SQL Server Database Corruption

SQL Server® PRO

Your Database Authority

APRIL 2014 / VOL. 16 / NO. 4

SQL Server 2014 New Features

Itzik Ben-Gan
T-SQL Interval Graphs, Part 1

SQL Server 2014's AMR Tool

Visual Studio Online FAQs

Code Documentation: A Necessary Evil?

Looking for continuous monitoring and real-time & historical analysis of SQL Server and OS health?

Give **SQL Diagnostic Manager** a spin.

How about a new way to discover, track, and manage your SQL Server environment effectively?

Get **SQL Elements**, hot off the presses!

What about a suite of tools that'll make your DBA skills superhuman?

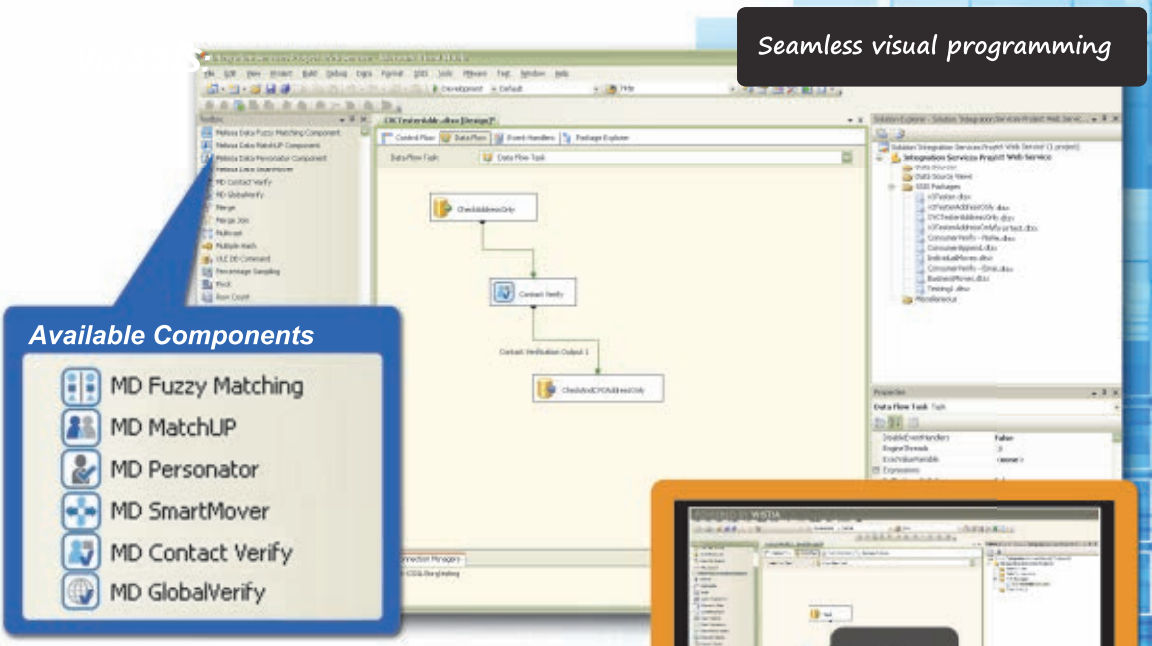
TechEd **Best of 2013** winner **SQL Toolbox** has it all.

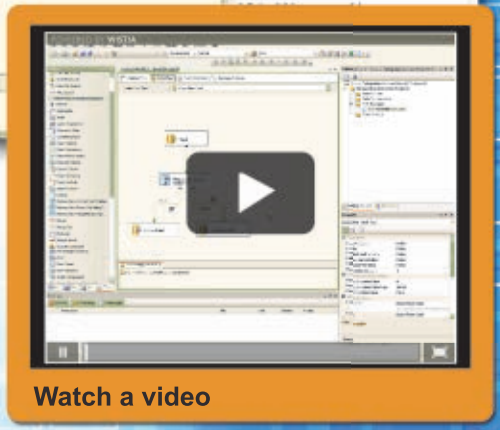
idera **A Solution for Every DBA**

Come check out our full line of SQL Server offerings and see what we can do to make your DBA life easier.

Data verification – anywhere, everywhere.

Global organizations rely on accurate, current and complete data in today's increasingly competitive marketplace. Melissa Data, a world-leading data quality vendor, offers solutions to easily verify addresses and geocode locations for **over 240 countries** – right in Microsoft SQL Server Integration Services (SSIS). We can scale our solutions to meet the needs of any global enterprise.





Powerful Data Cleaning Capabilities:

- Global address verification
- Global location geocoding
- Advanced fuzzy matching & record linkage
- Change of address processing

DOWNLOAD A FREE TRIAL NOW!
www.MelissaData.com/clean-ssis

1-800-MELISSA (635-4772)

MELISSA DATA®
Your Partner in Data Quality

APRIL 2014 / VOL. 16 / NO. 4

COVER STORY ▼

New Features in SQL Server 2014

8

— Michael Otey

SQL Server 2014 has several compelling new features, such as the In-Memory OLTP engine, that might justify an upgrade in your organization.

Features

Products

In Every Issue

Chat with Us

Facebook Twitter LinkedIn

IN-DEPTH SQL SERVER TRAINING — WITHOUT THE PITCH

IT/Dev CONNECTIONS

SEPTEMBER 15-19, 2014
ARIA RESORT • LAS VEGAS

Learn from the most experienced SQL Server professionals in the industry – all of our content is shaped by feedback from you and your peers.






SQL Server Topics covered:

Reporting Services | Integration Services | Performance Monitoring & Tuning | Big Data and SQL Server Database Development | Database Administration | In-Memory OLTP | AlwaysOn | Analysis Services Data visualization and Power BI tools | PowerPivot | SharePoint Integration | Business Intelligence Virtualization (Hyper-V, VMware) | Windows Azure SQL Database | And more

Why YOU need to be there

because you'll leave with tips and techniques you can implement immediately.

Who Attends

your peers and our industry experts, who are in the trenches facing similar challenges, eager and excited to share solutions.

What you'll learn

technical training, how-to information, next-level techniques that will escalate your SQL Server knowledge.

Register Now and Save $400 off the All-Access Pass! Rates go up May 31, 2014.

Go to www.devconnections.com/register

Documenting Code: A Necessary Evil, or Just Plain Wrong?

I'm a firm believer in comments—How about you?

Maybe it's the nature of the Internet, but some months are just full of odd pronouncements. In late January, it was, "The DBA is dead!" (See "The DBA Is Alive and Well—Here's Why.") The following week, some developers announced, "We're against documenting code!" As I understand it, the argument is that documentation itself isn't code, and although documentation can be well intended, it only prevents you from understanding what the code actually does. The correct approach to documenting code, according to this line of thinking, is that you should be writing Really Obvious Code (ROC). The only necessary comments are for your methods, telling other programmers what data and data types they should expect in each method's parameters and then what the function should return.

Michael Otey
is senior technical director for *Windows IT Pro* and *SQL Server Pro*.

Email

SQL Server PRO

EDITORIAL
Vice President, Content & User Engagement: Joe Panettieri
Editorial Director: Megan Keller
Editor-in-Chief, IT Group: Amy Eisenberg
Senior Technical Director: Michael Otey
Technical Director, IT Group: Sean Deuby
Editors: Jason Bovberg, Blair Greenwood, Caroline Marwitz
Website and Social Media Manager: Jayleen Heft
Managing Editor: Lavon Peters

CONTRIBUTING EDITORS
Itzik Ben-Gan, Michael K. Campbell, Kalen Delaney, Andrew J. Kelly, Brian Lawton, Douglas McDowell, Stacia Misner, Brian Moran, Michelle A. Poolet, Paul S. Randal, Kimberly L. Tripp, William Vaughn, Richard Waymire

ART & PRODUCTION
Senior Graphic Designer: Matt Wiebe
Group Production Manager: Julie Jantzer-Ward
Project Manager: Adriane Wineinger

ADVERTISING SALES
Strategic Accounts Director: Chrissy Ferraro • 970-203-2883
Account Executives: Megan Key • 970-203-2844
Barbara Ritter • 858-367-8058
Cass Schulz • 858-357-7649

CLIENT SERVICES
Senior Client Services Manager:
Michelle Andrews • 970-613-4964

MARKETING & CIRCULATION
Customer Service • 800-793-5697
Vice President, User Marketing & Marketing Analytics: Tricia Syed

TECHNOLOGY DIVISION & PENTON MARKETING SERVICES
Senior Vice President: Sanjay Mutha

CORPORATE
Chief Executive Officer: David Kieselstein
Chief Financial Officer/Executive Vice President: Nicola Allais

Penton

LIST RENTALS
Sarah Nowowiejski

REPRINTS
Reprint Sales: Wright's Media • 877-652-5295

Video

Michael Otey discusses the need for code documentation

As a former developer, I'm calling foul. There might be situations and certain types of code that can get away with such minimalistic types of comments, but for most applications—at least the types of applications that I have the questionable pleasure to work with—I don't think that's adequate. I believe that commenting your code is more than a nicety; it's a necessity.

You Need Them More Than You Think

No doubt, part of what makes a great developer is his or her ability to write ROC, but in the corporate environment there are usually a lot of hands in the pie. The sad truth is that not every one of those contributors is a great programmer—or even a good programmer. Business applications often have a long, sometimes decades-spanning lifecycle. And there are times when you'll be stuck working with code that uses poor or indecipherable variable names and questionable logic and flow. There are also times when you're not sure what a given piece of code is actually doing, let alone what it was intended to do. Often, the person who wrote that code has been gone for years and the only clues you have for figuring out the puzzle lie in the comments. When

that's the case, you're delighted with whatever comments you might happen to find in there.

I was a developer for a Fortune 100 company at my former job, and I distinctly remember a couple situations in which the comments saved a lot of time as I tried to decipher what was going on with the code I was working on. The most obvious one was a cost-estimating program for a corrugated-box manufacturer. Needless to say, cost estimating can be a fairly complex process, but the code worked as expected—except with one particular machine. Looking at the code revealed little at first; it looked OK to me. Fortunately, the comments described what the routine was *supposed* to do. Following through the code revealed a scoping problem in which a previous programmer had mistakenly created an almost identically named variable to the one the routine was working with. This variable was used for the local calculations, but sadly the code never assigned the contents of the working variable to the original variable. Sure, I could have found this problem without investigating the comments, but the comments gave me quick insight into some unfamiliar code, thereby speeding up the troubleshooting process. I'll even admit that there have been times when I've gone back to code that I've written—even code whose purpose I couldn't remember—and read my own comments to get an overview of what was going on.

True Believer

I'm a firm believer in commenting code: The more comments, the merrier. At a minimum, methods should begin with a description of their purpose as well as their input and output parameters. Long code blocks and loops should have a description that describes what the original developer intended. All code updates and modifications should have comments and dates that tell you the "who, what, and when" about the code changes. Just like the DBA, documentation is far from dead. As long as programmers still write code, there's a need for code comments. ■

New Features in SQL Server 2014

In-Memory OLTP engine and other new features might justify an upgrade

Michael Otey

is senior technical director for *Windows IT Pro* and *SQL Server Pro*.

Email

Microsoft's new release of SQL Server 2014 comes pretty close on the heels of the last SQL Server 2012 release. For many organizations, this could be a hurdle to adoption, because upgrading core pieces of an IT infrastructure can be both costly and resource-intensive. However, SQL Server 2014 has several compelling new features that can definitely justify an upgrade. Without a doubt, the most notable new feature is the new In-Memory OLTP engine, which promises some big performance improvements for OLTP applications. The past couple of SQL Server releases have had a strong focus on business intelligence (BI), which makes the new In-Memory OLTP engine an especially welcome addition for relational database professionals. Let's dig in and have a closer look at the new In-Memory OLTP engine and the other new features in SQL Server 2014.

New In-Memory OLTP Engine

By far the most important new feature in SQL Server 2014 is the In-Memory OLTP engine (formerly code-named Hekaton). By moving select tables and stored procedures into memory, you can drastically reduce I/O and improve performance of your OLTP applications.

Microsoft states that some applications can expect up to a 20x performance improvement. Edgenet, an early adopter, saw a 7x performance increase in its online and retail supply application.

The In-Memory OLTP engine is designed for high concurrency and uses a new optimistic concurrency control mechanism to eliminate locking delays. The In-Memory OLTP tables are copied into memory and made durable by transaction log writes to disk. An all-new lock-free engine processes the transactions for memory-resident tables. Stored procedure performance is improved by compiling the stored procedures into native code DLLs. Standard T-SQL stored procedures are interpreted, which adds overhead to the execution process. Compiling the stored procedures to native Win64 code makes them directly executable, thereby maximizing their performance and minimizing execution time.

