Acta Academiae Paedagogicae Agriensis, Sectio Mathematicae **31** *(2004) 69–77*

EXAMINATION OF THE MSSQL SERVER FROM THE USER'S POINT OF VIEW CONSIDERING DATA INSERTION

Tibor Radványi (Eger, Hungary)

Abstract. In this paper we summarize the experiences of the partial effectiveness examination made on the MSSQL server. We examined the effectiveness of the insert sample databases on the server. The client program was written in C++ language, in the Visual.NET system. We have done the examination of the data insert both from single- and multiclient environment. The examination contains insert options of the ADO.NET subsystem - given by the .NET system - and insert options of stored procedures that were stored on the MSSQL server. These comparisons were extended with the analysis of the different network speed environments. Tests were made on high - speed intranet and on Internet, ADSL (512 kbs), connection. We think that the profound and various examination of the database servers is very important. Here we relate test results that can be usable either in research in connection with database servers or in practical usage of the same systems.

AMS Classification Number: 68P30, 68P10

1. Introduction

The testing of the database systems and the measuring of their effectiveness has an important part in today's research fields [1]. When talking about systems with great data - traffic, the insert of data is an especially resource-required operation. In the case of the benchmark test both the server's and the client's software options must be kept in view [2], [3]. Our test expressly closes and examines the functioning of the database from the client's side. Comparison means the collation of the different opportunities given by the programming environment during the creation of the client software. The first task is the recording of environment that influences the test results - such as the size of the DataSet, the complexity of the SQL commands, hardware/software environment, the expandantion and capability of the network [1]. The goal of our test is to compare two data - uploading methods given by the new .NET technology. We can only do this with an appropriately built program on the client's side and with the measuring of the results on the client's side. The client program was made in the Visual.NET system, in C++ language. To reach the database, the ADO.NET technology is a new and effective tool. In order to get best performance we used the Microsoft's recommendations and research results [2], [5].

2. Hardware and software systems that were used during the test

A server computer that was indispensable for the test was installed on the Computing Department of the Károly Eszterházy College. This machine gave us the opportunity that show acceptable performance on the server's side and don't go off from the opportunities given by ensured by real user environments. As the formation of the current environment greatly influences the test results, the most important information for us are not the exact time information, but the differences shown between usage of the different programming tools, so we have to examine the proportion of the measured time values. The parameters of the server and the client machines can be found in the appendix. The program development was done on the C++ language that is part of the Microsoft Visual Studio .NET 2003. The database can be found on the Microsoft SQL Server 2000 Enterprise Edition.

3. The database

During the test the following database has been used:

Subtables: these simple tables contain basic data that are used for the random filling of the table with the subscriptor's data: (sHelysegnev, sVezeteknev, sKeresztnev, sUtcanev). These have no role in the test, they help to create the appropriate environment. As this system is a simplified model of a real system, the starting data - information about the subscriptors - are generated by a procedure that uses the subtables as a help. These subtables do not contribute to the database on the classical way, they do not take part in the test, they have no influence on it's results so their connection to the database through keys and references is superfluous and harmful. What still indicates their usage is the nearly 10 million generated record, that can be used later to test requests and to get readable results and lists that are true to life. The test is influenced by the data of the following tables, these are the ones that give results.

Elofiz: stores the data of the telephon company's subscriptors. These data will be generated by the help of the subtables.

Fields:

ID	Int (Identity)	The subscriptor's unique identifier. A serial number given by the system.
Vnev	Varchar(25)	The subscriber's family name (from the SVezeteknev table)
Knev	Varchar(20)	The subscriber's christian name (from the Skeresztnev table)
Lakhely	Varchar(25)	The city where he lives (from the SHelysegnev table)
Utca	Varchar(25)	Street (from the SUtcanev table)
SzulDatum	Datetime	Date of birth
SzemIg	Char(8)	ID Card's number (a randonly created series of characters)

Telszam: phone numbers that belong to the costumers
Fields:

Tszam	Char(12)	Unique phone number
IDElofiz	Int	The ID of the subscriber, foreign key, have connection with the Elofiz table. The connection between the two tables is one-more, as one subscriber may have more phone numbers, but one number can only belong to 1 subscriber.

Hmod: Type of call (line, cell, inland)
Fields:

ID	Int (Identity)	Unique identifier, primary key, a serial number given by the system.
Típus	Varchar(20)	To differentiate the many district numbers in the case of cell phones and to differentiate the line phone. Hungarian specific.
Cel	Varchar(1)	The destination of the inland or foreign call.

Forg: The traffic table that serves as the base of test, storing information about the calls. Approximately 10 million items were inserted into this table during the test. It'll will have a basic role during the experimentation of the requests.

Fields:

ID	Bigint (Identity)	Primary key, a serial number given by the system.
IDTszam	Char(12)	The costumer's phone number.
IDHMod	Int	The type of the call, foreign key to the HMod table, holds the connection between the tables.
Hszam	Char(12)	The called phone number
Hkezd	Datetime	The time of the call's begining
Hbef	Datetime	The time of the call's end
Hido	Int	The time period of the call, it's value is counted by a trigger.

