

Home

Forums

General - Windows

Windows OSs

Windows Server System

Other Computing - Windows

Development

SQL Server

General - SQL

SQL Server Mag Site Redesign

Server 2000 / 7.0

SQL Server Yukon Beta

SQL Server 6.x

CertTutor

General - CertTutor

MCSE Exams

The Real World

Other Certifications and Technologies

Events

Web Seminars

Roadshows

Conferences

All Events

Hot Topics

Active Directory

Desktop

Exchange & Outlook

Migration

Mobile & Wireless

Networking

Oracle on Windows

Scripting

Secure Messaging Center

Security

Small Business Center

SQL Server Magazine

Storage

Web Administration

All Articles by Subject

Publications Archive

Windows IT Pro

SQL Server Magazine

Exchange & Outlook Administrator

Windows Scripting Solutions

Windows IT Security

Email Newsletters

All Publications

Recent Articles

The Smart Guide to Building World-Class Applications

Administration

Backup and Restore

Change and Configuration Management

Data Modeling

Data Transformation Services

Enterprise Manager

High Availability

Interoperability

Monitoring and Analysis

Performance

Query Analyzer

Querying

Replication

Security

SQL Server Profiler

Storage

Task Automation and Scripting

T-SQL

View All

Development

Business Intelligence

SQL SERVER

magazine

.NET, HTML, ASP, XML get tech jobs ASAP

We speak your language. Get a tech job today with the leader in tech jobs.

[September 28,
2000]

Expanding Hierarchies

Muhammad Nadeem Akhter

T-SQL Black Belt Exchange

InstantDoc #15715

Web Exclusive

DOWNLOAD THE CODE:

📄 [15715.zip](#)

Microsoft has expanded the information about expanding hierarchies in SQL Server *Books Online (BOL)* since SQL Server 6.x. But the *BOL solution* merely displays the output; you can't join the results with other views, tables, or results. My T-SQL coding solution fills this gap. Also, the solution I present here is 5 to 6 times faster than the one provided by *BOL* for SQL Server 2000. In addition, I'll show you useful SQL Server 2000 features such as templates, functions, and new table data types. One more thing I feel that it is difficult for SQL Programmers to understand the logic of LOOPS, IF structures and so on. So I present a flowchart, so you'll have better understanding of the solution.

What Are Hierarchies?

DBAs deal with hierarchies every day, whether the hierarchy is an organizational chart, a file directory structure, geographically dispersed sale points, inventory, etc. When you map these hierarchies in databases, you either use redundant data, such as in an inventory table where you might have attributes such as No, Head, Level1, Level2, Level3,..., LevelN or use a self-referencing table (e.g., an Employees table having an attribute ReportsTo or ManagerID. The first example uses repeating groups, which don't fall in the first normal form and hence don't fall in higher normal forms, whereas the latter solution gives you a scalable, normalized solution, which contains no redundancy. But this is a complex solution. It means we have to expand the hierarchies; only then it can the solution be useful for us. Let's consider a particular example from *Books Online (BOL)* under Expanding Hierarchies. Look at the indented view a parent-child relationship, which [Figure 1](#) shows.

How can you expand these hierarchies by using SQL? The easy answer is to go to *BOL* and look at the code under Expanding Hierarchies. [Listing 1](#) shows the code sample from *BOL*.

This code performs the following:

SQL Server Magazine Options

- [Subscribe](#)
- [Renew](#)
- [Master CD](#)
- [Monthly Pass](#)
- [Back Issues](#)
- [Article Reprint](#)
- [Customer Service](#)
- [Group Discount](#)

📅 Top 20 Articles

- 1 [Microsoft Abruptly Pulls MSN Messenger 7.0 Beta](#)
- 2 [Common .pst File Questions](#)
- 3 [What You Need to Know About Windows Server 2003 SP1](#)
- 4 [WinInfo Short Takes: Week of March 21](#)
- 5 [How can I uninstall the Microsoft Java Virtual Machine \(JVM\) from Windows XP?](#)

[More Top Viewed Articles](#)

📅 Related Events

- "Reliability and Resilience:  [Keys to Achieving High Availability and Disaster Recovery for Microsoft Servers"](#)
- [Microsoft Tech•Ed 2005](#)
- [SQL Server Summit in Minneapolis](#)
- [Visual Studio Connections](#)

ALL EVENTS >>

📅 Related Topics

- [Scripting](#)