To help you evaluate how the In-Memory OLTP engine will improve your database performance, Microsoft includes the new Analysis, Migrate, and Report (AMR) tool. Like its name suggests, the AMR tool analyzes your database and helps you identify the tables and stored procedures that would benefit from moving them into memory. It lists the expected performance improvements as well as any incompatibilities that need to be addressed. In addition, the AMR tool can help you perform the actual migration of tables to the new memory-optimized format. (For more information about the AMR tool, see "SQL Server 2014's Analysis, Migrate, and Report Tool.")

The In-Memory OLTP engine works with commodity server hardware, but it has a number of limitations. For instance, not all of the data types are supported. Some of the data types that aren't supported for memory-optimized tables include geography, hierarchyid, image, text, ntext, varchar(max), and xml. In addition, several database features can't be used with the new In-Memory OLTP capability. Database mirroring, snapshots, computed columns, triggers, clustered indexes, identity columns, FILESTREAM storage, and FOREIGN KEY, CHECK, and UNIQUE constraints aren't supported.

Database backups in SQL Server now support built-in database encryption.

The In-Memory OLTP engine is supported on Windows Server 2012 R2, Windows Server 2012, and Windows Server 2008 R2 SP2. In addition, you need to be using the SQL Server 2014 Enterprise, Developer, or Evaluation edition. Notably, In-Memory OLTP won't be supported on the SQL Server 2014 Standard edition. For more information, check out "Rev Up Application Performance with the In-Memory OLTP Engine."

Enhanced Windows Server 2012 Integration

SQL Server 2014 provides improved integration with Windows Server 2012 R2 and Windows Server 2012. SQL Server 2014 will have the ability to scale up to 640 logical processors and 4TB of memory in a physical environment. It can scale up to 64 virtual processors and 1TB of memory when running on a virtual machine (VM).

SQL Server 2014 also provides a new solid state disk (SSD) integration capability that enables you to use SSD storage to expand SQL Server 2014's buffer pool. The new buffer pool enhancements can help increase performance in systems that have maxed out their memory capability by using high-speed nonvolatile RAM (NVRAM) in the SSD drives as an extension to SQL Server 2014's standard buffer pool. The new buffer pool extensions can provide the best performance gains for read-heavy OLTP workloads.

SQL Server 2014's Resource Governor provides a new capability to manage application storage I/O utilization. First introduced with SQL Server 2008, the Resource Governor originally enabled you to limit the amount of CPU and memory that a given workload can consume. SQL Server 2014 extends the reach of the Resources Governor so that you can now manage storage I/O usage as well. The SQL Server 2014 Resource Governor can limit the physical I/Os issued for user threads in a given resource pool, allowing you to have more predictable application performance.

SQL Server 2014 also integrates with several new and improved features in Windows Server 2012 R2 and Windows Server 2012. For example, SQL Server 2014 supports the OSs' new Storage Spaces

feature. With Storage Spaces, you can create pools of tiered storage to improve application availability and performance. In addition, SQL Server 2014 can take advantage of the OSs' Server Message Block (SMB) 3.0 enhancements to achieve high-performance database storage on Windows Server 2012 R2 and Windows Server 2012 file shares. Many enhancements were made to SMB 3.0, with the most notable being SMB Transparent Failover and SMB Direct. The new SMB Transparent Failover feature provides highly reliable SMB storage that's fully supported for applications like SQL Server and Hyper-V. With the new SMB Direct feature, you can leverage the NIC's Remote Direct Memory Access (RDMA) feature to provide access speeds for SMB file shares nearing the access speed for local resources.

Enhancements to AlwaysOn Availability Groups

SQL Server 2014's AlwaysOn Availability Groups has been enhanced with support for additional secondary replicas and Windows Azure integration. First introduced with SQL Server 2012, AlwaysOn Availability Groups boosted SQL Server availability by providing the ability to protect multiple databases with up to four secondary replicas. In SQL Server 2014, Microsoft has enhanced AlwaysOn integration by expanding the maximum number of secondary replicas from four to eight. Readable secondary replicas are now available for read-only workloads, even when the primary replica is unavailable. SQL Server 2014 also provides Windows Azure AlwaysOn integration. This new integration feature enables you to create asynchronous availability group replicas in Windows Azure for disaster recovery. In the event of a local database outage, you can run your SQL Server databases from Windows Azure VMs. The new Windows Azure AlwaysOn availability options are fully integrated into SQL Server Management Studio (SSMS).

Enhancements to Backups

Database backups in SQL Server now support built-in database encryption. Previous releases all required a third-party product to

encrypt database backups. The backup encryption process uses either a certificate or an asymmetric key to encrypt the data. The supported backup encryption algorithms are Advanced Encryption Standard (AES) 128, AES 192, AES 256, and Triple DES (3DES).

SQL Server 2014 also provides new Windows Azure integration to SQL Server's native backup capabilities. You can specify a Windows Azure URL as the target for your SQL Server 2014 database backups. This new Windows Azure backup feature is fully integrated into SSMS.

Updateable Columnstore Indexes

Columnstore indexes are another of Microsoft's high performance in-memory technologies. Microsoft introduced the columnstore index in SQL Server 2012 to provide significantly improved performance for data warehousing types of queries. Microsoft states that for some types of queries, columnstore indexes can provide up to 10x performance improvements. However, in the original implementation of the columnstore indexes, the underlying table had to be read-only. SQL Server 2014 eliminates this restriction. The new updateable columnstore index enables updates to be performed to the underlying table without first needing to drop the columnstore index. A SQL Server 2014 columnstore index must use all of the columns in the table, and it can't be combined with other indexes.

SQL Server Data Tools for Business Intelligence

Previously known as Business Intelligence Development Studio (BIDS) and SQL Server Data Tools (SSDT), the new SQL Server Data Tools for BI (SSDT BI) is used to create SQL Server Analysis Services (SSAS) models, SQL Server Reporting Services (SSRS) reports, and SQL Server Integration Services (SSIS) packages. SSDT BI is based on Microsoft Visual Studio 2012 and supports SSAS and SSRS for SQL Server 2014 and earlier, but SSIS projects are limited to SQL Server 2014. In the pre-release version of SQL Server 2014 Community

Technology Preview 2 (CTP2), SQL Server Setup doesn't install SSDT BI. Instead, you must download it separately from the Microsoft Download Center.

Power BI for Office 365 Integration

Power BI for Office 365 is Microsoft's cloud-based BI solution that leverages familiar Office 365 and Excel tools. Power BI for Office 365 provides business insights through data visualization and navigation capabilities. Power BI for Office 365 includes:

- Power Pivot (formerly PowerPivot). This Excel 2010 and Excel 2013 ProPlus add-in enables Excel to perform data analysis on large volumes of data.
- Power View. This Excel 2013 ProPlus add-in provides a Silverlight-based data visualization and navigation tool. Microsoft has extended Power View so that you can now use it with multidimensional models (OLAP cubes). Power View multidimensional models also support queries using Data Analysis Expressions (DAX). Power View's data visualization capabilities have also been enhanced. Power View now supports a number of data visualizations, including tables, matrixes, bubble charts, and geographical maps. To learn more about Power View's new multidimensional support, go to MSDN's Power View for Multidimensional Models web page.
- Power Query (formerly code-named Data Explorer). This Excel 2013 add-in lets you discover and integrate data into Excel. It supports SQL Server data sources as well as external sources such as Windows Azure, text files, XML files, Open Data Protocol (OData) feeds, web pages, Hadoop data sets, and public data sets (e.g., Open Government data from data.gov).
- Power Map (formerly code-named GeoFlow). This Excel 2013 ProPlus add-in provides 3D mapping visualizations.

As you can see, several of the Power BI for Office 365 components require Excel 2013, which must be acquired separately. You can learn

more about Microsoft's Power BI for Office 365 tools at Microsoft's Power BI for Office 365 website.

Unchanged Subsystems and Discontinued Features

Like you might expect after such a short release cycle, not every subsystem has been updated. There are no major changes to SQL Server 2014's replication, SSIS, or SSRS.

There are several older and outdated capabilities that Microsoft is dropping from the SQL Server 2014 release. To see a list of the features and functionality that Microsoft is dropping from SQL Server 2014, check out MSDN's Deprecated Database Engine Features in SQL Server 2014 and Discontinued Database Engine Functionality in SQL Server 2014 web pages.

Time to Upgrade?

Upgrading to a new release is usually a difficult decision for most organizations. However, SQL Server 2014's new In-Memory OLTP engine, with its promise of significantly improved application performance, offers a very compelling reason to upgrade for customers using SQL Server to support OLTP applications. One great way to find out the type of performance improvement that you might get out of SQL Server 2014's In-Memory OLTP capability is to download and install the SQL Server 2014 Evolution Edition and use the AMR tool to analyze your production workload. The AMR tool supports collecting data on SQL Server 2008 and later instances. This will give you a good idea of the type of performance improvements that you might expect to get using the new In-Memory OLTP engine as well as the changes that you might need to make to implement it. SQL Server 2014's In-Memory OLTP support promises to boost your database application performance to the next level. ■

T-SQL Interval Graphs Challenge, Part 1

Cursor-based and set-based solutions to the interval packing problem

I recently got a challenge from my friend John Paul Cook, who is a registered nurse and a fellow SQL Server MVP. The challenge involves drug prescriptions modeled as interval graphs. You can find the full details of the challenge in the paper "Counting Drug Exposure in SAS® with Interval Graph Modeling" by Xiaoqiang Wang. In this paper, Wang presents a problem and describes how to solve it with SAS. John's challenge to me was to use T-SQL to find an efficient solution to the problem.

I get to work on a lot of T-SQL puzzles. I love the process, which often involves many agonizing hours of trying to come up with efficient, elegant solutions. And in some of the cases, when you're really fortunate, you also manage to discover beautiful new patterns that can be used in solving other challenges. Such was the case with this challenge.

I'd like to thank not only John Paul Cook, but also Alejandro Mesa, Paul White, Adam Machanic, Peter Larsson, and Geri Reshef for their insights.

The Challenge

As I mentioned, the challenge involves drug prescriptions modeled as interval graphs. Run the code in Listing 1 to create the Patients and Prescriptions tables and fill them with a small set of sample data. The code in Listing 1 also creates indexes to support the solutions that I'll describe.

Itzik Ben-Gan

is a cofounder of SolidQ. He's a SQL Server MVP and is the author of several books about T-SQL, including *Microsoft SQL Server 2012 High-Performance T-SQL Using Window Functions* (Microsoft Press).

Email

Twitter

Website

Blog

Download

Download the code

Listing 1: DDL to Create the Prescriptions and Patients Tables with a Small Set of Sample Data

```
SET NOCOUNT ON;
USE tempdb;

IF OBJECT_ID(N'dbo.Prescriptions', N'U') IS NOT NULL
  DROP TABLE dbo.Prescriptions;
IF OBJECT_ID(N'dbo.Patients', N'U') IS NOT NULL
  DROP TABLE dbo.Patients;

CREATE TABLE dbo.Patients
(
  patientid INT NOT NULL,
  CONSTRAINT PK_Patients PRIMARY KEY(patientid)
);

CREATE TABLE dbo.Prescriptions
(
  prescriptionid INT  NOT NULL IDENTITY,
  patientid      INT  NOT NULL,
  drugid         INT  NOT NULL,
  startdate      DATE NOT NULL,
  numdays        INT  NOT NULL,
  enddate AS DATEADD(day, numdays, startdate),
  CONSTRAINT CHK_Prescriptions_ed_sd CHECK(numdays > Ø)
);

CREATE UNIQUE CLUSTERED INDEX idx_start
  ON dbo.Prescriptions
   (patientid, drugid, startdate, prescriptionid);

ALTER TABLE dbo.Prescriptions
  ADD CONSTRAINT PK_Prescriptions PRIMARY KEY
    NONCLUSTERED(prescriptionid);

-- code to fill tables with small set of sample data
```

Listing 1: *continued*

```
TRUNCATE TABLE dbo.Prescriptions;
TRUNCATE TABLE dbo.Patients;

INSERT INTO dbo.Patients(patientid) VALUES(1);

INSERT INTO dbo.Prescriptions
  (patientid, drugid, startdate, numdays) VALUES
  (1,  1, '20140101', 3),
  (1,  2, '20140101', 5),
  (1,  3, '20140102', 4),
  (1,  4, '20140102', 5),
  (1,  4, '20140103', 2),
  (1,  2, '20140105', 5),
  (1,  3, '20140106', 4),
  (1,  1, '20140107', 3),
  (1,  4, '20140108', 1),
  (1,  4, '20140120', 4),
  (1,  4, '20140122', 1),
  (1,  5, '20140212', 3),
  (1,  5, '20140215', 3),
  (1,  5, '20140220', 1),
  (1,  5, '20140223', 1),
  (1,  5, '20140226', 1);
```

You can use the small set of sample data to check the validity of your solutions against desired results that I'll provide. You'll need a larger set of sample data to test the performance of your solutions. The code in Listing 2 creates a helper table called GetNums that returns a sequence of integers in a requested range, then uses this function to populate the tables with a larger set of sample data.

