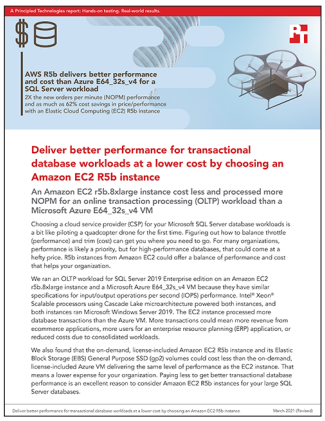

The science behind the report:

Deliver better performance for transactional database workloads at a lower cost by choosing an Amazon EC2 R5b instance

This document describes what we tested, how we tested, and what we found. To learn how these facts translate into real-world benefits, read the report Deliver better performance for transactional database workloads at a lower cost by choosing an Amazon EC2 R5b instance.

We concluded our hands-on testing on January 28, 2021. During testing, we determined the appropriate hardware and software configurations and applied updates as they became available. The results in this report reflect configurations that we finalized on December 18, 2020 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

To learn more about how we have calculated the wins in this report, go to http://facts.pt/calculating-and-highlighting-wins. Unless we state otherwise, we have followed the rules and principles we outline in that document.

Table 1: New orders per minute (NOPM) results from our testing of the two instances with a TPC-C-like workload from HammerDB.

	Amazon Elastic Cloud Computing (EC2) r5b.8xlarge	Azure E64_32s_v4
NOPM	1,468,743	720,157

As Table 2 shows instance and VM costs, Table 3 shows storage costs. We used the cost calculators provided by each vendor to estimate the cost for each solution using an on-demand model. To ensure instance costs included Microsoft Windows Server and Microsoft SQL Server 2019 Enterprise edition licensing, in the AWS pricing calculator, we selected "Windows Server with SQL Server Enterprise." In the Azure pricing calculator, we chose "SQL Server" for the Type dropdown and "SQL Enterprise" for the License dropdown. In the Azure calculator, we chose the pricing for the E64s v4 instance since this was the underlying VM the pricing is based on according to Azure: "The licensing charged for SQL Server will be constrained to the active vCPU(s) count while the compute cost, which includes OS licensing, remains the same one as the original size based on "underlying vCPU(s)."[1] To calculate SQL Server licensing cost on the constrained-CPU Azure instance with 32 vCPUs instead of 64, we took the listed 64 vCPU SQL Server licensing cost estimate ($17,520) and halved it ($8,670) since Azure charges SQL server licensing only on active vCPUs.

1 Microsoft, "Windows Virtual Machines Pricing," accessed March 3, 2021, https://azure.microsoft.com/en-us/pricing/details/virtual-machines/windows/

We found that costs for the Elastic Block Storage (EBS) General Purpose SSD (gp2) volumes for the EC2 instance were lower than the storage costs for the Azure VM. To match the reported input/output (I/O) performance of the drives for the two instances, we selected a larger storage capacity for the Azure VM than we did for the EC2 instance.[2,3] For a 730-hour (24/7 usage) month, storage for the EC2 instance cost 43 percent less than the Azure VM. The Azure managed disks offered 53 percent more storage than the EBS storage, but the EC2 instance with EBS storage cost 24 percent less than the Azure VM with managed disks we tested. If we reduce the cost of the Azure storage by the difference in storage, something customers cannot do, EBS still costs 13 percent less per storage GB than the Azure managed disks.

With combined instance and storage costs for a 730-hour month,[4] the EC2 instance cost 24 percent less than the Azure VM, saving more than $4,700 for a single month of 24/7 usage.[5,6]

Table 2: Pricing information and comparison for the two instances we tested.

	Amazon EC2 r5b.8xlarge	Azure E64_32s_v4	Percentage lower cost for the EC2 instance	Cost difference in dollars
730-hr usage scenario				
Instance hourly cost	$15.86	$18.98	16.4%	$3.12
Instance 730hr cost	$11,574.88	$13,852.48	16.4%	$2,277.60
Storage monthly cost 730hr	$3,210.40	$5,676.48	43.4%	$2,466.08
Total (730hr)	$14,785.28	$19,528.96	24.3%	$4,743.68
240-hr usage scenario				
Instance hourly cost	$15.86	$18.98	16.4%	$3.12
Instance 240hr cost	$3,805.44	$4,554.24	16.4%	$748.80
Storage monthly cost 240hr	$1,054.09	$1,866.24	43.5%	$812.15
Total (240hr)	$4,859.53	$6,420.48	24.3%	$1,560.95

Table 3: Pricing information and comparison for the storage of the two instances we tested.

	Amazon EC2 r5b.8xlarge	Azure E64_32s_v4
Azure E64_32s_v4 monthly storage cost (730hr)	$3,210.40	$5,676.48
Total GB storage	32,004	49,152
Per GB cost	$0.10	$0.12
Storage difference in GB (Azure)		17,148
Calculated cost for additional storage (Azure)		$1,980.39
Adjusted cost assuming equal storage in GB (Azure)		$3,696.09
Adjusted percentage lower cost for the EC2 instance		13.1%

2 Amazon, "Amazon EBS pricing," accessed February 2, 2021, https://aws.amazon.com/ebs/pricing/.
3 Microsoft, "Managed Disks pricing," accessed February 2, 2021, https://azure.microsoft.com/en-us/pricing/details/managed-disks/.
4 Instance pricing from both CSPs did not include operating system or SQL Server costs. To include those, we used each CSP's pricing calculator.
5 Amazon, "AWS Pricing Calculator," accessed February 2, 2021, https://calculator.aws/#/.
6 Microsoft, "Pricing calculator," accessed February 2, 2021, https://azure.microsoft.com/en-us/pricing/calculator/.

