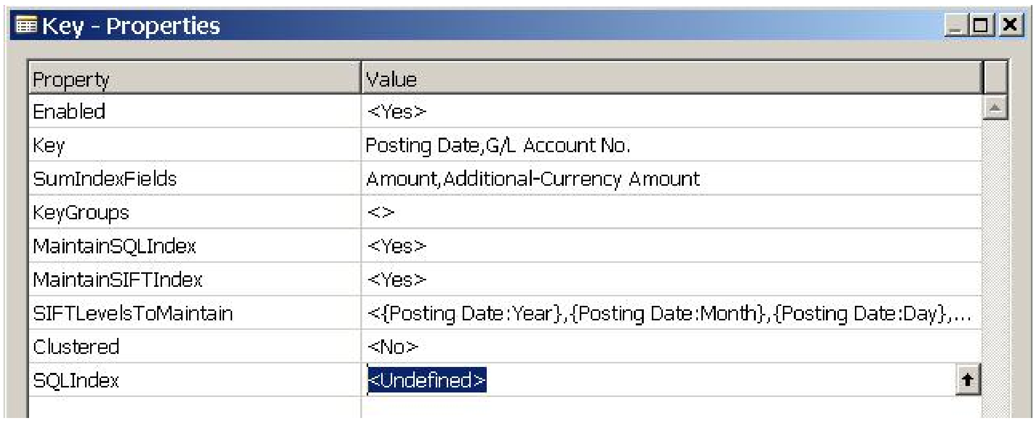
```
    SELECT * FROM Table
    WHERE Field1 = Code1 AND Field2 = Code2
```

ORDER BY clause forces query optimizer to use better index to process it.

When some Navision functionalities are not used it is recommended to delete related fields from keys and indices. For example if a company does not use item variants it is unnecessary to keep "Variant Code" field in keys.

In Navision key definition there are several parameters directly affecting locking and performance issues:

- MaintainSQLIndex – when that property is false a key in Navision will not force a SQL to create corresponding index. When it is true SQL Server will create an index for that key. SQL Server when performing a query often blocks a table and an index. When MaintainSQLIndex is false then SQL Server will have no index to be blocked for a key. This setting can slow down read statements but speed up a record update.
- SQLIndex – when MaintainSQLIndex is true then always a corresponding index is created. SQL Server does not have to use the same fields order as it is in the Navision key unless SQLIndex property is defined. This option is available for version 4.0SP1 and above.
- MaintainSIFTIndex – Sum Index Field Technology (SIFT) is an algorithm introduced by Navision for fast calculation of virtual fields. SQL Server stores SIFT structure in a special tables unless MaintainSIFTIndex is false.
- SIFTLevelsToMaintain – a developer can define which SIFT levels are stored and maintained by a SIFT table.
- Clustered – that permission helps set an index up as clustered or nonclustered. This option is available for version 4.0SP1 and above.



Property	Value
Enabled	<Yes>
Key	Posting Date,G/L Account No.
SumIndexFields	Amount,Additional-Currency Amount
KeyGroups	<>
MaintainSQLIndex	<Yes>
MaintainSIFTIndex	<Yes>
SIFTLevelsToMaintain	<{Posting Date:Year},{Posting Date:Month},{Posting Date:Day},...
Clustered	<No>
SQLIndex	<Undefined>

Fig. 3. Key properties definition
Rys. 3. Właściwości kluczy

Programmers have an access to keys definition, so they can activate and deactivate keys. But end users usually do not have a licence to change the table structure. “Key groups” is a very useful tool to maintain rarely used keys by a Navision user.

By making the keys members of key groups a user can activate and deactivate various combinations of keys in the tables by enabling and disabling the key groups respectively.