The Prescriptions table contains information about drug prescriptions for patients. The table contains the following columns: prescriptionid (surrogate key), patientid, drugid, startdate (start of prescription period), and numdays (number of days the prescription covers). The

Listing 2: Code to Create the Helper Function GetNums and a Large Set of Sample Data

```
IF OBJECT_ID(N'dbo.GetNums', N'IF') IS NOT NULL
  DROP FUNCTION dbo.GetNums;
GO

CREATE FUNCTION dbo.GetNums(@low AS BIGINT, @high AS BIGINT) RETURNS TABLE
AS
RETURN
  WITH
    L0   AS (SELECT c FROM (SELECT 1 UNION ALL SELECT 1)
                AS D(c)),
    L1   AS (SELECT 1 AS c FROM L0 AS A CROSS JOIN L0 AS B),
    L2   AS (SELECT 1 AS c FROM L1 AS A CROSS JOIN L1 AS B),
    L3   AS (SELECT 1 AS c FROM L2 AS A CROSS JOIN L2 AS B),
    L4   AS (SELECT 1 AS c FROM L3 AS A CROSS JOIN L3 AS B),
    L5   AS (SELECT 1 AS c FROM L4 AS A CROSS JOIN L4 AS B),
    Nums AS (SELECT ROW_NUMBER() OVER(ORDER BY (SELECT NULL))
                AS rownum
              FROM L5)
  SELECT TOP(@high - @low + 1) @low + rownum - 1 AS n
  FROM Nums
  ORDER BY rownum;
GO

DECLARE
  @numpatients      AS INT  = 10000,
  @numdrugs         AS INT  = 10,
  @numprescriptions AS INT  = 10,
  @firststartdate   AS DATE = '20140101',
  @laststartdate    AS DATE = '20141231',
  @maxnumdays       AS INT  = 30;

TRUNCATE TABLE dbo.Prescriptions;
TRUNCATE TABLE dbo.Patients;

INSERT INTO dbo.Patients(patientid)
```

Listing 2: *continued*

```
  SELECT PT.n AS patientid
  FROM dbo.GetNums(1, @numpatients) AS PT;

INSERT INTO dbo.Prescriptions
  (patientid, drugid, startdate, numdays)
  SELECT
    PT.n AS patientid,
    D.n AS drugid,
    DATEADD(day,
      ABS(CHECKSUM(NEWID()))
        % DATEDIFF(day, @firststartdate, @laststartdate),
      @firststartdate ) AS startdate,
    ABS(CHECKSUM(NEWID())) % @maxnumdays + 1 AS numdays
  FROM dbo.GetNums(1, @numpatients) AS PT
    CROSS JOIN dbo.GetNums(1, @numdrugs) AS D
    CROSS JOIN dbo.GetNums(1, @numprescriptions) AS PR;
```

table also defines a computed column called enddate that computes the date on which the prescription runs out (the day following the last consumption day). For example, a 3-day prescription starting on January 1, 2014 (with a number of days of 3) will get an end date of January 4, 2014. The patient consumes the drug on January 1, January 2, and January 3, and then doesn't have a dose for January 4.

The challenge described in "Counting Drug Exposure in SAS® with Interval Graph Modeling" involves two parts:

1. Packing intersecting prescription periods
2. Identifying periods in which the patient was exposed to a certain minimum number of drugs

When John sent me the challenge, he said (and I quote), "Researchers are studying drug prescriptions to look for potential adverse outcomes. More bluntly, and I'm really not being overly dramatic, the objective is to keep patients from being killed."

If that's not a motivator to find an efficient solution, I don't know what is!

The objective is to keep patients from being killed.

What's also interesting is that when I started working on this challenge, I thought the packing problem would be the easy part. Little did I know that this part would turn out to be a formidable challenge that would require most of the effort. In this article, I describe the packing problem and provide both a cursor-based solution and a set-based solution. I'll cover the counting problem in my next article.

The Packing Problem

The packing problem involves packing intersecting prescription periods for the same patient and drug. But there's a twist: A patient doesn't consume more than one dose of a drug on the same day even if there are multiple concurrent prescription periods. Instead, the prescription that started later (or, in case of a tie, the prescription with the greater surrogate key value) is pushed forward in time to the date when the prescription that started earlier expires. This is probably easier to comprehend visually.

Figure 1 graphically depicts the input drug prescription periods based on the small set of sample data, as well as arrows representing the packed periods (again, with the end date representing the day following the last consumption day). Observe the first three prescription periods for Patient 1, Drug 4:

1—startdate: 1/2; numdays: 5
2—startdate: 1/3; numdays: 2
3—startdate: 1/8; numdays: 1

The first prescription starts on January 2 and lasts for 5 days, so it expires on January 7. The second prescription starts before January 7, so it's pushed forward in time to that date. Because the second prescription lasts for 2 days, it expires on January 9. The third prescription starts before January 9, so it's pushed forward to that date.

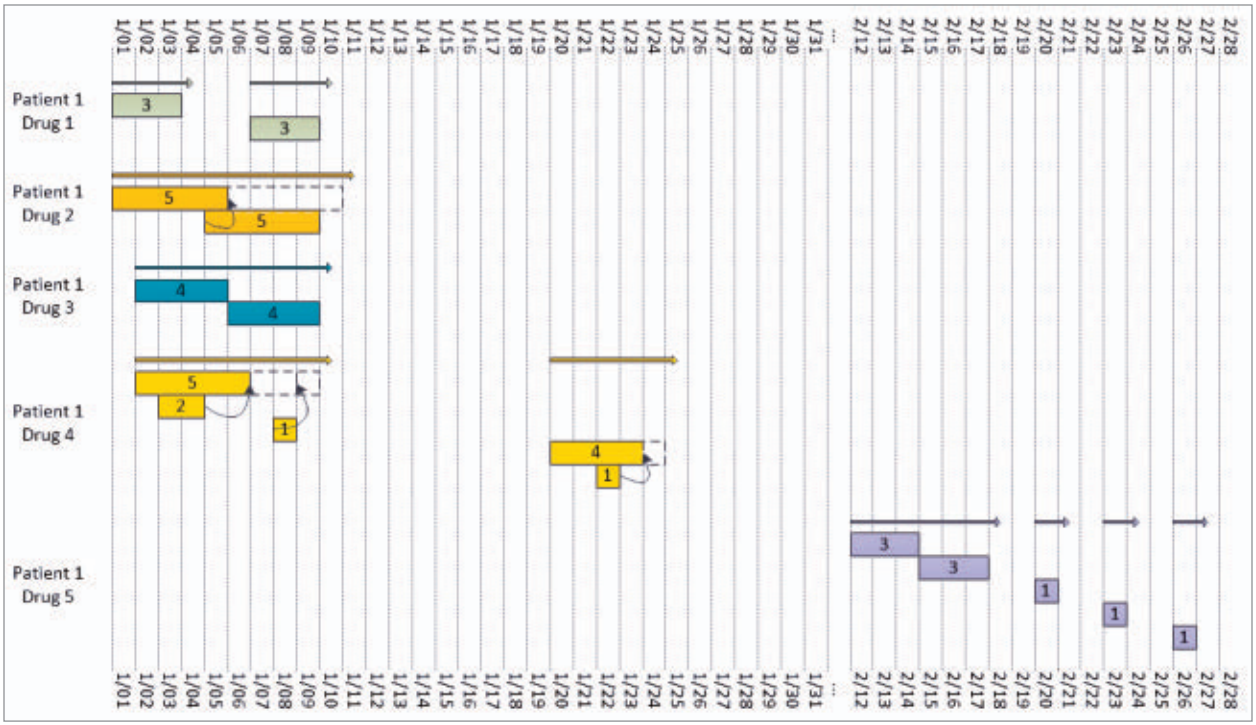


Figure 1
Input and Packed Intervals

Because the third prescription lasts for 1 day, it expires on January 10. The packed prescription period is represented by the arrow above the individual prescription periods; it starts on January 2 and expires on January 10.

Your task is to write a solution that returns the packed prescription periods for each patient and drug, based on this logic. Figure 2 shows the desired results based on the small set of sample data.

I cover both a cursor-based solution and a set-based one. I provide the performance measures I obtained when running the solutions against the large set of sample data, with results discarded in SQL Server Management Studio (SSMS). To discard results after execution, right-click an empty area in the query pane, select Query Options, and select the *Discard results after execution and confirm* option under Results-Grid. This action will suppress printing the result rows in the grid output.

Figure 2: Packed Prescriptions

patientid	drugid	startdate	enddate
1	1	2014-01-01	2014-01-04
1	1	2014-01-07	2014-01-10
1	2	2014-01-01	2014-01-11
1	3	2014-01-02	2014-01-10
1	4	2014-01-02	2014-01-10
1	4	2014-01-20	2014-01-25
1	5	2014-02-12	2014-02-18
1	5	2014-02-20	2014-02-21
1	5	2014-02-23	2014-02-24
1	5	2014-02-26	2014-02-27

Cursor-Based Solution

The cursor-based solution is quite straightforward, involving a single pass over the data. Listing 3 contains the complete solution code. The code declares a table variable called @Result where the packed intervals are stored.

Listing 3: Cursor-Based Solution for Packing Prescriptions

```
SET NOCOUNT ON;

DECLARE @Result AS TABLE
(
  patientid INT  NOT NULL,
  drugid    INT  NOT NULL,
  startdate DATE NOT NULL,
  enddate   DATE NOT NULL
);

DECLARE
  @patientid      AS INT,
  @drugid         AS INT,
```

Listing 3: *continued*

```
  @startdate      AS DATE,
  @numdays        AS INT,
  @sumnumdays     AS INT,
  @prevpatientid  AS INT,
  @prevdrugid     AS INT,
  @firststartdate AS DATE;

DECLARE C CURSOR FAST_FORWARD FOR
  SELECT patientid, drugid, startdate, numdays
  FROM dbo.Prescriptions
  ORDER BY patientid, drugid, startdate, prescriptionid;

OPEN C;

FETCH NEXT FROM C INTO
  @patientid, @drugid, @startdate, @numdays;

SELECT
  @prevpatientid  = @patientid,
  @prevdrugid     = @drugid,
  @firststartdate = @startdate,
  @sumnumdays     = 0;

WHILE @@fetch_status = 0
BEGIN
  IF    @prevpatientid <> @patientid
     OR @prevdrugid    <> @drugid
     OR DATEADD(day, @sumnumdays, @firststartdate)
          < @startdate
  BEGIN
    INSERT INTO @Result(patientid, drugid, startdate, enddate)
      VALUES(@prevpatientid, @prevdrugid, @firststartdate,
        DATEADD(day, @sumnumdays, @firststartdate));

    SELECT
```

Listing 3: *continued*

```
      @firststartdate = @startdate,
      @sumnumdays     = 0;
  END

  SELECT
    @sumnumdays    += @numdays,
    @prevpatientid  = @patientid,
    @prevdrugid     = @drugid;

  FETCH NEXT FROM C INTO
    @patientid, @drugid, @startdate, @numdays;
END

IF @sumnumdays > 0
  INSERT INTO @Result(patientid, drugid, startdate, enddate)
    VALUES(@prevpatientid, @prevdrugid, @firststartdate,
      DATEADD(day, @sumnumdays, @firststartdate));

CLOSE C;

DEALLOCATE C;

SELECT * FROM @Result;
```

The rows from the Prescriptions table are fed to the cursor, sorted first by patientid and drugid (because the calculation is performed for each patient and drug separately), then by startdate and prescriptionid (as a tiebreaker). The query that feeds the data to the cursor is optimized with an ordered scan of the index idx_start, so no sort operation is required.