System configuration information

Table 4: Detailed information on the system we tested.

System configuration information	32vCPU AWS EC2 r5b.8xlarge instance	64vCPU Azure E64_32s_v4 VM
Tested by	Principled Technologies	Principled Technologies
Test date	12/23/2020	12/23/2020
Cloud service provider (CSP) / Region	AWS us-east-1b	Microsoft Azure East US (Zone 1)
Workload & version	HammerDB v3.3 TPC-C-like	HammerDB v3.3 TPC-C-like
WL specific parameters	MAXDOP 1, Lock Pages in Memory, Perform Volume Maintenance Tasks, 90% Reserved SQL Memory	MAXDOP 1, Lock Pages in Memory, Perform Volume Maintenance Tasks, 90% Reserved SQL Memory
Iterations and result choice	3 runs, median	3 runs, median
Server platform	r5b.8xlarge	E64_32s_v4
BIOS name and version	Amazon EC2 1.0, 10/16/2017	American Megatrends Inc. 090008, 12/7/2018
Operating system name and version/build number	Microsoft Windows Server 2019 Datacenter 10.0.17763/Build 17763	Microsoft Windows Server 2019 Datacenter 10.0.17763/Build 17763
Date of last OS updates/patches applied	10/09/2020	10/09/2020
Processor		
Number of processors	1	2
Vendor and model	Intel® Xeon® Platinum 8259CL	Intel Xeon Platinum 8272CL
Core count (per processor)	24	26
Core frequency (GHz)	2.50	2.60
Stepping	7	7
Hyper-Threading	Yes	Yes
Turbo	Yes	Yes
Number of vCPU per VM	32	64/32
Memory module(s)		
Total memory in system (GB)	253	504
NVMe memory present?	No	No
Total memory (DDR + NVMe RAM)	253	504
General HW		
Storage: NW or Direct Att / Instance	NW Att	NW Att
Network BW / Instance	N/A	N/A
Storage BW / Instance	N/A	N/A

System configuration information	32vCPU AWS EC2 r5b.8xlarge instance	64vCPU Azure E64_32s_v4 VM
Local storage		
OS		
Number of drives	1	1
Drive size (GB)	100	127
Drive information (speed, interface, type)	gp2	Standard SSD
Data drive		
Number of drives	6	6
Drive size (GB)	5,334	8,192
Drive information (speed, interface, type)	gp2	Premium SSD (P60)
Temporary drive		
Number of drives	0	0
Drive size (GB)	N/A	N/A
Network adapter		
Vendor and model	Amazon Elastic Network Adapter	Microsoft Hyper-V Network Adapter
Number and type of ports	1 x 25Gb	1 x 50Gb

How we tested the AWS instance

For our testing, we used two VMs on each CSP – a client VM and a server VM. We created a single base VM (see "Creating the Windows Server 2019 baseline image") and saved an image from this VM for the client VM (see "Creating an AMI of your baseline client instance"). With the base VM, we then installed SQL Server 2019 Enterprise Edition, saved a new image of the VM, and used the new image as our base for the second VM.

Creating a database backup

Creating the TPC-C-like database

1. Open SQL Server Management Studio.
2. Right-click Databases, and choose New Database….
3. Name the database.
4. Click Filegroups, and click Add Filegroup.
5. Name the new filegroup, and set it to Default.
6. In the general tab, click Add to create 1 additional data file.
7. Name each file, and set the initial size for the new file to 200GB. Make sure the file is on the new Filegroup you created.
8. If necessary, change all data and log file paths to point to the data and log drives on the system.
9. On the options tab, set the Recovery Model to simple.
10. To start the build, click OK. Generating the data and loading it into the database
11. Open HammerDB.
12. Select Options→Benchmark.
13. Choose MSSQL Server, and TPC-C.
14. Expand SQl Server→TPC-C→Schema Build.
15. Double-click Options, enter the IP of the SQL Server, change the driver to ODBC Driver 17 for SQL Server, enter the SQL Server Database name, set Number of Warehouses to 1,000, and select an appropriate number of Virtual Users to Build Schema. Click OK.
16. To begin the build, double-click Build.

Backing up the database

1. Open Microsoft SQL Management Studio, and right-click the TPC-C database.
2. Navigate to tasks→Backup.
3. To set the name of your backup file and the location to store the file, on General, click Add….
4. In the left pane, click Backup Options.
5. Under Compression, select Compress backup.
6. Click Ok.

