ISSN: 2393-8528

IJICSE
Open Access

Contents lists available at www.ijicse.in

International Journal of Innovative Computer Science & Engineering

Volume 1 Issue 2; Page No. 09-11

Examining the Two Data Access Techniques: Stored Procedure and Dynamic SQL

Deepika Jangid

Assistant Professor, Arya Institute of Engineering & Technology, India

ARTICLE INFO

Received 10 Oct. 2014
Accepted 20 Nov. 2014

Corresponding Author:

Deepika Jangid

1 Assistant Professor, Arya Institute of Engineering & Technology, India

ABSTRACT

A stored procedure is a group of SQL statements that form a logical unit and perform a particular task, and they are used to encapsulate a set of operations or queries to execute on a database server. For example, operations on an employee database (hire, fire, promote, lookup) could be coded as stored procedures executed by application code. They perform mid-way processing on the database server and so exclude transmission of excess data across the network. Dynamic SQL is a programming technique that enables you to build SQL statements dynamically at runtime. You can create more general purpose, flexible applications by using dynamic SQL because the full text of a SQL statement may be unknown at compilation. This paper states the comparison and investigation of two relational database access mechanisms i.e. Stored Procedures and Dynamic SQL. Stored procedures when implemented resulted in speedy database operations in comparison to dynamic SQL over the Internet.

1. INTRODUCTION

A stored procedure is a subroutine available to applications that access a relational database system. Stored procedures (proc, sproc, StoPro, StoredProc, or SP) can be defined as collection of subroutines. They are stored in the data dictionary of the database. An application uses this data dictionary to perform database related tasks. Stored procedures can be used for data validation (integrated into the database), access control mechanisms and centralize logic that was originally implemented in applications [2]. The control to a stored procedure on the database server is passed by a client application.

A stored procedure is a group of SQL statements that has been created and stored in the database. Stored procedure will accept input parameters so that a single procedure can be used over the network by several clients using different input data. Stored procedure will reduce network traffic and increase the performance. If we modify stored procedure, all the clients will get the updated stored procedure. Stored procedure performs intermediate processing on the database server, and so reduces transmission of spare data across the network. Only the records that are actually required by the client application are transmitted. If required, we can also use Nested stored procedures by executing one stored procedure from within another. Stored procedures seem similar to user-defined functions a lot. The only major difference between the two is that user defined functions can be used like expression within SQL statements, whereas stored procedures needs to be invoked by user [1]. The technique of using stored procedure in database avoids duplicating database code, saving time and effort when you make updates due to schema changes, tune the performance of queries, or add new database operations for logging, security, and so on.

Dynamic SQL refers to SQL code that is generated within an application or from the system tables and then executed against the database to manipulate the data. The SQL code is not stored in the source program, but rather it is generated based on user input. This can include determining not only what objects are involved, but also the filter criteria and other qualifiers that define the set of data being acted on. Using this approach, we can develop powerful applications that allow us to create database objects and manipulate them based on user input. This can be a great time saver because you can: Automate repetitive tasks, Write code that will work in any database or server, and Write code that dynamically adjusts itself to changing conditions.

Dynamic SQL is an enhanced form of Structured Query Language (SQL) that facilitates the automatic generation and execution of program statements i.e. Dynamic SQL

is the SQL code that is generated programmatically by the application before it is executed [3].

2. Related Work

Stored procedures provide a strong and good way for coding the application logic that can be stored on the server. Stored procedures are popularly used in Informix Dynamic Server and Oracle both. Oracle also uses functions. A function is a type of subprogram designed to perform a special task. The language used to code stored procedures is a database-specific procedural extension of SQL. In Oracle it is PL/SQL and in Informix Dynamic Server it is Informix Dynamic Server Stored Procedure Language (SPL). These languages differ considerably. However, most of the individual SQL statements and the procedural constructs, such as if-then-else, are similar in both languages [2].

Triggers provide a way of executing PL/SQL code [2] on the occurrence of specific database events. For example, you can maintain audit logs by setting triggers to fire when insert or update operations are carried out on a table. The insert and update triggers add an entry to an audit table whenever the table is altered.

Dynamic SQL is an extended form of Structured Query Language (SQL) [3]. Unlike Static SQL, Dynamic SQL facilitates the automatic generation and execution of program statements. This feature can be helpful when there is need to write code that can be set for different databases, conditions, or servers. It also makes it easier to automate tasks that are repeated many times.