LogTab: We store the results of the tests in this table. The system automatically generates a record for every test in this table.

Fields:

ID	int (Identity)	Primary key, a serial number given by the system.
Midopont	Datetime	The costumer's phone number.
Mkezd	Datetime	The tests beginning date and time
Mbef	Datetime	End of test
Mido	Float	Time period of the test
MtipSQL	Char(10)	The type of SQL command that we test. In our case, the type is 'INSERT'
Rekordszam	Bigint	The number of inserted records during the test.
TriggerAll	Bit	Counter to show that if every trigger was active or not. It is a factor in the system's load
Mtip	Char(10)	Type of test
Gepszam	Smallint	Number of machines in the test
Cel	Char(10)	Destination datatable, Forg in our case
Modszer	Char(10)	StoredProc/ADO comparison

Triggers: two triggers belong to the 'forgalom' table:

`forg_hmod`: Sets the time and type of the call after the record was inserted. It worth using the automatic data-definition as it can reduce the network's data-traffic.

`forg_hbef`: Counts the call's the time period after the 'Hívás befejezése' field was filled, than it puts it to the record's appropriate column.

4. The program

The client program's technology uses the latest Microsoft development, the Visual.Net system. The software was written on C++ language, that gives a flexible tool to do the appropriate tests.

As our test included the Microsoft MSSQL server's data-insert partition, we chose the DataSet solution from the options of the DataReader on-line read-only connection and the DataSet off-line solution. The DataSet class' communication with the SQL server is well represented by the picture below. The program in its current state - from the tests' view point - uses two different datahandling method. One amplifies the Rows Collection of the SqlDataSet's DataTable class given by the ADO.NET frame with new records and at the end of the amplification, it uses the SqlDataAdapter class' Update method to actualise the content of the database. The other does the same by using stored procedures. Practically, holding the connection with the database lays on ADO.NET bases in both cases, but in the last case the procedures stored on the server are responsible for the uploading that we call with parameters by the SqlCommad class' help. When using a stored procedure for uploading we only need the SQLCommand class with right parameters and the running of the command. So the goal of the test is to compare the two data - uploading methods given by the new .NET technology. We can only do this with an appropriately built program on the client's side and with the measuring of the results on the client's side. The test includes the examination of the whole system, as it'll seem from the results shown later, the results are unambiguously and consistently influenced by the speed of the network and the server's software and hardware preparation. As our goal is the test on the client's side, the results are valid to this given system. Inasmuch as we would only test the performance of the SQL server, we could only make test with programs run on the server to exclude the clients and the network. This is possible, but the goal of the article is not that. The test of the two methods was our goal, and we'll show the results of these now.

5. Tests and results

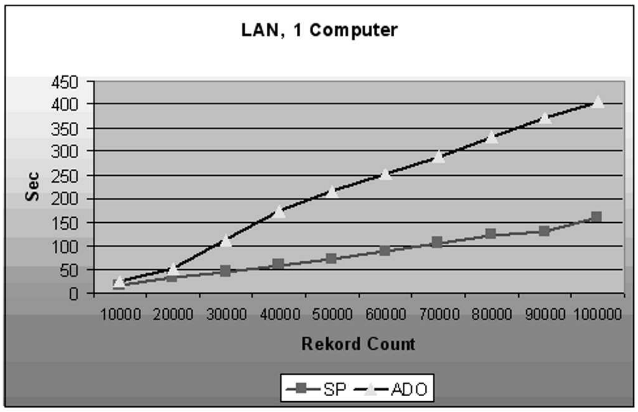
With the tests, we kept in view that many factors may influence the results due to the complexity of the system. A test result row starts with the selection of given method (Stored Procedures (SP) or DataSet (ADO)) and with the definition record's number that will inserted. We repeat such a test for fifty times to exclude errors. We did approximately 800 tests with the different record numbers. The test results went through an examination before they were averaged and the once or twice occurred extreme results didn't get in to the average. These deviations always had cause that was independent from the test (hardware error, non-planned load on the server). We stopped all other resource - requiring processes on the server for the test's duration. No other SQL servers (Oracle, MySQL) were running. This was the way we tried to ensure the most undisturbed conditions.

Signs, abbreviations:

Rcount:	number of inserted records;
SP:	usage of Stored Procedures;
ADO:	usage of DataNet.