Creating the Windows Server 2019 baseline image

Creating the baseline image instance

1. Log into AWS, and navigate to the AWS Management Console.
2. Click EC2
3. Click Launch instance, and to open the Launch Instance in the dropdown wizard, click Launch instance.
4. In the search window, type Windows Server, and press enter.
5. On Quick Start, click the Select button next to Microsoft Windows Server 2019 Base.
6. On Choose Instance Type, select t2.micro, and click "Next: Configure Instance Details."On Configure Instance, set the following:
    a. Number of instances: 1
    b. Purchasing option: Leave unchecked
    c. Network: Default VPCSubnet: Choose the region you are working inAuto-assign Public IP: EnablePlacement Group: Leave uncheckedCapacity Reservation: Open
    d. Domain join directory: No Directory
    e. IAM role: None
    f. Shutdown behavior: Stop
    g. Click Next: Add StorageOn Add Storage, set the following:
    h. Size: 30GB
    i. Volume Type: gp2
    j. Delete on Termination: Checked
    k. Encryption: Not Encrypted
    l. Click Next: Add Tags

7. On Add Tags, add any tags you wish to use. Click Next: Configure Security Group.
8. On Configure Security Group, leave defaults, and click Review and Launch.
9. On Review, click Launch.
10. Choose the appropriate option for the key pair, and click Launch Instances.

Configuring Windows 2019

1. Open Server Manager, and click Local Server.
2. Disable IE Enhanced Security Configuration.
3. Change the time zone to your local time zone.
4. Change the name of your server, and when prompted, reboot.
5. Open Server Manager again, and click Local Server.
6. Click to run updates.
7. Run updates, rebooting when prompted, until the server shows no new updates to install.

Installing HammerDB 3.3

1. Navigate to https://hammerdb.com/download.html, and download HammerDB.
2. Double-click the .exe file, choose English, and click OK.
3. Click Yes.
4. Click Next.
5. Chose a destination location, and click Next.
6. Click Next.
7. Click Finish.

Creating an AMI of your baseline client instance

1. Log into AWS, and navigate to the AWS Management Console.
2. Click EC2.
3. Click Running instances.
4. Place a checkmark next to the instance from which you wish to create an image.
5. Click Action, and select Image→Create Image.
6. Enter the Image name, and click Create Image.
7. To see your new image, in the menu on the left side of the page, navigate to Images -> AMIs.

Adding networking rules to security group

1. Log into AWS, and navigate to the AWS Management Console.
2. Click EC2.
3. Click Security Groups.
4. Choose your newly created Security Group.
5. Under Inbound rules, click Edit inbound rules.
6. Click Add Rule.
7. Under Type, select All traffic.
8. Under Source, select the Security Group.
9. Click Add Rule.
10. Under Type, select RDP.
11. Under source, select My IP.
12. Click Save Rules.

Installing Microsoft SQL Server 2019 Enterprise

Note that each CSP has dropdown options during instance creation to pre-install SQL Server Enterprise edition. We chose to install SQL Server Enterprise edition manually.

1. Using the baseline instance, create a new VM of the applicable instance type.
2. Download or copy the ISO to the server, and unzip it.
3. Double-click the Setup application.
4. Click Installation→New SQL Server Standalone installation or add features to an existing installation.

5. Choose the trial version, and click Next.
6. Check the I accept the license terms and Privacy Statement box, and click Next.
7. Check the Use Microsoft Update to check for updates (recommended) box, and click Next.
8. On the Install Rules page, click Next.
9. Check the boxes for the following features, and click Next:
    a. Database Engine Services
    b. Full-Test and Semantic Extractions for Search
    c. Client Tools Connectivity
    d. Client Tools Backwards Compatibility
10. Leave the Default instance, and click Next.
11. Leave the default Service Accounts, and click Next.
12. On the Server Configuration tab, choose Mixed Mode, and enter and confirm a Password for the SQL Server system administrator (sa) account.
13. Click Add Current User to Specify the SQL Server administrators.
14. Click Next.
15. Once you've passed the rule check, click Next.
16. Click Install.
17. When the install finishes, navigate to the SQL Server Installation Center, and click Install SQL Server Management Tools.
18. Download the SSMS file, and install with defaults.
19. When prompted, reboot the serverd.
20. To ensure there aren't any new updates for SQL, run Windows Update once more (make sure Windows Updates are set to get updates for other Microsoft products).
21. To disable Windows Update service once you've installed all available updates, click the Start button , type `services` to open the Services list, and disable the Windows Update service.

Editing MSSQLSERVER User Rights

1. Click Start, and type Local Security Policy. When the program appears in the search, open it.
2. Expand Local Policies, and click User Rights Assignment.
3. In the right-hand pane, scroll down, and double-click Lock pages in memory.
4. Click Add User or Group, type `NT Service\MSSQLSERVER`, and click OK.
5. To close the Properties window, click OK.
6. Double-click Perform Volume Maintenance Tasks.
7. Click Add User or Group, type `NT Service\MSSQLSERVER`, and click OK.
8. To close the Properties window, click OK, and close the Local Security Policy window.

Configuring SQL on the baseline image

Setting the SQL memory reserve and max degree of parallelism (MAXDOP)

1. Open the SQL Server Management Studio.
2. Right-click the SQL Instance, and click Properties.
3. Click Advanced node, scroll down to the Max Degree of Parallelism, and change the value to 1. Click OK.
4. Right-click the SQL Instance again, and navigate to Memory.
5. Set the Max Memory to 90% of the total memory in the system. Click OK, and close the Properties window.
6. Right-click the SQL instance, and restart the service. When prompted, click Yes.