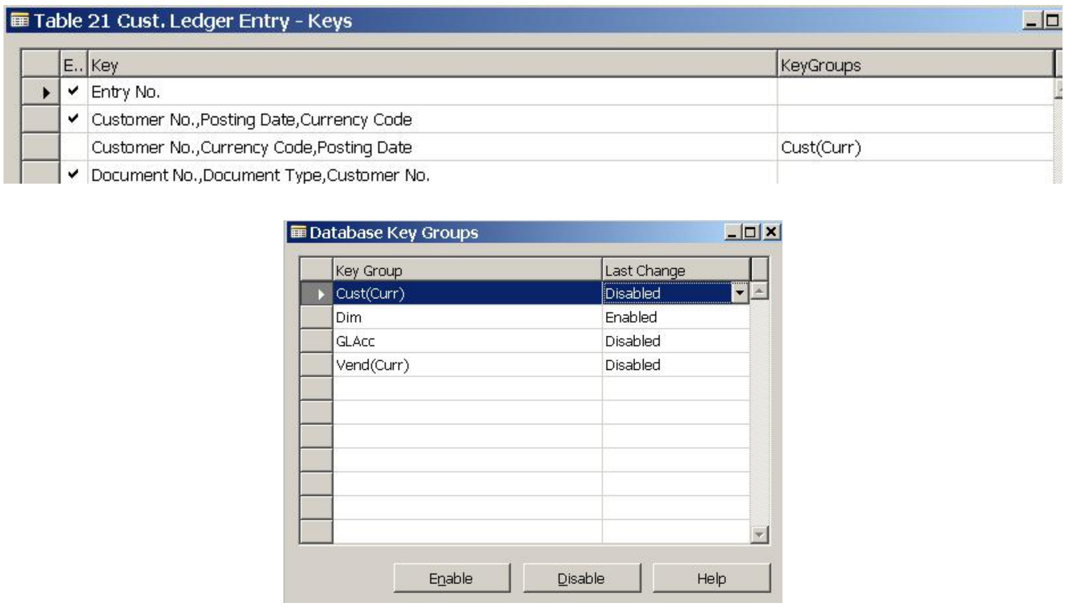


Table 21 Cust. Ledger Entry - Keys

E..	Key	KeyGroups
✓	Entry No.	
✓	Customer No.,Posting Date,Currency Code	
	Customer No.,Currency Code,Posting Date	Cust(Curr)
✓	Document No.,Document Type,Customer No.	

Database Key Groups

Key Group	Last Change
Cust(Curr)	Disabled
Dim	Enabled
GLAcc	Disabled
Vend(Curr)	Disabled

Fig. 4. Key Groups functionality
Rys. 4. Działanie funkcjonalności Key Groups

Very important clue is to keep locking order the same. Otherwise it is likely the users will experience the deadlocks. It will not prevent the system from deadlocks at all but will reduce them. Deadlocks happen when two processes block each other waiting for table release.

First process runs below commands locking at first Table1 and then Table2.

```
Table1.LOCKTABLE;
Table2.LOCKTABLE;
```

Second process analogically runs below sequence locking tables in reverse order:

```
Table2.LOCKTABLE;
Table1.LOCKTABLE;
```

When both processes are run in the same time, process no. 1 has to wait for Table2 to lock it and in the same time process no. 2 is waiting for Table1 which is blocked by process 1 and will not be released until Table2 is not free. That situation is called a deadlock.

LOCKTABLE command on SQL Server differs from NAV server. SQL Server applies table lock when reading a record. NAV server locks whole table when explicit LOCKTABLE was used or transaction starts. By default transaction isolation level is readuncommited. LOCKTABLE may use two parameters both Boolean type. If first parameter is true a server waits until pending transaction is unlocked. Second parameter checks a version of the record before it is updated.

Temporary tables help avoiding deadlocks as well. Temporary tables allow updating data in some protected triggers, e.g. OnAfterGetCurrRecord on form. They are processed on a client computer not on a server, so there is always only one copy of a temporary variable pointing the real table.

Navision in most cases when performing a query sends all data from and to the server. C/SIDE client filters them out, sorts them and process to return a result. C/AL commands like INSERT, MODIFY, DELETE are all processed by a client and afterwards results are sent to the server. Code like:

```
Table.SETRANGE(FieldX, ValueX);
IF Table.FIND('-') THEN
  REPEAT
    Table.FieldY := ValueY;
    Table.MODIFY;
  UNTIL Table.NEXT = 0;
```

can be replaced by:

```
Table.SETRANGE(FieldX, ValueX);
Table.MODIFYALL(FieldY, ValueY);
```