The code loops through the cursor records one at a time, fetching the current record's values into local variables. The @firststartdate variable holds the start date of the most recent packed interval, and the @sumnumdays variable holds the running total number of

prescription days for the most recent packed interval. For each record fetched, the code checks the following condition to know whether to close the current packed interval and start a new one:

```
IF    @prevpatientid <> @patientid
   OR @prevdrugid    <> @drugid
   OR DATEADD(day, @sumnumdays, @firststartdate) < @startdate
```

The current packed interval closes when any of the following conditions is met:

- Any of the group elements (patientid or drugid) changes.
- The start date of the current packed interval (@firststartdate) plus the running total number of prescription days since it started (@sumnumdays) is before the start date of the new prescription interval.

In case of a match, the code closes the new packed interval, writes its info to the table variable, then opens a new packed interval initializing @firststartdate with @startdate and @sumnumdays with 0. If the current packed interval isn't closed (i.e., the new interval should extend it), the code increments @sumnumdays by @numdays.

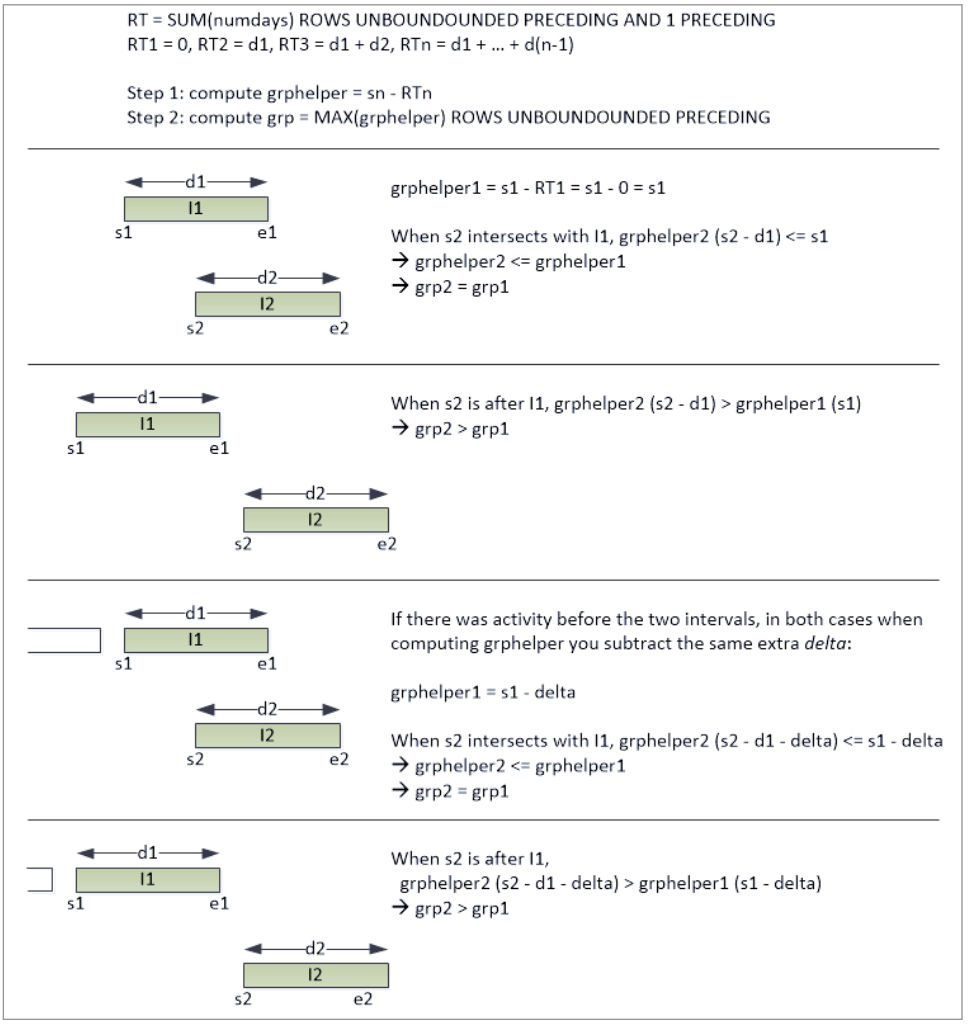
When the loop is done, the code inserts the last packed interval info into the table variable (if relevant), then closes and deallocates the cursor. The cursor-based solution completed in 31 seconds on my system.

Set-Based Solution

The set-based solution involves computing a group identifier (call it grp) that associates each prescription interval with the packed interval that it belongs to. This is done in two steps, which are illustrated in Figure 3, along with examples that explain why grp is unique per packed interval.

The first step calculates a value called grphelper. This value is computed as the current startdate value minus the running total sum of

Figure 3
Computing grphelper and grp



all numdays values of the intervals so far, exclusive of the current interval. The following query implements this calculation:

```
SELECT prescriptionid, patientid, drugid, startdate, numdays,
  DATEADD(day,
    - SUM(numdays) OVER(PARTITION BY patientid, drugid
                        ORDER BY startdate, prescriptionid
                        ROWS UNBOUNDED PRECEDING) + numdays,
```

```
    startdate) AS grphelper
FROM dbo.Prescriptions;
```

Figure 4 shows the output of this query, for the small set of sample data.

The second step is calculating grp as the maximum grphelper value so far. The following query adds this computation on top of the previous query represented by the CTE C1:

```
WITH C1 AS
(
  SELECT prescriptionid, patientid, drugid, startdate, numdays,
    DATEADD(day,
      - SUM(numdays) OVER(PARTITION BY patientid, drugid
                          ORDER BY startdate, prescriptionid
```

Figure 4: Calculating grphelper

prescriptionid	patientid	drugid	startdate	numdays	grphelper
1	1	1	2014-01-01	3	2014-01-01
8	1	1	2014-01-07	3	2014-01-04
2	1	2	2014-01-01	5	2014-01-01
6	1	2	2014-01-05	5	2013-12-31
3	1	3	2014-01-02	4	2014-01-02
7	1	3	2014-01-06	4	2014-01-02
4	1	4	2014-01-02	5	2014-01-02
5	1	4	2014-01-03	2	2013-12-29
9	1	4	2014-01-08	1	2014-01-01
10	1	4	2014-01-20	4	2014-01-12
11	1	4	2014-01-22	1	2014-01-10
12	1	5	2014-02-12	3	2014-02-12
13	1	5	2014-02-15	3	2014-02-12
14	1	5	2014-02-20	1	2014-02-14
15	1	5	2014-02-23	1	2014-02-16
16	1	5	2014-02-26	1	2014-02-18

```
                                    ROWS UNBOUNDED PRECEDING) + numdays,
        startdate) AS grphelper
  FROM dbo.Prescriptions
)
SELECT *,
  MAX(grphelper) OVER(PARTITION BY patientid, drugid
                      ORDER BY startdate, prescriptionid
                      ROWS UNBOUNDED PRECEDING) AS grp
FROM C1;
```

Figure 5 shows the output of this query. It contains both the grphelper and grp values.

If you examine Figure 3, you'll see four examples, with accompanying explanations leading to the conclusion that grp is unique per

Figure 5: Calculating grp

prescriptionid	patientid	drugid	startdate	numdays	grphelper	grp
1	1	1	2014-01-01	3	2014-01-01	2014-01-01
8	1	1	2014-01-07	3	2014-01-04	2014-01-04
2	1	2	2014-01-01	5	2014-01-01	2014-01-01
6	1	2	2014-01-05	5	2013-12-31	2014-01-01
3	1	3	2014-01-02	4	2014-01-02	2014-01-02
7	1	3	2014-01-06	4	2014-01-02	2014-01-02
4	1	4	2014-01-02	5	2014-01-02	2014-01-02
5	1	4	2014-01-03	2	2013-12-29	2014-01-02
9	1	4	2014-01-08	1	2014-01-01	2014-01-02
10	1	4	2014-01-20	4	2014-01-12	2014-01-12
11	1	4	2014-01-22	1	2014-01-10	2014-01-12
12	1	5	2014-02-12	3	2014-02-12	2014-02-12
13	1	5	2014-02-15	3	2014-02-12	2014-02-12
14	1	5	2014-02-20	1	2014-02-14	2014-02-14
15	1	5	2014-02-23	1	2014-02-16	2014-02-16
16	1	5	2014-02-26	1	2014-02-18	2014-02-18

packed interval. Notice in the figure that RT represents the running total sum of numdays so far, exclusive of the current interval.

The first two examples deal with the first two intervals for a given patient and drug, with the interval I1 representing the very first interval, and I2 representing the second interval (started at the same time or after I1 started). The last two examples are similar, but with existence of prescriptions before I1 and I2.

The first example assumes that I2 intersects I1. Here, grphelper1 = s1 - RT1 = s1 - 0 = s1.

Because in this example I2 intersects I1, grphelper2 (s2 - d1) < = s1. Therefore, grphelper2 < = grphelper1. And therefore, grp2 = grp1. In other words, when the two intervals intersect, both get the same grp value and therefore belong to the same packed interval.

The second example assumes that I2 starts after I1 expires. Here, grphelper2 (s2 - d1) > grphelper1 (s1). Therefore, grp2 > grp1. In other words, the two intervals will get different grp values and therefore belong to different packed intervals.

The third and fourth examples are similar to the first and second examples, respectively; however, I1 and I2 aren't the first two prescription intervals. What's different in these examples is that the RT values for both I1 and I2 include an extra delta that's equal to the running total of numdays values for the intervals that started before the l1 and l2 intervals.

The third example is when I2 intersects with I1, but there were intervals before I1. You need to include the delta in your calculation—but since in the calculations you subtract the same delta from both sides, it doesn't change the result. In this example, I2 intersects I1, so grphelper2 (s2 - d1 - delta) < = s1 - delta. Therefore, grphelper2 < = grphelper1. And therefore, grp2 = grp1. So, when I2 intersects I1, even if there were previous prescriptions, the grp values of both intervals are still the same, and they therefore belong to the same packed interval.

The fourth example is when I2 is after I1 but with previous existing prescriptions. Here, grphelper2 (s2 - d1 - delta) > grphelper1

(s1 - delta). Therefore, grp2 > grp1. Again, the two intervals get different grp values and therefore belong to different packed intervals.

Now that you have a grp value that uniquely identifies packed intervals within the same patient and drug group, what's left is to group the rows by patientid, drugid, and grp, and compute the start and end of each packed interval. Listing 4 contains the complete solution query. Figure 6 shows the query plan.

What's interesting to observe about the query plan is that the index idx_start is scanned only once in order, and both window function

Listing 4: Set-Based Solution for Packing Prescriptions

```
WITH C1 AS
(
  SELECT prescriptionid, patientid, drugid, startdate, numdays,
    DATEADD(day,
      - SUM(numdays) OVER(PARTITION BY patientid, drugid
                          ORDER BY startdate, prescriptionid
                          ROWS UNBOUNDED PRECEDING)
      + numdays,
      startdate) AS grphelper
  FROM dbo.Prescriptions
),
C2 AS
(
  SELECT *,
    MAX(grphelper) OVER(PARTITION BY patientid, drugid
                        ORDER BY startdate, prescriptionid
                        ROWS UNBOUNDED PRECEDING) AS grp
  FROM C1
)
SELECT
  patientid, drugid,
  MIN(startdate) AS startdate,
  DATEADD(day, SUM(numdays), MIN(startdate)) AS enddate
FROM C2
GROUP BY patientid, drugid, grp;
```

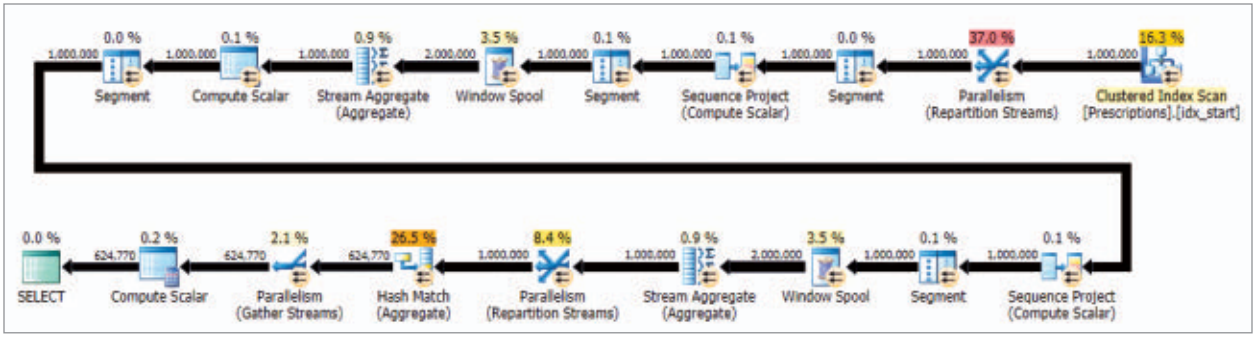


Figure 6
Plan for Query in Listing 4

computations rely on this order. None of them require an explicit sort operation. This makes the solution efficient with linear scaling. This solution completed in 4 seconds on my system.