Configuring the tempdb database

1. Open the SQL Server Management Studio.
2. Expand Databases and System databases, and right-click tempdb.
3. Add files, and change the starting size as necessary. We used eight 1GB files.
4. Right-click the SQL instance, and restart the service. When prompted, click Yes.

Creating an AMI of your baseline SQL instance

1. Log into AWS, and navigate to the AWS Management Console.
2. Click EC2.
3. Click Running instances.
4. Place a checkmark next to the instance from which you wish to create an image.
5. Click Action, and select Image -> Create Image.
6. Enter the image name, and click Create Image.
7. To see your new image, in the menu on the left side of the page, navigate to Images -> AMI.

Creating your instances with the baseline images

Creating the instance from your image

1. Log into AWS, and navigate to the AWS Management Console.
2. Click EC2.
3. Click on Images→AMIs.
4. Check the box next to the image you created in the previous step, and click Launch.
5. On Choose Instance Type, select your VM size, and click Next: Configure Instance Details.
6. On Configure Instance, set the following:
7. Number of instances: 1
8. Purchasing option: Leave uncheckedNetwork: Default VPCSubnet: Choose the region you are working inAuto-assign Public IP: EnablePlacement Group: Leave uncheckedCapacity Reservation: Open
9. Domain join directory: No Directory
10. IAM role: None
11. Shutdown behavior: Stop
12. Click Next: Add StorageOn Add Storage, set the following:
13. Size: <Size>
14. Volume Type: Set your volume typeDelete on Termination: UncheckedEncryption: Not EncryptedClick Next: Add Tags
15. On Add Tags, add any tags you wish to use. Click Next: Configure Security Group.
16. On the Configure Security Group tab, leave the defaults, and click Review and Launch.
17. On the Review Tab, click Launch.
18. Choose the appropriate option for the key pair, and click Launch Instances.
19. Using the client AMI to create a client instance, repeat steps 3 through 11.

Creating, attaching, and formatting data disks

Creating and attaching data disks

1. Log into AWS, and navigate to the AWS Management Console.
2. Click EC2.
3. In the navigation pane, choose Elastic Block Store, Volumes.
4. Click Create Volume.
5. Under Volume Type, select General Purpose SSD (gp2).
6. Set size to 5,334GiB (the minimum size to max out IOPs on gp2).
7. Choose your desired Availability Zone.
8. Add any Tags you wish to use.
9. Click Create Volume.
10. On the Elastic Block Store, Volumes page, select your newly created drive.
11. Click Actions, and select Attach Volume.
12. From the list, choose your instance, and click Attach.
13. Repeat steps 4 through 12 until you have six attached data disks.

Partitioning and formatting the drives

1. Log into the VM under test.
2. Open a Powershell instance with Administrator privileges.
3. To pool, partition, and format the disks as a single striped volume, run the following command:

```
$PhysicalDisks = Get-PhysicalDisk | Where-Object {$_.CanPool -eq $true}
New-StoragePool -FriendlyName "DataFiles" -StorageSubsystemFriendlyName "Windows Storage*" `
-PhysicalDisks $PhysicalDisks | New-VirtualDisk -FriendlyName "Datafiles" `
-Interleave 65536 -NumberOfColumns $PhysicalDisks.Count -ResiliencySettingName simple `
-UseMaximumSize |Initialize-Disk -PartitionStyle GPT -PassThru |New-Partition -AssignDriveLetter `
-UseMaximumSize |Format-Volume -FileSystem NTFS -NewFileSystemLabel "DataDisks" `
-AllocationUnitSize 65536 -Confirm:$false
```

Optimizing the database

Amazon Web Services (AWS) provided us with a script that modified the base TPC-C-like database. The script altered some row and page locking settings and modified some stored procedures to improve OPM results for any solution running the benchmark.

Running the optimization script

1. Log into the SUT instance.
2. Open SQL Server Management Studio (SSMS).
3. Restore the tpcc database backup.
4. Run the tpcc_optimze.sql query (available at the end of this document).
5. Back up the optimized database.

Running the tests

In this section, we list the steps required to run the OLTP workload driven by a client instance against our instance under test. We use HammerDB's CLI and have included the scripts we used.

Running the TPC-C-like OLTP workload

1. Log into the client instance.
2. Navigate to the HammerDB folder, and edit load_parameters.tcl to point to the IP of your instance under test.
3. On the instance under test, start Perfmon.
4. On the client instance, open a command prompt, and navigate to the HammerDB folder.
5. Type the following command:

```
hammerdbcli.bat auto execture_test.tcl
```

6. When the run finishes, stop Perfmon, and save the HammerDB results file and Perfmon output.
7. Delete and restore the TPC-C database on the SQL VM.
8. Reboot the VMs.
9. Repeat the test two more times for a total of three runs, and record the median run.