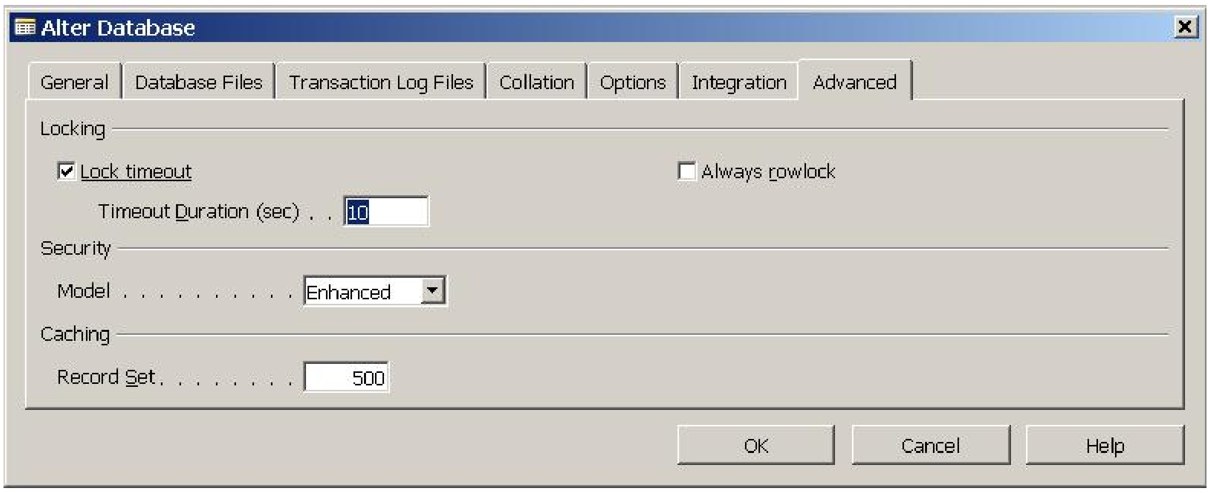
which is processed by a server. It helps to avoid sending all data by network to clients. Similar command DELETEALL should replace DELETE wherever possible. Command REPEAT … UNTIL Table.NEXT(<Step>) = 0; blocks current record and one before and one after the current one. That situation increases a risk of deadlocks which can be reduced by MODIFYALL/DELETEALL.

In Navision there are several of commands to search for a record. One of them is FIND which scans the table basing on current filter (SETRANGE/SETFILTER) and sorting key (SETCURRENTKEY). FIND returns true when a record was found and false in a contrary situation basing on a parameter which defines a search method. FIND('-') finds first record and FIND('+') tries to find last record matching to the selection criteria. Disadvantage of FIND command is the fact that it results with a full scan of a table: SELECT * FROM Table even if it finds only one record. For FIND command a cursor is created.

In Navision version 4.0 SP1 and above there are new commands FINDLAST, FINDFIRST and FINDSET which do not scan all table but only selected row set. For example FINDFIRST and FINDLAST statements are equal to SELECT TOP 1. These new commands do not create cursors.

The command ISEMPTY checks if a table or a record set is empty. That command does not create any cursor and result with SELECT TOP 1.

For version 4.0 SP1 and above some new database parameters are introduced.


Alter Database

General | Database Files | Transaction Log Files | Collation | Options | Integration | Advanced

Locking

☑ Lock timeout ☐ Always rowlock

Timeout Duration (sec) . . 10

Security

Model . . . . Enhanced

Caching

Record Set 500

OK | Cancel | Help

Fig. 5. Alter Database window
Rys. 5. Zmiana parametrów bazy danych

- Lock timeout – this setting allows a user specify whether a session will wait to place a lock on a resource that has already been locked by another session.
- Timeout Duration – this parameter allows a user specify the maximum length of time that a session will wait to place a lock on a resource that has already been locked by another session. The default value is 10 seconds. If this option is left unchecked, the session will wait indefinitely.
- Always rowlock – this setting allows specify that Navision always places row-level locks instead of page- and table-level locks.
- C/AL command LOCKTIMEOUT(False) – which allows temporarily change Lock timeout property.

Navision allows optimizing tables (menu File, Database, Information, Tables, Optimize). Its internal optimization process works in two ways [1]:

- for each table, SQL indices other then primary key are rebuilt to optimize their layout and usage,
- for each SIFT structure, all entries containing zero values in all numeric fields are removed.

6. Summary

Proper database server administering and maintenance is a vital key to a correct and a long life of the server and its applications. Optimization and daily routines help avoid deadlocks, shorten database processes and transactions.

Optimization is composed of both hardware and software issues. Devices can usually be easily and pretty fast replaced. Hardware upgrade is done every certain span of time. An improperly designed database and an application have much more negative consequences and their rebuild is more time consuming, expensive and very often impossible. According to the statistics application causes 80÷90% of performance problems. Problems caused by infrastructure vary between 10÷20%. In application design phase developers and designers should consider future data growth, coding plans, optimization issues and customer expectations.

Recent Navision versions work with three types of servers: Microsoft Dynamics NAV Database Server, Microsoft SQL Server 2000 and Microsoft SQL Server 2005. Microsoft did some performance tests which proved the fastest cooperation with SQL Server 2005. SQL Server 2000 and NAV Server had fairly similar accomplishments. Navision cooperating with SQL Server 2000 uses NDBCS driver which uses temporary cursors to simulate native (NAV) server.