1. Creates a temporary table – #stack item char(20), level int)
2. Declares the variable @level and assigns it the value 1
3. Inserts the parameter @current to the #stack table along with @level, which is initially 1
4. Runs Steps 5 and 6 in a loop While @level >0
5. If there is no row with level=@level, then decrease @level by 1
6. Otherwise
 - a. Pop one row and [print](#) it
 - b. Insert rows whose parent is @current into the #stack table
 - c. If any row was inserted, then increase @level by 1

So far, so good. However, you can use the code that *BOL* gives only to produce output on the display. You can't join the results other views, tables or results. In addition, you can improve the performance if you add non-leaf nodes to the #stack table as in step 6.b and insert leaf nodes directly to another temporary table called #hierarchy. The reason for the potential performance improvement is that if you look at any given hierarchical structure, you'll find more leaves than branches, whether it's a tree, a file directory structure or an inventory table, etc. To test this solution, I used data from my accounts receivables table, which encompasses almost 8000 rows and the response time was reduced to almost 15 percent compared with performance of the *BOL* technique. The *BOL* technique returned data in 35 seconds, and the technique I propose returned data in just 5 seconds.

Now, how can you return data from one stored procedure and use the data from a calling procedure? One possible solution is that your stored procedure might return data by using a select statement, and the calling procedure can use an INSERT SELECT statement to store the results in a table, as follows:

```
CREATE TABLE #temp(SerialNo int NULL,No int NULL, tName VARCHAR(255))
INSERT INTO #temp EXEC ExpandHierarchy 5
```

We can even improve the code above by having the calling procedure create a temporary table, and the called procedure populates it. The calling procedure can have the following code:

```
CREATE TABLE #Holder(SerialNo int,No int, Head varchar(300))
EXEC ExpandHierarchy 5
SELECT * FROM #Holder
```

The code is looking fine, but there's one more problem to fix: Users will have to remember the syntax to call the stored procedure. The good news is that if you're using SQL Server 2000, you can use templates and then insert these templates into Query Analyzer. Write the following code in Query Analyzer, and from the File menu select Save As. Under Save as Type Select Template SQL files (*.sql):

```
CREATE TABLE #Holder(SerialNo INT,No INT, Head VARCHAR(300))
EXEC ExpandHierarchy <@start int>
SELECT * FROM #Holder
```

Now, whenever you want to call this stored procedure, Select Insert Template from the Edit menu, select the template, and press Enter. Now you can join your existing queries with the #Holder table.

So, in the end you have a template that you can distribute to the [developers](#). Let's test our template. Suppose you want to take the balances of a branch, say 5, and its children from the Journal table. [Listing 2](#) shows the code that does this. The code in [Listing 3](#) shows the revised ExpandHierarchy Stored Procedure. Figure 2, which you can open by clicking [Download the code](#) in the Article Information box, shows the flowchart for this stored procedure.

I have another solution to present. SQL Server 2000 has among its many great features user-defined functions and the table data type. So, in SQL Server 2000 your function might look like the code that [Listing 4](#) shows. On client side, you can simply write:

```
SELECT a.No,a.Head,a.SerialNo,
SELECT (SUM(b.Dr-b.Cr) FROM Journal b WHERE a.NO=b.Ac_No) AS sum
FROM funcExpandHierarchy(5) a
ORDER BY SerialNo
```

To test this solution, you can use the sample tables and data that [Listing 5](#) shows.

Reader Comments

[POST A COMMENT ►](#)

It's nice to know that SQL Server has a work around but I think the real solution here is to copy Oracle's 'CONNECT BY' syntax.

:)

Greg Mattson -December 28, 2001

POST YOUR COMMENTS HERE

If you don't have a username & password, please

[register now](#)

(Registration not required to post as Anonymous).

[Forgot your username/password?](#)

Username::

Password: :

Rate this article:

less useful		more useful		
1 ☐	2 ☐	3 ☐	4 ☐	5 ☐

Your Comments: :

[SUBMIT ►](#)

Windows IT Pro Marketplace

Improve IT Efficiency.

See who's saving time with Microsoft Windows Server System.

Remote Database Administration

Remote DBA Services , Expand DBA Resources, Reduce Costs

Windows Server System

Turn IT Capabilities into Business Results.

Featured Links

Want the new Reporting Services Poster?

Get the April issue of SQL Server Magazine and get the new R.S. poster

SQL Server Magazine Master CD - 25% off

Subscribe and get the full SQL Server Magazine article database on CD

Find Answers with the SQL Monthly Pass

Get monthly access to every article on www.sqlmag.com



SPONSORED LINK

[Conduct effective online meetings](#)

Try Microsoft® Office Live Meeting free for 14 days.
www.microsoft.com/liveishere

[what's this?](#)