execute_test.tcl

```
#!/usr/bin/tclsh
proc runtimer { seconds } {
set x 0
set timerstop 0
while {!$timerstop} {
 incr x
 after 1000
  if { ![ expr {$x % 60} ] } {
  set y [ expr $x / 60 ]
  puts "Timer: $y minutes elapsed"
  }
 update
 if {  [ vucomplete ] || $x eq $seconds } { set timerstop 1 }
```

```
    }
return
}
vuset logtotemp 1
source load_common_parameters.tcl
source load_parameters.tcl
loadscript
set intra_run_script ""
print dict
puts "*** TEST SEQUENCE STARTED ***"
foreach z { 144 } {
puts "***+ VU:$z"
if {$intra_run_script ne ""} {
  set log_time [clock seconds]
  puts "***+ tcl_timing +***{\"event_description\":\"Executing $intra_run_
    script\",\"timestamp\":$log_time}"
  if { [catch { exec bash $intra_run_script <<db_username>> <<db_password>> <<db_host>> <<db_name>>
    2>> "results/intra_run_script_output.txt" } msg] } {
    puts "Intra run script wrote to stdout, but that's OK"
  }
  set log_time [clock seconds]
  puts "***+ tcl_timing +***{\"event_description\":\"Execution of $intra_run_script is
    complete\",\"timestamp\":$log_time}"
}
set start_time [clock seconds]
puts "***+ Start Test:$start_time"
vuset vu $z
vuset showoutput 1
vuset delay 2000
vucreate
vurun
runtimer 1680
set end_time [clock seconds]
puts "***+ End Test:$end_time"
set elapsed [ expr { $end_time - $start_time } ]
puts "***+ tcl_timing +***{\"event_description\":\"Executed test VU$z\",\"timestamp\":$end_
    time,\"elapsed_time\":$elapsed}"
vudestroy
after 5000
}
puts "*** TEST SEQUENCE COMPLETE ***"
exit
```

load_common_parameters.tcl

```
#!/usr/bin/tclsh
puts "***+ Run Set ID:20200917211908"
puts "***+ Run ID:20200917211908-0"
puts "***+ Run Sequence:0"
puts "***+ Warehouse Count:1000"
puts "*** LOADING COMMON PARAMETERS ***"
dbset db mssqls
dbset bm TPC-C
diset tpcc mssqls_count_ware 1000
diset tpcc mssqls_rampup 3
diset tpcc mssqls_duration 20
diset tpcc mssqls_driver timed
diset tpcc mssqls_allwarehouse true
```

```
diset tpcc mssqls_raiseerror true
diset tpcc mssqls_timeprofile true
diset tpcc mssqls_total_iterations 2147483648
puts "*** COMMON PARAMETERS LOADED ***"
```

load_parameters.tcl

```
#!/usr/bin/tclsh
diset connection mssqls_linux_server [SQL VM IP]
diset connection mssqls_server [SQL VM IP]
diset connection mssqls_uid sa
diset connection mssqls_pass [password]
diset connection mssqls_authentication sql
```