BIBLIOGRAPHY

1. Microsoft Business Solutions Navision 4.0 Course: 8404B Installation and Configuration Training. Microsoft Corporation, 2004.
2. Dupont-Roc P., Gatchalian R., Serrano D.: Microsoft Dynamics NAV 4.0 Hardware Guide. 2006.
3. Muhlbaher H.: Optimizing Dynamics NAV on SQL Server – Application. 2007.
4. Muhlbaher H.: Optimizing Dynamics NAV on SQL Server – Infrastructure. 2007.
5. Raheem M., Sonkin D., D'Hers T., LeMonds K.: Inside SQL Server 2005 Tools. Addison Wesley Professional, Boston 2006.
6. Woody B.: Administrator's Guide to SQL Server 2005. Addison Wesley Professional, Boston 2006.
7. Rankins R., Jensen P., Bertucci P.: Microsoft SQL Server 2000. Księga eksperta. Helion, Gliwice 2003.
8. Bieniek D., Dyess R., Hotek M., Loria J., Machanic A., Soto A., Wiernik A.: Microsoft SQL Server 2005 Implementation and Maintenance, Self Paced Training Kit. MSPress, Redmont 2006.
9. Nielsen P.: SQL Server 2005 Bible. Wiley Publishing, Inc., Indianapolis 2007.
10. Thomas O., McLean I.: Optimizing and Maintaining A Database Administration Solution by Using Microsoft SQL Server 2005. Microsoft Press, Redmont 2006.

11. Microsoft Dynamics NAV 5.00 Application Designer's Guide. Microsoft Corporation, Redmont 2007.

Recenzent: Dr inż. Paweł Kasprowski

Wpłynęło do Redakcji 3 kwietnia 2007 r.

Omówienie

Obecne wersje Microsoft Dynamics NAV oprócz serwera natywnego pozwalają na współpracę z MS SQL Server 2000 oraz MS SQL Server 2005. Prawidłowa administracja serwerem oraz systemem Navision umożliwia optymalizację zapytań oraz manipulację danymi.

Płaszczyzny administracji można podzielić na 3 grupy:

- zarządzanie sprzętem,
- administracja SQL Server,
- administracja systemem Navision.

Sama decyzja zakupu sprzętu powinna uwzględniać planowane wdrożenie, w szczególności obszar systemu ERP, który będzie używany, liczbę wszystkich użytkowników systemu oraz przewidywaną liczbę równoległych, konkurencyjnych sesji. Należy także brać pod uwagę możliwy przyrost bazy danych. Im mocniejszy sprzęt, tym z jednej strony lepsze osiągi systemu, ale też większe koszty.

Przy większych bazach danych należy rozważyć zainstalowanie i wdrożenie hurtowni danych, po to aby ona odciążyła bazę OLTP przejmując większość zapytań. W niektórych przypadkach instaluje się kolejny serwer OLTP, który przechowuje tylko zapisy archiwalne. Wtedy hurtownia zbiera dane zarówno z bazy aktualnej, jak i archiwalnej.

Rutynowe prace administratora serwera oraz systemu Navision obejmują między innymi śledzenie logów, pilnowanie blokad, zmianę parametrów wraz ze wzrostem bazy danych. Istnieje szereg parametrów oraz narzędzi, które pozwalają monitorować pracę systemu.

Ostatnim, ale równie ważnym elementem wpływającym na wydajność systemu jest odpowiednio zaprojektowana i napisana baza danych. Prawidłowe nawyki programistyczne, takie jak używanie tabel tymczasowych czy zachowanie odpowiedniej kolejności dostępu do tabel, pozwalają przyspieszyć pracę systemu oraz uniknąć zbędnych blokad i zakleszczeń. Istotnym elementem jest odpowiednie zaprojektowanie kluczy, indeksów, pamiętając, że zbyt

ich duża liczba spowalnia usuwanie, dodawanie i modyfikowanie rekordów. Należy także używać opowiednich filtrów i kluczy sortujących przy zapytaniach.

Testy wydajności przeprowadzane przez Microsoft z użyciem pakietu Application Benachmark Toolkit pokazały, że najwyższa wydajność została osiągnięta przy Microsoft SQL Server 2005. SQL Server 2000 oraz NAV Server mają mniej więcej podobne osiągi.

Adres

Marcin WOCH: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16,
44-100 Gliwice, Polska, marcinwoch@wp.pl.