Still, there's potential for improvement in how the optimizer handles the window aggregate functions. At the moment, window aggregate functions with a frame option are optimized with a pair of operators called Window Spool and Stream Aggregate. The Window Spool operator represents the frame of rows, and the Stream Aggregate operator aggregates those rows. When the frame starts with UNBOUNDED PRECEDING, the optimizer identifies the case as a "fast track" case, and instead of writing all applicable rows to the frame and then aggregating them, it takes the previous row's accumulation and adds the current row's value to compute the new running total.

The problem is that SQL Server writes one row to the spool with the aggregate, and another with the detail (remember that window functions aren't supposed to hide the detail like group functions do). There's great potential here to improve the optimization of the special (and very common) case of the frame ROWS UNBOUNDED PRECEDING. This case could theoretically be optimized like the ROW_NUMBER function is: namely, with a Sequence Project operator that computes the result values on the fly, as opposed to the extra work of writing the rows to a Window Spool (a work table) and aggregating them. Such optimization should easily allow a much faster query, probably around 1 second of run time instead of the current 4 seconds. I hope that Microsoft will

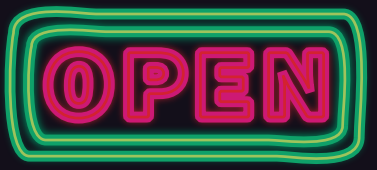
Learning Path

"Interval Queries in SQL Server"
"Intervals and Counts, Part 1"
"Intervals and Counts, Part 2"
"Intervals and Counts, Part 3"
"Intervals and Counts, Part 4"
"TSQL Challenge: Packing Date and Time Intervals"
"Clarifications Regarding TSQL Challenge: Packing Date and Time Intervals"
"Solutions to Packing Date and Time Intervals Puzzle"

further enhance the optimization of window functions, to improve the performance of set-based solutions.

What's Next?

In this article I covered the first part of the drug prescriptions puzzle—packing intersecting prescription periods. I presented a slow cursor-based solution that took 31 seconds to run. I then presented a much faster set-based solution that uses window aggregate functions to compute a group identifier for the packed intervals. In my next article, I'll cover the second part of the puzzle—identifying prescription periods during which the same patient was exposed to more than a certain minimum number of drugs. ■



SQL Server Pro Store

eLearning Classes

eBooks

On-Demand Training

Plus you can RENEW your subscription while you're there!

Stop by the store today!

SQL Server PRO

SQL Server 2014's Analysis, Migrate, and Report Tool

Determine which tables and stored procedures would benefit from In-Memory OLTP

With SQL Server 2014's new In-Memory OLTP engine, you can load tables and stored procedures in memory, which provides fast response times. The goal isn't to load all database tables and stored procedures in memory—just tables that are crucial to performance and stored procedures that have complex logical calculations.

To help identify which tables and stored procedures will provide the best performance gain after migration to In-Memory OLTP, SQL Server 2014 provides the new Analysis, Migrate, and Report (AMR) tool. Built into SQL Server Management Studio (SSMS), the tool consists of the:

- Transaction performance collector (which collects data about existing tables and stored procedures in order to analyze workloads) and transaction performance analysis reports (which gives recommendations about the tables and stored procedures to migrate to In-Memory OLTP based on the collected data)
- Memory Optimization Advisor (which guides you through the process of migrating a table to a memory-optimized table)
- Native Compilation Advisor (which helps you identify T-SQL elements that need to be changed before migrating a stored procedure to a natively compiled stored procedure)

The AMR tool leverages the new Transaction Performance Collection Sets for gathering information about workloads and the Management Data Warehouse (a relational database) to store the collected data. The Transaction Performance Collection Sets includes the:

Stéphane Savorgnano

is a consultant at dbi Services in Switzerland. He's an MCSA for SQL Server 2012 and an MCITP for SQL Server 2008. He has more than 15 years of experience in Microsoft software development and in SQL Server database solutions. He holds a master's degree in Informatics from Mulhouse University.

- Stored Procedure Usage Analysis collection set (which captures information about stored procedures for a future migration to natively compiled stored procedures)
- Table Usage Analysis collection set (which captures information about disk-based tables for a future migration to memory optimized tables)

Before you can use the AMR tool, you need to configure the Management Data Warehouse and the data collection process. After showing you how to do so, I'll demonstrate how to run the transaction performance analysis reports and how to use the two advisors.

Configuring the Management Data Warehouse

To configure the Management Data Warehouse, go to Object Explorer in SSMS. Expand the Management folder, right-click Data Collection, select Tasks, and click Configure Management Data Warehouse. This will launch the Configure Management Data Warehouse Wizard.

After the Welcome page, you'll find yourself on the Select Configuration Task page. On this page, select the option to configure a Management Data Warehouse.

On the Configure Management Data Warehouse Storage page, you need to specify the name of database that will host the Management Data Warehouse and the name of the server on which that database resides. If you need to create the database, click the New button.

On the *Map Logins and Users* page, you'll find the existing logins allowed for the server that will host the Management Data Warehouse. If needed, you can edit the logins or map users to the administrator, reader, and writer roles for the Management Data Warehouse.

On the *Complete the Wizard* page, you need to verify the Management Data Warehouse configuration. If it's OK, click Finish. When the configuration of the Management Data Warehouse has successfully completed, you should see a page like that in Figure 1. The Management Data Warehouse setup is now finished.

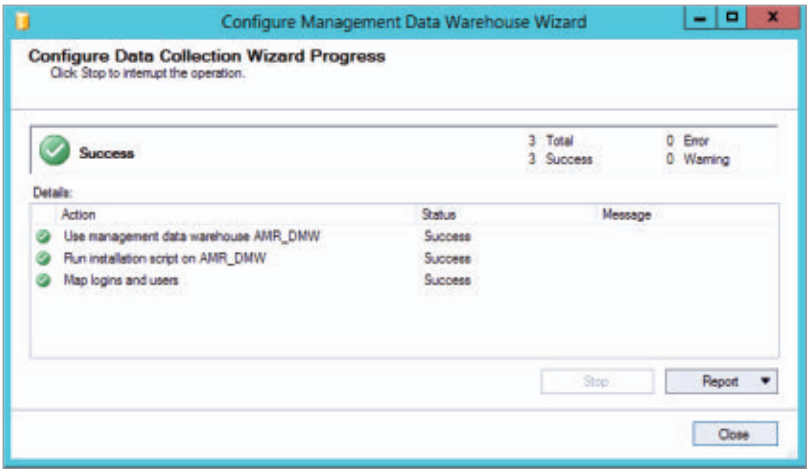


Figure 1
Verifying Configuration of the Management Data Warehouse

Configuring the Data Collection Process

To configure the data collection process, go to Object Explorer in SSMS. Expand the Management folder, right-click Data Collection, select Tasks, and click Configure Data Collection. This will launch the Configure Data Collection Wizard. After the Welcome page, you'll find the Setup Data Collection Sets page shown in Figure 2. Besides

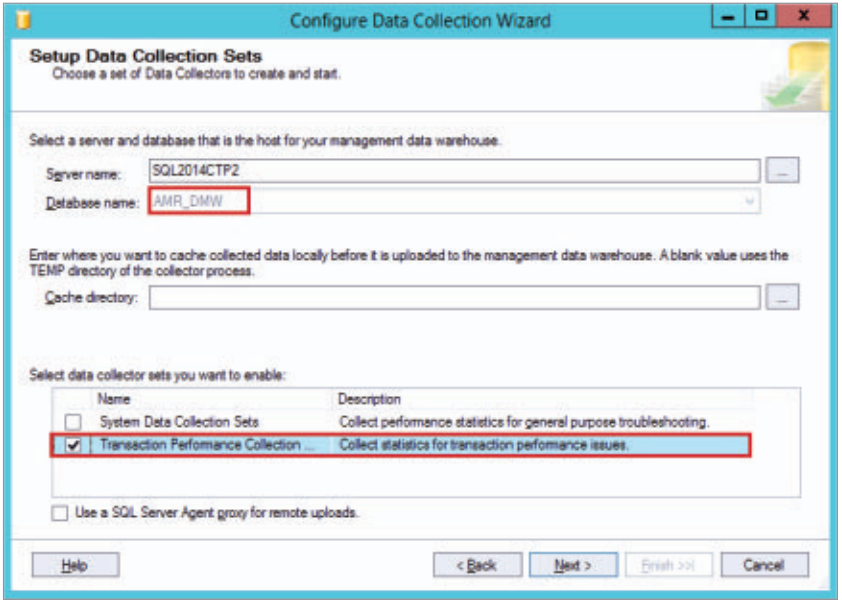


Figure 2
Specifying the Data Collector Sets

needing to specify the server and database that will host the Management Data Warehouse, you need to specify the data collector sets. In the list of collection sets, select the Transaction Performance Collection Sets check box so that the data collector will collect statistics for transaction performance issues.

If the Management Data Warehouse is located on a different SQL Server instance from the data collector and if SQL Server Agent isn't running under a domain account that has dc_admin permissions on the remote instance, you have to use a SQL Server Agent proxy. If that's the case, be sure to select the *Use a SQL Server Agent proxy for remote uploads* check box.

Once you're done configuring the Setup Data Collection Sets page, click Finish. When the wizard completes the configuration, you'll have an enabled data collection process that will collect information about all user databases. Note that SQL Server Agent must be started on the instance that will collect the data.

In the SQL Server Agent's Jobs folder, you'll see the jobs used to collect data from your workloads and the jobs used to upload the collected data into the Management Data Warehouse. The data collection jobs use the naming convention collection_set_*N*_collection, where *N* is a number. The upload jobs use the naming convention collection_set_*N*_upload, where *N* is a number.

By default, the AMR tool collects data from three dynamic management views every 15 minutes for both the Stored Procedure Usage Analysis and Table Usage Analysis collection sets. The upload job runs every 30 minutes for the Stored Procedure Usage Analysis collection set and every 15 minutes for the Table Usage Analysis collection set. If you want to speed your upload, you can execute these jobs manually. Uploading the data has a minimal impact on performance.

Running the Transaction Performance Analysis Reports

To access the recommendations based on the information collected about all your user databases on the workload server, you need to run

the transaction performance analysis reports. To access them, right-click your Management Data Warehouse database, select Reports, choose Management Data Warehouse, and click Transaction Performance Analysis. From the Transaction Performance Analysis Overview page, you can choose to run three reports, depending on what type of information you need:

- Recommended Tables Based on Usage
- Recommended Tables Based on Contention
- Recommended Stored Procedures Based on Usage

Recommended Tables Based on Usage. This report tells you which tables are the best candidates for migration to In-Memory OLTP based on their usage. Figure 3 shows a sample report. On the left side, you can select the database and how many tables you'd like to see from that database. The chart will then show the selected tables. The horizontal axis represents the amount of work needed to migrate a