Dynamic SQL statements are stored as strings of characters that are entered when the program runs. They can be entered by the programmer or generated by the program itself, but unlike static SQL statements, they are not embedded in the source program. Also in contrast to static SQL statements, dynamic SQL statements can change from one execution to the next.

Dynamic SQL statements [3] can be written by people with comparatively little programming experience, because the program does most of the actual generation of the code. A potential problem is reduced performance (increased processing time) if there is too much dynamic SQL running at any given time.

3. Experimental Part

There are various methods to carryout database operations in web application built on .Net platform that uses Microsoft SQL Server 2008. Two of the most common approaches are to dynamically generate SQL statements and stored procedures. Does either of the two give any performance gain?

1) Purpose

The experiment was carried out with an aim to compare the query execution time of SQL select statements generated dynamically with that of an equivalent stored procedure from within a web application developed in .NET environment.

2) Functionality

Experiment was conducted by keeping in mind that the functionality provided by both dynamic SQL statement and stored procedure is exactly the same. It was also essential that the execution time of a variety of statements should be compared. For each SQL statement an equivalent stored procedure were executed against the sample database. The three processes are:

- Select all data of a column from a table.
- Select a subset of rows from a table using where clause.
- Select a subset of rows from a table using where and like clauses.
- Select a subset of rows from a table using where and between clauses.
- Select a subset of rows from multiple tables using join and order by clauses.

The SQL statements were held in stored procedures with a parameter where required. These were matched identically in the C# code.

Since the execution speed of the statements are too fast and thus difficult to measure, each statement was executed reputedly five thousand times by constructing a loop and average execution for each statement was recorded.

```
SELECT * FROM Customers
SELECT Company, Country FROM Customers WHERE
Country <> 'INDIA'
SELECT * FROM Persons
WHERE City LIKE 'v%'
SELECT * FROM Persons
WHERE LastName
NOT BETWEEN 'Gupta' AND 'Yadav'
SELECT E_Name FROM Employees_Norway
UNION
SELECT E_Name FROM Employees_USA
SELECT Persons.LastName, Persons.FirstName,
Orders.OrderNo
FROM Persons
LEFT JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

4. Results

The table shows the average timings for each of the three statements. The first and second columns show the results for the dynamically created SQL statements and the stored procedures respectively. The third column shows the percentage difference between tests.

Table 1: Experimental results

S No.	Statements Type	Dynamic SQL	Stored Procedure	Difference
1.	Simple Select	3.24s	3.22s	0.61%
2.	Select with Where Clause	1.98s	1.92s	3.03%
3.	Select with where and like	2.34s	2.29s	2.13%
4.	Select with where and between	2.88s	2.84s	1.38%
5.	Select with Join and Order by	3.98s	4.06s	-2.02%

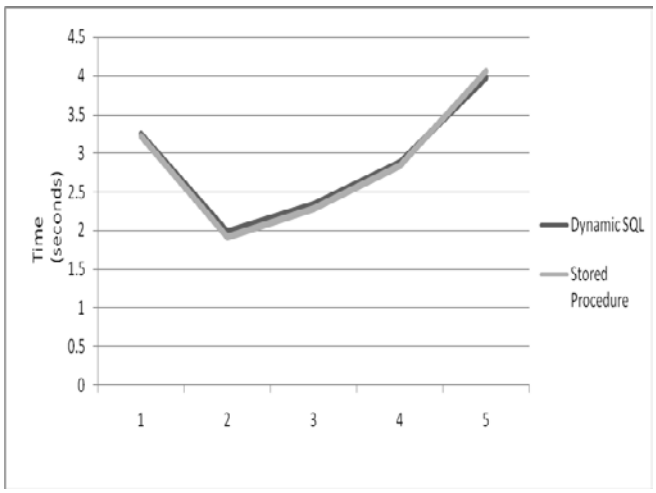

Figure 1: Graphical Analysis

REFERENCES:

1. Ke Wei, M. Muthuprasanna and Suraj Kothari (Iowa State University). 'Preventing SQL Injection Attacks in Stored Procedures' .Software engineering conference 2006.
2. Sunderic, Dejan. Woodhead, Tom. SQL Server 2000 Stored Procedure Programming. Berkeley:Osbourne/McGraw-Hill, 2001. p346-356, 466-526.
3. Russell A. McClure and Ingolf H. Krüger : SQL DOM: Compile Time Checking of Dynamic SQL Statements.
4. Chr*is Anley : Advanced SQL Injection In SQL Server Applications.*