tpcc_optimize.sql

```
USE [master]
GO
PRINT N'Changing database settings...';
GO
ALTER DATABASE [tpcc] SET AUTO_CREATE_STATISTICS OFF
GO
ALTER DATABASE [tpcc] SET AUTO_UPDATE_STATISTICS OFF WITH NO_WAIT
GO
ALTER DATABASE [tpcc] SET TARGET_RECOVERY_TIME = 300 SECONDS WITH NO_WAIT
GO
ALTER DATABASE [tpcc] SET DELAYED_DURABILITY = DISABLED WITH NO_WAIT
GO

USE [tpcc]
GO

PRINT N'Altering indexes on [dbo].[customer]...';
GO
DROP INDEX dbo.customer.customer_i2
GO
CREATE NONCLUSTERED INDEX [customer_c_last]
 ON [dbo].[customer]([c_w_id] ASC, [c_d_id] ASC, [c_last] ASC, [c_first] ASC, [c_id] ASC) WITH (ALLOW_
    PAGE_LOCKS = OFF);
 GO


PRINT N'Altering indexes on [dbo].[district]...';
GO
DROP INDEX [dbo].[district].[d_details]
GO
CREATE NONCLUSTERED INDEX [district_d_id]
 ON [dbo].[district]([d_id] ASC) WITH (ALLOW_PAGE_LOCKS = OFF);
 GO
ALTER INDEX [PK_DISTRICT] ON [dbo].[district] SET (ALLOW_PAGE_LOCKS = OFF)
GO

PRINT N'Altering indexes on [dbo].[history]...';
GO
CREATE CLUSTERED INDEX [history_i1]
 ON [dbo].[history]([h_c_w_id] ASC, [h_date] ASC, [h_c_d_id] ASC, [h_c_id] ASC, [h_amount] ASC);
GO
```

```
PRINT N’Altering indexes on [dbo].[item]...’;
GO
ALTER INDEX [PK_ITEM] ON [dbo].[item] SET (ALLOW_ROW_LOCKS = OFF)
GO

PRINT N’Altering indexes on [dbo].[new_order]...’;
GO
ALTER INDEX [new_order_i1] ON [dbo].[new_order] SET (ALLOW_ROW_LOCKS = OFF)
GO
ALTER INDEX [new_order_i1] ON [dbo].[new_order] SET (ALLOW_PAGE_LOCKS = ON)
GO

PRINT N’Altering indexes on [dbo].[order_line]...’;
GO
ALTER INDEX [order_line_i1] ON [dbo].[order_line] SET (ALLOW_ROW_LOCKS = OFF)
GO
ALTER INDEX [order_line_i1] ON [dbo].[order_line] SET (ALLOW_PAGE_LOCKS = ON)
GO

PRINT N’Altering indexes on [dbo].[orders]...’;
GO
ALTER INDEX [orders_i1] ON [dbo].[orders] SET (ALLOW_ROW_LOCKS = OFF)
GO
ALTER INDEX [orders_i1] ON [dbo].[orders] SET (ALLOW_PAGE_LOCKS = ON)
GO
ALTER INDEX [orders_i2] ON [dbo].[orders] SET (ALLOW_ROW_LOCKS = OFF)
GO
ALTER INDEX [orders_i2] ON [dbo].[orders] SET (ALLOW_PAGE_LOCKS = ON)
GO

PRINT N’Altering indexes on [dbo].[stock]...’;
GO
ALTER INDEX [PK_STOCK] ON [dbo].[stock] SET (ALLOW_PAGE_LOCKS = OFF)
GO

PRINT N’Altering indexes on [dbo].[warehouse]...’;
GO
ALTER INDEX [PK_WAREHOUSE] ON [dbo].[warehouse] SET (ALLOW_PAGE_LOCKS = OFF)
GO
DROP INDEX [dbo].[warehouse].[w_details]
GO

GO
PRINT N’Altering SqlProcedure [dbo].[ostat]...’;


GO



ALTER PROCEDURE [dbo].[ostat]
@os_w_id int,
@os_d_id int,
@os_c_id int,
@byname int,
@os_c_last char(20)
```

```
AS
BEGIN
SET ANSI_WARNINGS OFF
DECLARE
@os_c_first char(16),
@os_c_middle char(2),
@os_c_balance money,
@os_o_id int,
@os_entdate datetime2(0),
@os_o_carrier_id int,
@os_ol_i_id        INT,
@os_ol_supply_w_id INT,
@os_ol_quantity INT,
@os_ol_amount      INT,
@os_ol_delivery_d DATE,
@namecnt int,
@i int
BEGIN TRANSACTION
BEGIN TRY

IF (@byname = 1)
BEGIN
  SELECT TOP 1
    @os_c_id = c_id,
    @os_c_balance = c_balance,
    @os_c_first = c_first,
    @os_c_middle = c_middle,
    @os_c_last = c_last
  FROM (
    SELECT TOP 50 PERCENT c_id, c_balance, c_first, c_middle, c_last
    FROM dbo.customer WITH (repeatableread)
    WHERE
      c_last = @os_c_last AND
      c_w_id = @os_w_id AND
      c_d_id = @os_d_id
    ORDER BY c_first) X
  ORDER BY c_first desc
END

ELSE
BEGIN
  SELECT @os_c_balance = customer.c_balance, @os_c_first = customer.c_first
  , @os_c_middle = customer.c_middle, @os_c_last = customer.c_last
  FROM dbo.customer WITH (repeatableread)
  WHERE customer.c_id = @os_c_id AND customer.c_d_id = @os_d_id AND customer.c_w_id = @os_w_id
END


BEGIN
SELECT TOP (1) @os_o_id = o_id, @os_o_carrier_id = o_carrier_id, @os_entdate = o_entry_d
FROM dbo.orders WITH (serializable)
WHERE orders.o_d_id = @os_d_id
AND orders.o_w_id = @os_w_id
AND orders.o_c_id = @os_c_id
ORDER BY orders.o_id DESC

IF @@ROWCOUNT = 0
```

```
PRINT 'No orders for customer';
END

SELECT order_line.ol_i_id
, order_line.ol_supply_w_id
, order_line.ol_quantity