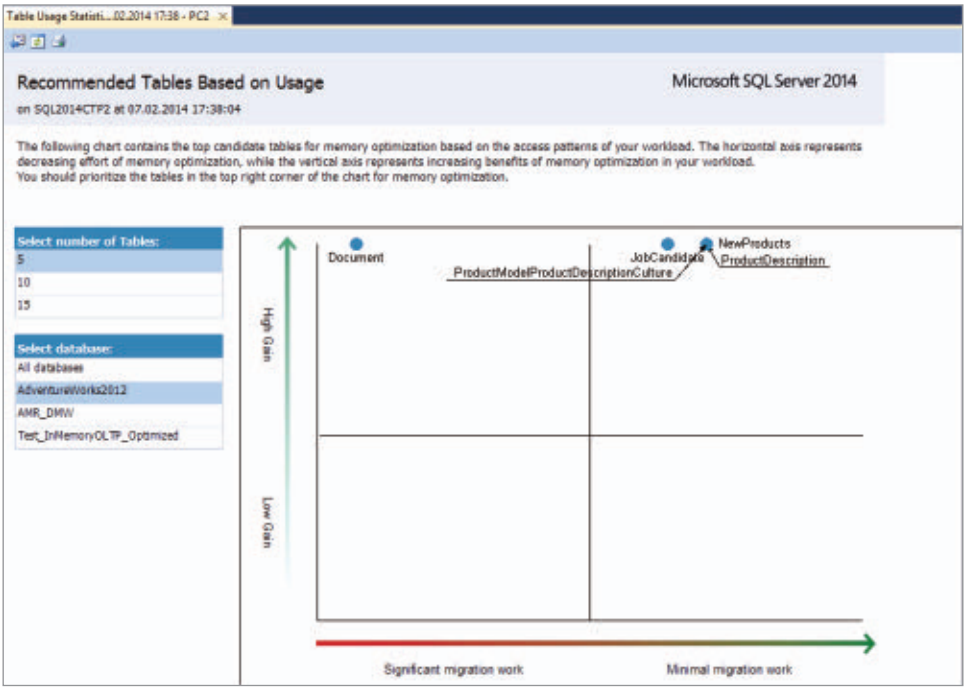
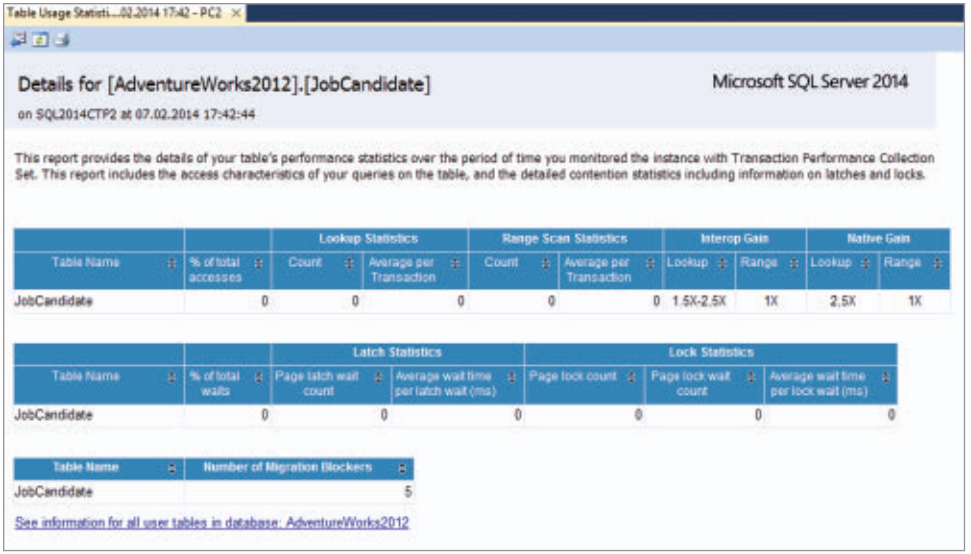


Figure 3
Determining Which Tables Are the Best Candidates for Migration Based on Usage

table to In-Memory OLTP. The vertical axis represents the gains you'll achieve after migrating the table. The best candidates for In-Memory OLTP are located in the top right corner. As you can see, they can be easily migrated and will give you the best performance gain.

You can access a detailed report for a table by clicking its name in the chart. As Figure 4 shows, this report provides the table's access statistics (e.g., lookups, range scan) and contention statistics (e.g., latches, locks), as well as when this information was captured.

Figure 4
Reviewing the Detailed Performance Statistics for a Table



Recommended Tables Based on Contention. This report tells you which tables are the best candidates for migration to In-Memory OLTP based on their contention. If you compare the contention analysis report in Figure 5 with the usage analysis report in Figure 3, you'll see that they're very similar. You can select the database and how many tables you'd like to see from that database. The resulting chart shows the amount of work needed to migrate the tables (horizontal axis) and the gains you'll achieve after migrating them (vertical axis). In the top right corner, you'll find the best candidates for migration based on contention. You can click a table name in the chart to access

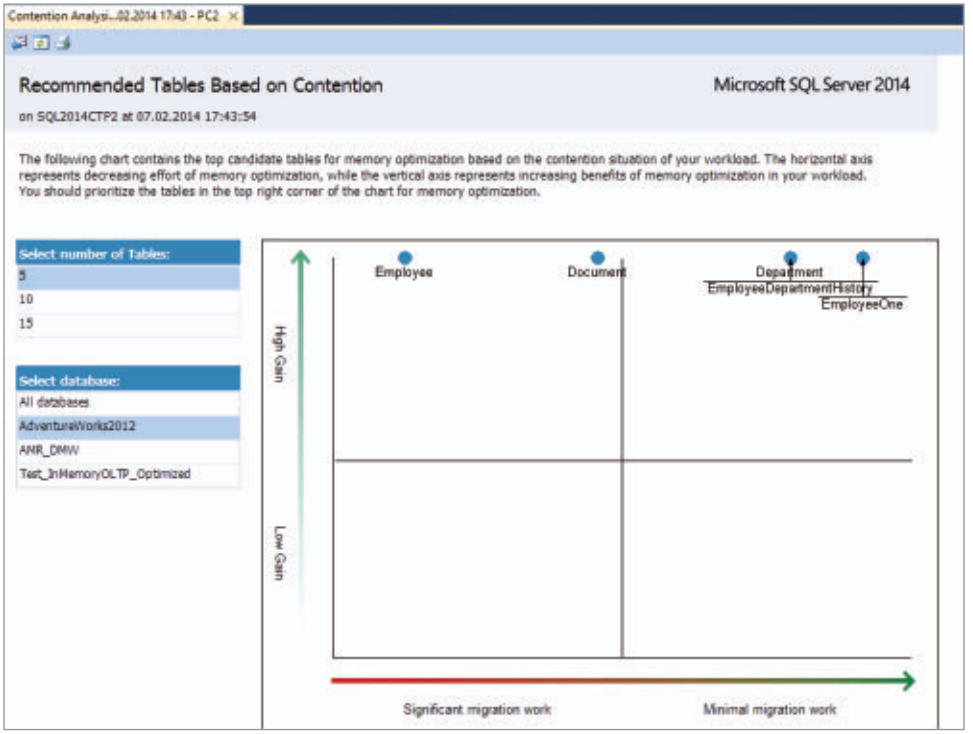


Figure 5
Determining Which Tables Are the Best Candidates for Migration Based on Contention

a detailed report showing the table's statistics. This report provides the table's access and contention statistics.

Recommended Stored Procedures Based on Usage. This report shows you which stored procedures are the top candidates for an In-Memory OLTP migration based on their usage (i.e., total CPU time). After selecting the database and how many stored procedures you'd like to see from that database, the resulting chart shows the top candidates for migration, as Figure 6 shows. If you want to see the detailed usage statistics for a specific stored procedure, you can click its blue bar. Figure 7 shows an example of the report you'll receive.

Using the Memory Optimization Advisor

After you know which tables you want to migrate to In-Memory OLTP, you can use the AMR tool's Memory Optimization Advisor to help you with the migration process. To access this advisor, open Object

Figure 6
Seeing Which Stored Procedures Are the Top Candidates for Migration Based on Usage

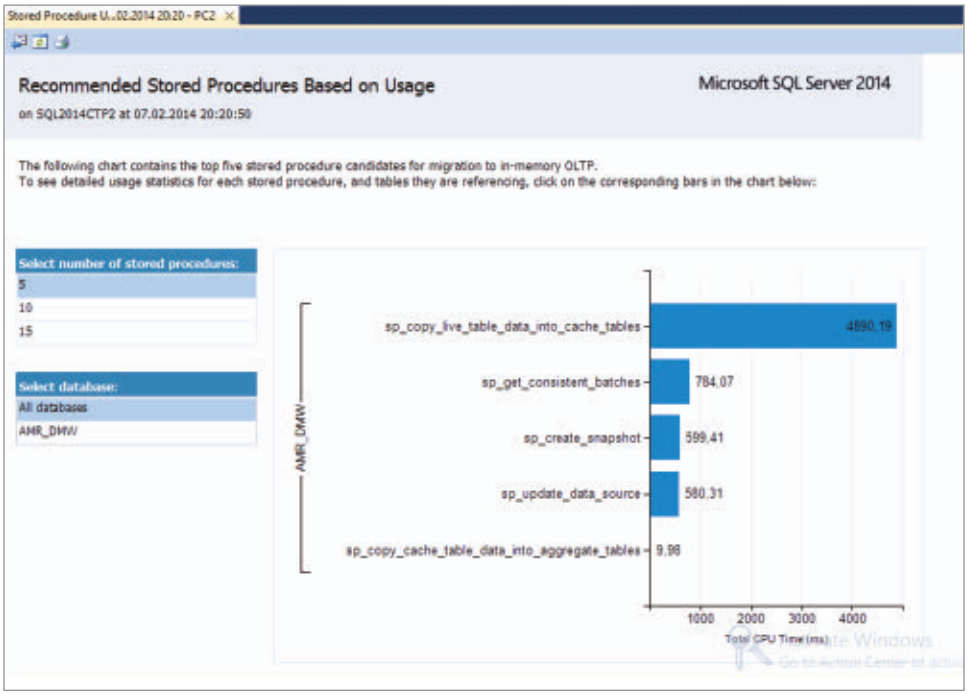
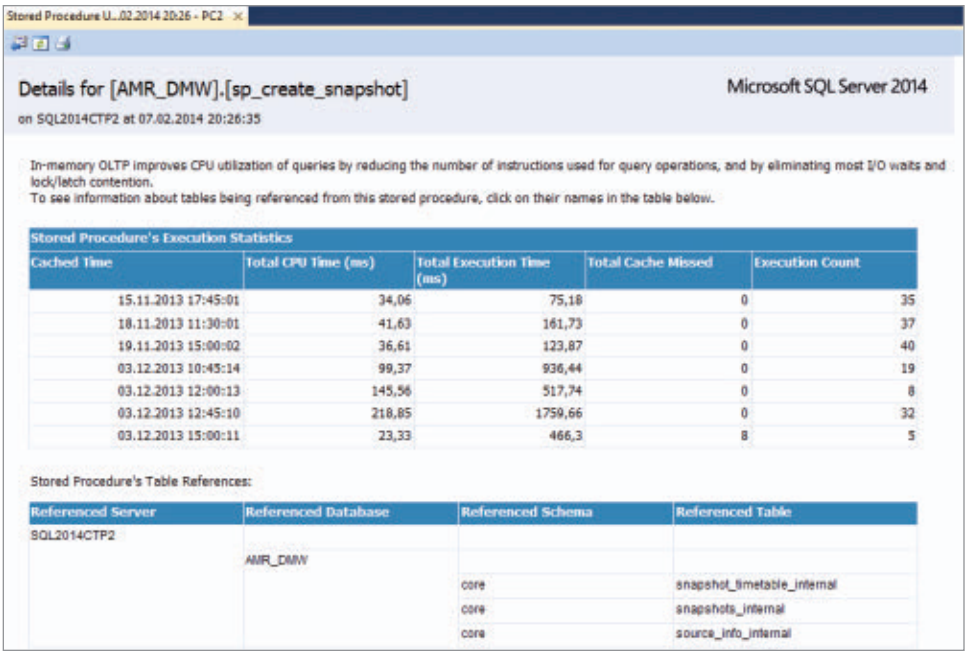


Figure 7
Reviewing the Detailed Usage Statistics for a Stored Procedure



Explorer in SSMS and navigate to the table you want to migrate. Right-click the table and choose Memory Optimization Advisor.

The advisor will launch with the Introduction page, which you can read or skip. Clicking Next brings you to the Migration Optimization Checklist page, where the advisor will check to see if your table can be migrated. If one or more validation items fail, the migration process will stop. If needed, you can generate a report for this analysis. If all you see are green checkmarks, your table doesn't have any features that could prevent the migration process, in which case you can proceed to the next page.

The Native Compilation Advisor doesn't migrate stored procedures like the Memory Optimization Advisor migrates tables.

On the Migration Optimization Warnings page, you'll find important information about what isn't supported in memory-optimized tables and other types of issues. The issues listed won't prevent the table from being migrated, but they might cause other objects to fail or behave in an unexpected manner.