(a) Local Area Network, one client machine (results in seconds)

Rcount	SP	ADO
10000	14.26565	24.2969
20000	30.9583	52.224
30000	44.401	113.5
40000	59.4896	174.3
50000	71.32825	216.016
60000	89.25	289.2373
70000	106.37	330.556914
80000	121.474	371.876529
90000	129.271	406.723
100000	159.161	289.2373



The curve that took shape can be approximated by a linear equation, where, from the

$$y = mx + b$$

equation, we examine the value of the m parameter compared to each other. We did the definition of the equation with the method of the smallest squares, that's how we fit the line on the measured value pairs. The results from this count:

$$mSP = 0.00150933 \quad mADO = 0.00411515$$

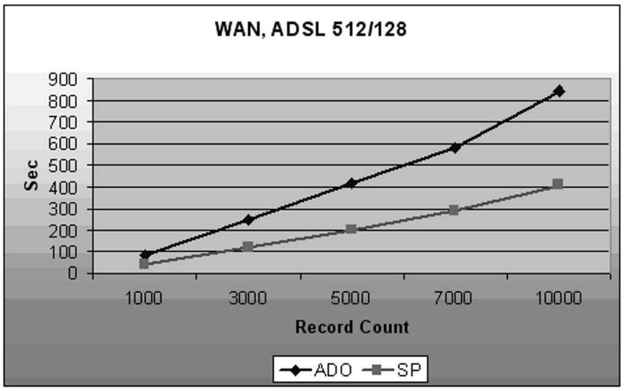
As the graph shows the usage of the stored procedure is more even amd have a better rate of effectiveness:

$$M = mADO/mSP = 2.7265$$

This shows that the usage of the stored procedure, in this case, gives a three times faster speed than the DataSet class ensured by the ADO.NET as a tool. An important note: If we would use the Update method not after the creation of the full record group in the memory, but after each and every record, this number could grow to a 100 times bigger. So if we are inserting thousands of records and the momental actualisation is not a must, than we should do it after the inserting of the records, but at least after greater groups.

(b) WAN network, through ADSL connection

Rcount	ADO	SP
1000	82.8625	39.9775
3000	248.073	120.266
5000	415.618	200.004
7000	583.255	286.318
10000	847.462	407.74



The curve that took shape can again be described with a linear equation. After the counting, the following factors remain:

$$mSP = 0.040665 \quad mADO = 0.084036$$

As the graph shows, the usage of the stored procedure is more even and have a better rate of effectiveness:

$$M = mADO/mSP = 2.06652$$

The redundancy of the rate of effectiveness can be influenced by the different speed of the network and by it's stability.

6. Conclusions, further directions

The programming of databases, it's access from application sofwares is a wide - spread and major problem in many places that occurs in many fields of live. The first step of handling data - their storage - a method that occurs in every system, uses great resources from the given frame at some places. Our goal with this test was to examine the reducing possibilities in the case of a wide spread system. The test results unambiguously supports that the system's inserting effectiveness can be greatly improved if we use the options given by the SQL servers, the use of the stored procedures, even in the case of such tasks that seem to be easily solved by other methods. We will expand the examination of the insert method to the Oracle, the IBM DB2 and to the Interbase SQL servers. We will not only do this by comparing the different methods, but will also compare the test results to find the most effective data - insert method on the above mentioned servers. For a more flexible and easier handling, we also need to upgrade the client program written in C++ language. It'll be a task to create different classes for the different database-handling devices, for the different methods. All classes must have the same procedures for in the main program, we only need to use an object of the appropriate class instead of the conditional, that are getting more and more complex. The timing system should be altered to a form, where the timing should not be set again and again on each and every machine, be we only need to put them into timing mode. The actual timings would appear centrally in the database, and the timed programs would continuously check if there is a task for them. This would greatly improve and make the testing easier, even in the case of a small number of computers, and it is obligatory for a large number of clients.

Appendix

Server(dragon.ektf.hu)
Processor type: 2 db Intel Pentium III Xeon
Memory: 1024 MB
HDD: 2 db SCSI controlled, 30 Gb size, no Raidbe
Operating system: Microsoft Windows 2003 server
Database server: Microsoft SQL Server Enterprise Edition
Version Number: 8.00.760 (SP3)

Workstation
Processor: Intel Pentium 4 (1600 MHz)
Memory: 256 MB
HDD: 1 db 40 Gb size, IDE controlled 7200 turn/min
Operating system: Microsoft Windows XP professional SP1

Network
Internal network: 100 Mbps, DHCP, DNS options
External network: 512 Kbps ADSL, DHCP and DNS options

References

[1] Ailamaki, A., Shao, M., DBMbench: Microbenchmarking Database Systems in a Small, Yet Real World in Confidential, (*submitted to ICDE 2004*).

[2] Microsoft Co.: Improving .NET Application Performance and Scalability, (2004), 639–682.

[3] Ruthruff, M. (Microsoft Co.), Microsoft SQL server 2000 Index Defragmentation Best Practices, 2003.

[4] Gray, J., *The Benchmark Handbook for Database and Transaction Processing Systems*, Morgan Kaufman Publishers, Inc. 2nd edition, 1993.

[5] Gray, J., `http://research.microsoft.com/gray`.

Tibor Radványi
Department of Computing
Károly Eszterházy College
Leányka str. 6.
H-3300 Eger, Hungary
E-mail: dream@aries.ektf.hu