, order_line.ol_amount
, order_line.ol_delivery_d
FROM dbo.order_line WITH (repeatableread)
WHERE order_line.ol_o_id = @os_o_id
AND order_line.ol_d_id = @os_d_id
AND order_line.ol_w_id = @os_w_id

SELECT@os_c_id as N'@os_c_id', @os_c_last as N'@os_c_last', @os_c_first as N'@os_c_first', @os_c_middle
    as N'@os_c_middle', @os_c_balance as N'@os_c_balance', @os_o_id as N'@os_o_id', @os_entdate as
    N'@os_entdate', @os_o_carrier_id as N'@os_o_carrier_id'
END TRY
BEGIN CATCH
SELECT
ERROR_NUMBER() AS ErrorNumber
,ERROR_SEVERITY() AS ErrorSeverity
,ERROR_STATE() AS ErrorState
,ERROR_PROCEDURE() AS ErrorProcedure
,ERROR_LINE() AS ErrorLine
,ERROR_MESSAGE() AS ErrorMessage;
IF @@TRANCOUNT > 0
ROLLBACK TRANSACTION;
END CATCH;
IF @@TRANCOUNT > 0
COMMIT TRANSACTION;
END
GO
PRINT N'Altering SqlProcedure [dbo].[delivery]...';


GO


ALTER PROCEDURE [dbo].[delivery]
@d_w_id int,
@d_o_carrier_id int,
@timestamp datetime2(0)
AS
BEGIN
SET ANSI_WARNINGS OFF
DECLARE
@d_no_o_id int,
@d_d_id int,
@d_c_id int,
@d_ol_total int
BEGIN TRANSACTION
BEGIN TRY
DECLARE
@loop_counter int
SET @loop_counter = 1
WHILE @loop_counter <= 10
BEGIN
```

```
SET @d_d_id = @loop_counter


SELECT TOP 1
@d_no_o_id = no_o_id
FROM dbo.new_order WITH (serializable updlock)
WHERE no_w_id = @d_w_id AND
no_d_id = @d_d_id
ORDER BY no_o_id ASC

IF (@@rowcount <> 0)
BEGIN
    -- claim the order for this district
    DELETE dbo.new_order
    WHERE no_w_id = @d_w_id AND
    no_d_id = @d_d_id AND
    no_o_id = @d_no_o_id

    UPDATE dbo.orders
    SET o_carrier_id = @d_o_carrier_id
    , @d_c_id = orders.o_c_id
    WHERE orders.o_id = @d_no_o_id
    AND orders.o_d_id = @d_d_id
    AND orders.o_w_id = @d_w_id

    SET @d_ol_total = 0

    UPDATE dbo.order_line
    SET ol_delivery_d = @timestamp
        , @d_ol_total = @d_ol_total + ol_amount
    WHERE order_line.ol_o_id = @d_no_o_id
    AND order_line.ol_d_id = @d_d_id
    AND order_line.ol_w_id = @d_w_id

    UPDATE dbo.customer SET c_balance = customer.c_balance + @d_ol_total
    WHERE customer.c_id = @d_c_id
    AND customer.c_d_id = @d_d_id
    AND customer.c_w_id = @d_w_id
END



PRINT
'D: '
+
ISNULL(CAST(@d_d_id AS nvarchar(4000)), '')
+
'O: '
+
ISNULL(CAST(@d_no_o_id AS nvarchar(4000)), '')
+
'time '
+
ISNULL(CAST(@timestamp AS nvarchar(4000)), '')
SET @loop_counter = @loop_counter + 1
END
```

```
SELECT@d_w_id as N'@d_w_id', @d_o_carrier_id as N'@d_o_carrier_id', @timestamp as N'@TIMESTAMP'
END TRY
BEGIN CATCH
SELECT
ERROR_NUMBER() AS ErrorNumber
,ERROR_SEVERITY() AS ErrorSeverity
,ERROR_STATE() AS ErrorState
,ERROR_PROCEDURE() AS ErrorProcedure
,ERROR_LINE() AS ErrorLine
,ERROR_MESSAGE() AS ErrorMessage;
IF @@TRANCOUNT > 0
ROLLBACK TRANSACTION;
END CATCH;
IF @@TRANCOUNT > 0
COMMIT TRANSACTION;
END
GO
PRINT N'Altering SqlProcedure [dbo].[payment]...';


GO

ALTER PROCEDURE [dbo].[payment]
@p_w_id int,
@p_d_id int,
@p_c_w_id int,
@p_c_d_id int,
@p_c_id int,
@byname int,
@p_h_amount numeric(6,2),
@p_c_last char(16),
@TIMESTAMP datetime2(0)
AS
BEGIN
SET ANSI_WARNINGS OFF
DECLARE
@p_w_street_1 char(20),
@p_w_street_2 char(20),
@p_w_city char(20),
@p_w_state char(2),
@p_w_zip char(10),
@p_d_street_1 char(20),
@p_d_street_2 char(20),
@p_d_city char(20),
@p_d_state char(20),
@p_d_zip char(10),
@p_c_first char(16),
@p_c_middle char(2),
@p_c_street_1 char(20),
@p_c_street_2 char(20),
@p_c_city char(20),
@p_c_state char(20),
@p_c_zip char(9),
@p_c_phone char(16),
@p_c_since datetime2(0),
@p_c_credit char(32),
```

```
@p_c_credit_lim  numeric(12,2),
@p_c_discount  numeric(4,4),
@p_c_balance money,       --numeric(12,2),
@p_c_data varchar(500),
@namecnt int,
@p_d_name char(11),
@p_w_name char(11),
@p_c_new_data varchar(500),
@h_data varchar(30)

BEGIN TRY

IF (@byname = 1)
BEGIN
SELECT TOP 1
  @p_c_id = c_id
FROM (
  SELECT TOP 50 PERCENT c_id, c_first
  FROM dbo.customer WITH (repeatableread)
  WHERE
    c_last = @p_c_last AND
    c_w_id = @p_c_w_id AND
    c_d_id = @p_c_d_id