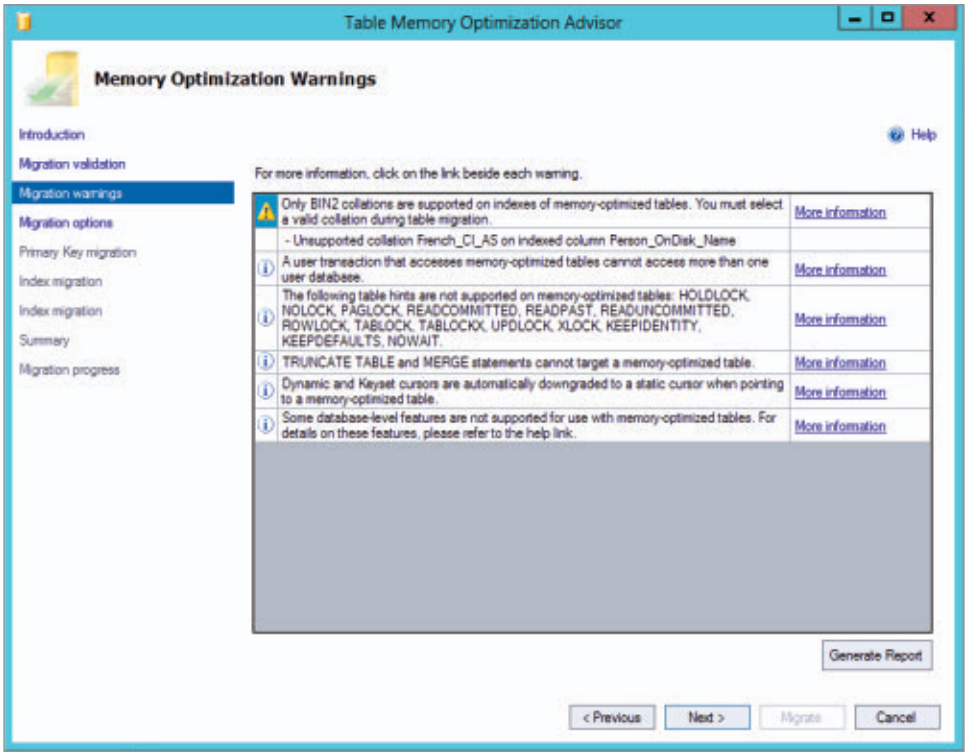
If a warning applies to the table you selected for migration, an exclamation point in a yellow triangle will appear next to the warning, as shown in Figure 8. In this case, the selected table has an unsupported French_CI_AS collation on the indexed column named Person_OnDisk_Name. (Only BIN2 collations are supported for indexes in memory-optimized tables.) Thus, the index collation will need to be changed later in the migration process.

On the Review Optimization Options page, which Figure 9 shows, you have the option to change the defaults listed for the:

- Name of memory-optimized file group (only one memory-optimized file group is allowed per instance)
- Logical filename
- Path where the logical file will be saved
- New name given to the original table (the original table is renamed to prevent naming conflicts)

You can also copy data from the original table to the new memory-optimized table, and you can change the durability of the memory-

Figure 8
Reviewing the Migration Optimization Warnings



optimized table. By default, its DURABILITY option is set to schema_and_data, but you can change it to schema_only by selecting *Check this box to migrate this table to a memory-optimized table with no data durability.* If you do so, the data will be lost after the SQL Server service is restarted. In other words, just the table's schema is persistent.

Finally, the Review Optimization Options page shows the estimated current memory cost for the memory-optimized table. If there isn't sufficient memory, the migration process might fail.

Once you're done with the Review Optimization Options page, you can click Next to go to the Review Primary Key Conversion page. When the migration process begins, it will start by converting the primary key. You can convert it to:

- A nonclustered hash index, which gives the best performance for point lookups. If you select this option, you also need to

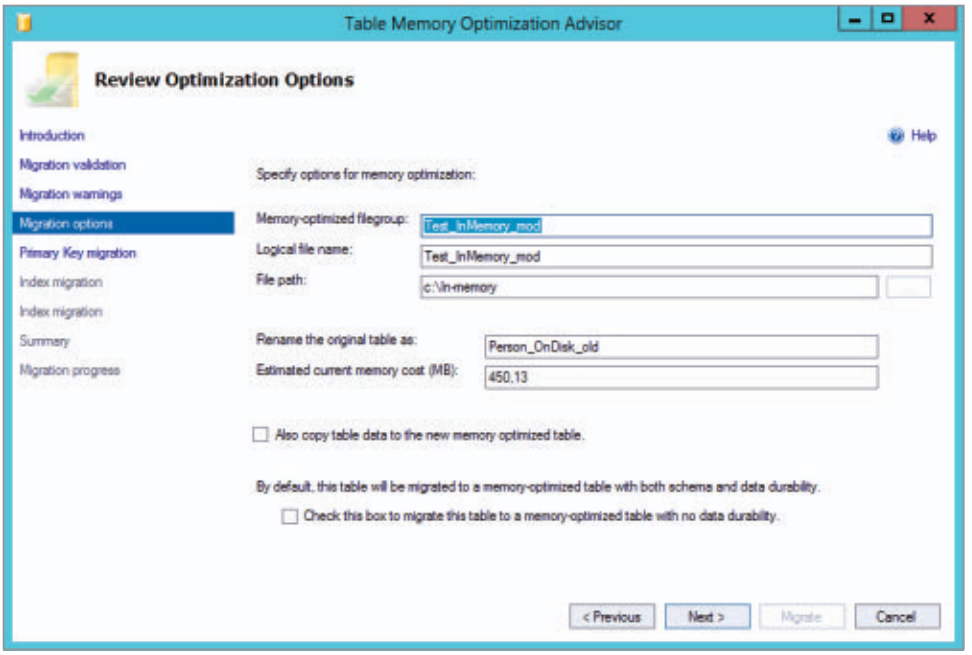


Figure 9
Reviewing the Optimization Options

specify the bucket count, which should be twice the expected number of rows.

- A nonclustered index, which gives the best performance for range predicates.

For each index you have in the table being migrated, you'll be presented with a Review Index Conversion page that has been populated with the columns and data types for that index. The options you can configure in the Review Index Conversion page are similar to those in the Review Primary Key Conversion page. In this case, for the indexed column Person_OnDisk_Name with the unsupported French_CI_AS collation, you'd have to select BIN2 collation as the Char data type.

On the Verify Migration Actions page, you'll see all operations that will be performed to migrate your table to In-Memory OLTP. You have the option to script those operations by clicking the Script button. After verifying all the options, you can click the Migrate button to start the migration process.

Figure 10
Memory-Optimized Table in SSMS

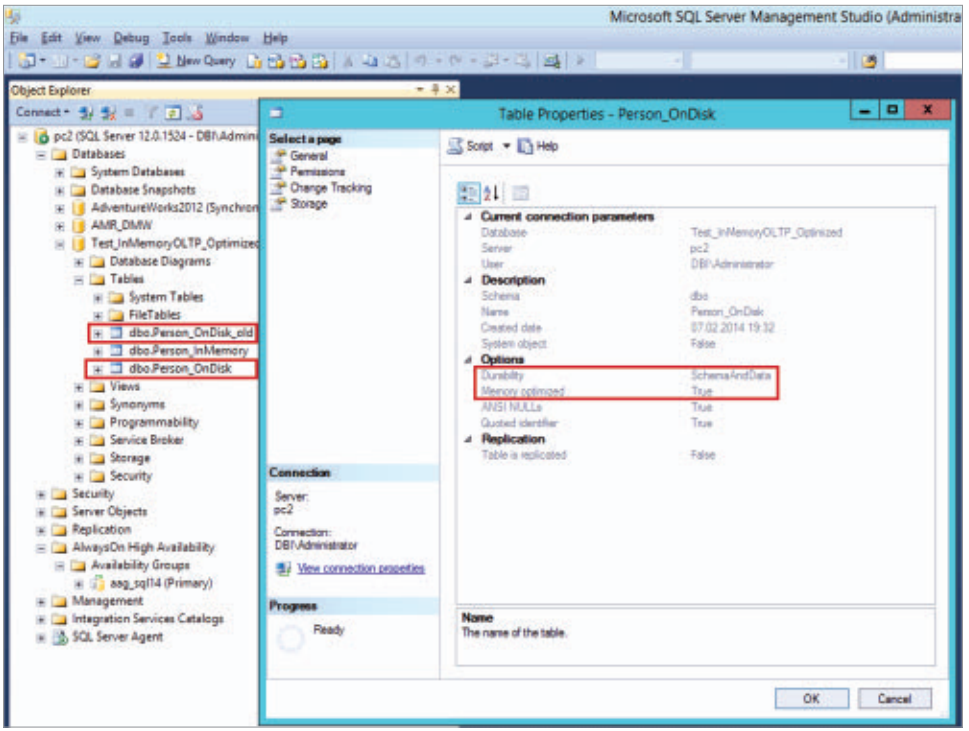


Figure 10 shows how the new memory-optimized table appears in SSMS. If you view its properties, you'll see that the *Memory optimized* property is set to True and that the schema and data are durable for this table. You can also see how the original table has been renamed.

Using the Native Compilation Advisor

After you know which stored procedures you want to migrate to In-Memory OLTP, you can use the AMR tool's Native Compilation Advisor to help you with their migration. To access this advisor, open Object Explorer in SSMS and navigate to the stored procedure you want to migrate. Right-click the stored procedure and choose Native Compilation Advisor.

After clicking through the Welcome page, you'll be presented with the Stored Procedure Validation page, which will give you warnings if your stored procedure contains some T-SQL elements that aren't

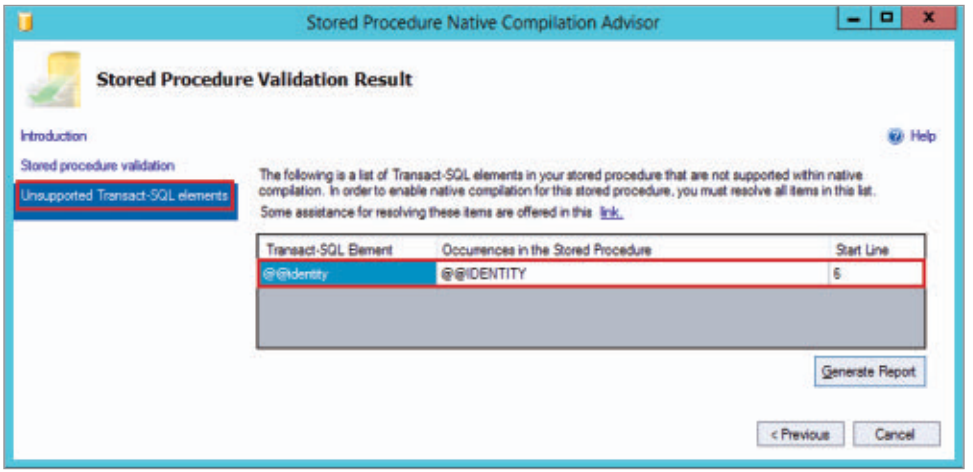


Figure 11
Reviewing the Unsupported T-SQL Elements

supported by native compilation. If the stored procedure is valid, it can become a natively compiled stored procedure without modification. However, the Native Compilation Advisor doesn't migrate stored procedures like the Memory Optimization Advisor migrates tables. You have to do the migration on your own.

If the stored procedure has unsupported T-SQL elements, the validation will fail. To see the details about the unsupported elements, you need to click Next to go to the Stored Procedure Validation Result page, which Figure 11 shows. You have to modify the unsupported elements before you can migrate your stored procedure to a natively compiled stored procedure.

Eliminate the Guesswork

The AMR tool is useful because it eliminates the guesswork in determining which tables and stored procedures would benefit from In-Memory OLTP. After identifying which tables to migrate, you can use the Memory Optimization Advisor to quickly migrate them. Although the Native Compilation Advisor can help you identify the T-SQL elements you need to change before migrating your stored procedure to a natively compiled one, it unfortunately doesn't guide you through the migration process. ■

SQL Server Database Corruption, Part IX: Responding to Corruption

Michael K. Campbell

is a contributing editor for *SQL Server Pro* and *Dev Pro*. He's a consultant with years of SQL Server DBA and developer experience. He enjoys consulting, development, and creating free videos for www.sqlservervideos.com.

Email

Twitter

Website

In "SQL Server Database Corruption, Part VIII: The Importance of Backups," I discussed the importance of backups when it comes to dealing with logical corruption (caused by a person or application) or physical corruption (caused by the I/O subsystem). Now let's take a look at some best practices for responding to physical corruption when it's detected.

Responding to Physical Corruption

When physical corruption is encountered, there's a definite order of operations to follow to decrease the potential for data loss and decrease downtime while remedying the situation. With that said, all forms of corruption aren't created equal when it comes to their potential scope or impact.

For example, corruption might simply occur within a nonclustered index or within another structure that's essentially an ancillary feature of your database. In this case, you might be able to recover from the corruption by merely dropping the index and re-creating it, with no downtime and only a brief disruption when you're working with the locks or blocks necessary to replace the index. In other situations, corruption might occur within data pages that store the data that's needed by your database. Even worse, corruption might occur within the allocation pages that are used by the database to map where your data, indexes, and other key structures are found. When corruption occurs in these highly specialized pages, it can be fatal, which means that there's no option for recovery other than to revert to backups.

Editor's Note

This article is the ninth in a series. For the rest of the series, go to the *SQL Server Pro* website.

The following ordered list outlines what you should (and shouldn't) do to minimize the scope and downtime associated with recovering from corruption.

1. Don't reboot your server. Although some Windows problems will go away if you reboot the box, the presence of corruption won't go away if you reboot a SQL Server box. Instead, it's very possible that a reboot will make things much worse. A reboot might cause SQL Server to take your database offline and render it suspect, making recovery much harder.

2. Don't attempt to detach, then re-attach your databases. This is an absolute worst practice (no matter what you might have read in some online forums). Detaching, then re-attaching your database when corruption is present is almost certainly going to make recovery much harder.

3. Determine the nature and scope of the corruption before attempting any repairs. If you're getting I/O errors (e.g., error 823, 824, or 825) from SQL Server, you should run the following command as soon as possible to see if there are any problems:

```
DBCC CHECKDB(yourDbName) WITH NO_INFOMSGS, ALL_ERRORMSGS
```

(You'll need to replace *yourDbName* with the name of your database.) If you already found problems by means of regular checks or backups, you should evaluate all the error messages, save them in a .txt or .sql file, and store the file in a safe location.

4. Consider rerunning DBCC CHECKDB if it initially reported only a minor problem. This might sound lame or even superstitious, but in rare cases, you can actually rerun DBCC CHECKDB and have the minor corruption (i.e., just one or two errors) disappear. This occurs when SQL Server wrote to a bad spot (or the write was mangled) and SQL Server was able to transparently recover by writing to a new location or executing a new write that completed correctly. Such scenarios are very rare, but they do happen. So, if you can rerun

DBCC CHECKDB quickly, it might be in your best interest to try this option—just don't get your hopes up too high that the problem will magically disappear.

In the event that rerunning DBCC CHECKDB shows the problem has been fixed, you still need to watch for a pattern. Although one instance of corruption here or there is sadly expected, multiple problems over a short period of time typically indicates a pattern of systemic problems, in which case you need to start contemplating replacing the hardware as needed.

5. Size up corruption and remediation options before doing anything. There are simply too many horror stories out there about DBAs who jumped the gun at the first sign of trouble and either destroyed data unnecessarily or increased downtime by responding to symptoms of corruption instead of the root causes. Accordingly, make sure you review every single error *after* SQL Server completes its error reporting. In some cases, the screen might be chock-full of red error messages about corruption in various pages, but you might find out at the end of the check that almost all the errors are being caused by a non-clustered index that can be simply dropped and re-created. So, don't jump the gun.

Similarly, some forms of corruption are simply the result of the metadata used by SQL Server getting out of sync with reality. In these cases, SQL Server might tell you that you can run DBCC UPDATEUSAGE or another operation to safely and easily recover from corruption without incurring any downtime or losing any data. So again, make sure you have all the facts and size up your options accordingly.

6. Validate any repair options in a test environment first. Unless you spend gobs of time working with corruption, the various modes and types of repair are going to be things that you're going to potentially cut your teeth on, which means you'll probably learn some great and valuable lessons the hard way. So why do that on your production data? Copy your database somewhere, then practice and validate your intended changes there. Yes, this will probably take longer

in most cases than just executing your changes in production. Then again, if you screw something up in your production database, which approach ends up taking longer? (Personally, I always approach every recovery operation by looking for options that leave me with the most fallback capabilities and contingencies possible.)

7. Run DBCC CHECKDB with the REPAIR_REBUILD option. In many cases when corruption is minimal, SQL Server will inform you that running DBCC CHECKDB with the REPAIR_REBUILD option might be a viable approach to recovering data. If this is the case or if you just want to see whether it will work, you can safely run the command

```
DBCC CHECKDB(yourDbName) WITH REPAIR_REBUILD
```

with no worries about data loss. (You'll need to replace *yourDbName* with the name of your database.) The only thing you stand to lose would be time, because you must put the database into Single User mode to execute this option. So, if you think this command has a potential to correct your errors, it's a viable approach. If SQL Server indicates something more aggressive is needed (e.g., you need to use backups or use repair options that result in data loss), running this command will just waste time.

8. Execute page-level restore operations if possible. If you have full-blown data corruption within a few pages (as opposed to the corruption being in indexes), you'll often be able to spot those pages by querying msdb's suspect pages table, like so:

```
SELECT * FROM msdb..suspect_pages
GO
```

After you know which pages are corrupt, you can effectuate a page-level restore from SQL Server using your existing backups. This means that you'll need to instruct SQL Server to "reach in" to

previous backups, grab the data from the pages that were corrupt, then replace the corrupt pages with known-safe pages from your backups. More important, because any operations since your last backup will also have been logged (assuming you perform regular full backups—and possibly differential backups—along with transaction log backups), you'll be able to replay any changes against that data by means of replaying the data in the transaction log. (For more information about how this works, see the free video "SQL Server Logging Essentials.") Make sure that you back up the tail end of your transaction log before beginning this operation to make sure that you don't lose any transactions that haven't been backed up. (I show you how to do this in "SQL Server Database Corruption, Part X: Page Level Restore Operations.")

Video

"SQL Server Logging Essentials"

9. Execute a full recovery. If there are large numbers of suspect or corrupted pages (i.e., so many that manually recovering each one would take longer than a full recovery) or if critical system pages have been destroyed, you'll need to execute a full recovery. Like a page-level restore operation, the full recovery operation must begin with a backup of the tail end of the current transaction log. (I'll follow up with an example of how to do this in a subsequent article.)

10. Use DBCC CHECKDB with the REPAIR_ALLOW_DATA_LOSS option. Using the REPAIR_ALLOW_DATA_LOSS option will result in the loss of data, so it's not recommended. Furthermore, if you're going to use this option, Microsoft recommends that you initiate a full backup beforehand, because once the operation completes, you'll have no other way to recover the data you lost.

These caveats beg the question, Why would anyone want to use this technique? There are two possible reasons:

- You need to use this technique if you don't have any existing backups to use to recover from major corruption. (Therefore, make sure that you always have viable backups!)
- In some cases, organizations actually favor uptime over data continuity, or *data purity*, when it comes to their databases. In

other words, these organizations would prefer to avoid downtime at the expense of data purity. In these cases, there are advanced scenarios in which the use of the REPAIR_ALLOW_DATA_LOSS option might be acceptable, assuming the organizations understand the trade-offs. For more information about these cases, check out my article "SQL Server Database Corruption, Part VIII: The Importance of Backups."

Scary and Ugly, But Not Fatal

Addressing problems of corruption can be scary and ugly, but not fatal, if you keep a level head, understand what's going on, and have regularly tested backups to fall back on. Moreover, if you regularly practice the techniques I'll discuss in upcoming articles, you'll be able to handle any sort of disaster—including corruption—much better than if you wait until a disaster strikes to learn these techniques. ■

Learning Path

***SQL SERVER PRO* RESOURCES**

For the previous articles in this series, see:

"SQL Server Database Corruption, Part I: What Is Corruption?"
"SQL Server Database Corruption, Part II: Simulating Corruption"
"SQL Server Database Corruption, Part III: Preventing Corruption"
"SQL Server Database Corruption, Part IV: Checksum Page Verification"
"SQL Server Database Corruption, Part V: Storage Problem Alerts"
"SQL Server Database Corruption, Part VI: Regular Corruption Checks"
"SQL Server Database Corruption, Part VII: Backups"
"SQL Server Database Corruption, Part VIII: The Importance of Backups"

Visual Studio Online FAQs

Michael Otey

is senior technical director for *Windows IT Pro* and *SQL Server Pro*.

Email

It's no surprise that Microsoft's recent shift in focus from software to devices and services has resulted in more of its software catalog being transformed from on-premises software to cloud-based services. Visual Studio Online is one of Microsoft's latest online cloud-based releases. Before you worry that Microsoft is taking away your favorite development tool and moving it to the cloud, be aware that the name "Visual Studio Online" is something of a misnomer; Visual Studio Online isn't the cloud-based version of the Visual Studio development environment. Here are answers to FAQs about Visual Studio Online.

Q: What is Visual Studio Online?

A: Visual Studio Online isn't a hosted or web-based version of the Visual Studio desktop code IDE. Visual Studio Online is the updated version of Team Foundation Service. Visual Studio Online provides developers with a hosted team development platform and code repository and eliminates the need to set up any on-premises infrastructure to support team development. Visual Studio Online runs on Windows Azure, and you can connect to it from some of the most popular development tools, including Visual Studio, Eclipse, and Xcode. A separate cloud development product, code-named Monaco, will allow code development.

Q: What features are available in Visual Studio Online?

A: The primary purpose of Visual Studio Online is to support team development. To that end, the updated development platform provides features that were formerly in Team Foundation Service, including hosted source control, work item tracking, Agile project planning, collaboration, build services, and load-testing services. A new Application Insights feature lets you monitor web

applications, as well as collect application performance and telemetry data. Although it's probably somewhat redundant considering the nature of the tool, Visual Studio Online also includes the free Visual Studio Express for Web, Windows, or Windows Desktop versions.

Q: How is Visual Studio Online different from the on-premises Team Foundation Server (TFS)?

A: Visual Studio Online is essentially the newer, cloud-based version of TFS. Microsoft's new cloud-first methodology means that the cloud versions of products will be on a faster update cycle, with the newest features and fixes being introduced to the cloud products first. Although Microsoft has rapidly accelerated the update cycle for Visual Studio, the development tool is essentially on a quarterly update cycle and will not be updated at the same rate that the online products will be.

Q: How much does Visual Studio Online cost?

A: Visual Studio Online is available in three different versions. Each version supports team projects, hosted code repositories, project planning, bug tracking, and IDE integration. As of February 2014, Visual Studio Online is in preview mode and the pricing is set at a 50 percent discount. The three different Visual Studio Online versions are as follows:

- Visual Studio Online Basic—The basic version is free and supports up to five users; additional users are $10 per month.
- Visual Studio Online Professional—A level above the basic version, the professional version supports a maximum of 10 users per account for $22.50 per month. This version includes Visual Studio Professional IDE.
- Visual Studio Online Advanced—This version supports an unlimited number of users per account at $30 per month. It also includes Team Rooms and Agile Portfolio Management features.

Q: Since Visual Studio Online is a developer tool, are there any benefits to being an MSDN subscriber?

A: Visual Studio Premium with MSDN provides all the features of Visual Studio Online Advanced, and more. MSDN subscribers with the following types of subscriptions can be added at no charge to a Visual Studio Online account:

- Visual Studio Professional with MSDN
- Visual Studio Test Professional with MSDN
- MSDN Platforms
- Visual Studio Premium with MSDN
- Visual Studio Ultimate with MSDN

In addition, users with Visual Studio Ultimate 2013 or later with MSDN get 15,000 user minutes of free load-testing each month. You can find out more on the Visual Studio Online web page. ■

SQL Server Pro, April 2014, Vol. 16, No. 4, ISSN 1522-2187. *SQL Server Pro* is published monthly by Penton. Copyright ©2013 Penton. All rights reserved. No part of this publication may be reproduced or distributed in any way without the written consent of Penton.

SQL Server Pro, 748 Whalers Way, Fort Collins, CO 80525, 800-621-1544 or 970-663-4700. Customer Service: 800-793-5697.

We welcome your comments and suggestions about the content of *SQL Server Pro*. We reserve the right to edit all submissions. Letters should include your name and address. Please direct all letters to Megan Keller at megan.keller@penton.com. SQL Server pros interested in writing for *SQL Server Pro* can submit articles to articles@sqlmag.com.

Every effort has been made to ensure examples in this publication are accurate. It is the reader's responsibility to ensure procedures and techniques used from this publication are accurate and appropriate for the user's installation. No warranty is implied or expressed. Please back up your files before you run a new procedure or program or make significant changes to disk files, and be sure to test all procedures and programs before putting them into production.

SQL Server is a registered trademark of Microsoft Corporation, and *SQL Server Pro* is used by Penton, under license from owner. *SQL Server Pro* is an independent publication not affiliated with Microsoft Corporation. Microsoft Corporation is not responsible in any way for the editorial policy or other contents of the publication.