  ORDER BY c_first) X
ORDER BY c_first desc
END


BEGIN TRANSACTION

-- get customer info and update balances
UPDATE dbo.customer
SET
  @p_c_balance = c_balance = c_balance - @p_h_amount,
  c_data =
    CASE
      WHEN c_credit <> 'BC' THEN c_credit
      ELSE LEFT(
           ISNULL(CAST(@p_c_id AS char), '') +   ' '     +
           ISNULL(CAST(@p_c_d_id AS char), '') + ' ' +
           ISNULL(CAST(@p_c_w_id AS char), '') + ' ' +
           ISNULL(CAST(@p_d_id AS char), '') + ' ' +
           ISNULL(CAST(@p_w_id AS char), '') + ' ' +
           ISNULL(CAST(@p_h_amount AS CHAR(8)), '') + ' ' +
           ISNULL(CAST(@TIMESTAMP AS char), '') + ' ' +
           ISNULL(@p_w_name, '') + ' ' +
           ISNULL(@p_d_name, '') + ' ' +
           c_data,
           500)
    END,
  @p_c_first = c_first,
  @p_c_middle = c_middle,
  @p_c_last = c_last,
  @p_c_street_1 = c_street_1,
  @p_c_street_2 = c_street_2,
  @p_c_city = c_city,
```

```
  @p_c_state = c_state,
  @p_c_zip = c_zip,
  @p_c_phone = c_phone,
  @p_c_credit = c_credit,
  @p_c_credit_lim = c_credit_lim,
  @p_c_discount = c_discount,
  @p_c_since = c_since
WHERE
  c_id = @p_c_id AND
  c_w_id = @p_c_w_id AND
  c_d_id = @p_c_d_id

SET @h_data = (ISNULL(@p_w_name, '') + '    ' + ISNULL(@p_d_name, ''))

INSERT dbo.history( h_c_d_id, h_c_w_id, h_c_id, h_d_id, h_w_id, h_date, h_amount, h_data)
VALUES ( @p_c_d_id, @p_c_w_id, @p_c_id, @p_d_id, @p_w_id, @TIMESTAMP, @p_h_amount, @h_data)

-- get district data and update year-to-date
UPDATE dbo.district
SET
  d_ytd = d_ytd + @p_h_amount,
  @p_d_street_1 = d_street_1,
  @p_d_street_2 = d_street_2,
  @p_d_city = d_city,
  @p_d_state = d_state,
  @p_d_zip = d_zip,
  @p_d_name = d_name
WHERE
  d_w_id = @p_w_id AND
  d_id = @p_d_id

-- get warehouse data and update year-to-date
UPDATE dbo.warehouse
SET
  w_ytd = w_ytd + @p_h_amount,
  @p_w_street_1 = w_street_1,
  @p_w_street_2 = w_street_2,
  @p_w_city = w_city,
  @p_w_state = w_state,
  @p_w_zip = w_zip,
  @p_w_name = w_name
WHERE
  w_id = @p_w_id

SELECT@p_c_id as N'@p_c_id', @p_c_last as N'@p_c_last', @p_w_street_1 as N'@p_w_street_1'
, @p_w_street_2 as N'@p_w_street_2', @p_w_city as N'@p_w_city'
, @p_w_state as N'@p_w_state', @p_w_zip as N'@p_w_zip'
, @p_d_street_1 as N'@p_d_street_1', @p_d_street_2 as N'@p_d_street_2'
, @p_d_city as N'@p_d_city', @p_d_state as N'@p_d_state'
, @p_d_zip as N'@p_d_zip', @p_c_first as N'@p_c_first'
, @p_c_middle as N'@p_c_middle', @p_c_street_1 as N'@p_c_street_1'
, @p_c_street_2 as N'@p_c_street_2'
, @p_c_city as N'@p_c_city', @p_c_state as N'@p_c_state', @p_c_zip as N'@p_c_zip'
, @p_c_phone as N'@p_c_phone', @p_c_since as N'@p_c_since', @p_c_credit as N'@p_c_credit'
, @p_c_credit_lim as N'@p_c_credit_lim', @p_c_discount as N'@p_c_discount', @p_c_balance as
    N'@p_c_balance'
```

```
, @p_c_data as N'@p_c_data'


END TRY
BEGIN CATCH
SELECT
ERROR_NUMBER() AS ErrorNumber
,ERROR_SEVERITY() AS ErrorSeverity
,ERROR_STATE() AS ErrorState
,ERROR_PROCEDURE() AS ErrorProcedure
,ERROR_LINE() AS ErrorLine
,ERROR_MESSAGE() AS ErrorMessage;
IF @@TRANCOUNT > 0
ROLLBACK TRANSACTION;
END CATCH;
IF @@TRANCOUNT > 0
COMMIT TRANSACTION;
END
GO
PRINT N'Altering SqlProcedure [dbo].[neword]...';


GO

ALTER PROCEDURE [dbo].[neword]
@no_w_id int,
@no_max_w_id int,
@no_d_id int,
@no_c_id int,
@no_o_ol_cnt int,
@TIMESTAMP datetime2(0)
AS
BEGIN
SET ANSI_WARNINGS OFF
DECLARE
@no_c_discount smallmoney,
@no_c_last char(16),
@no_c_credit char(2),
@no_d_tax smallmoney,
@no_w_tax smallmoney,
@no_d_next_o_id int,
@no_ol_supply_w_id int,
@no_ol_i_id int,
@no_ol_quantity int,
@no_o_all_local int,
@o_id int,
@no_i_name char(24),
@no_i_price smallmoney,
@no_i_data char(50),
@no_s_quantity int,
@no_ol_amount int,
@no_s_dist_01 char(24),
@no_s_dist_02 char(24),
@no_s_dist_03 char(24),
@no_s_dist_04 char(24),
@no_s_dist_05 char(24),
@no_s_dist_06 char(24),
```

```
@no_s_dist_07 char(24),
@no_s_dist_08 char(24),
@no_s_dist_09 char(24),
@no_s_dist_10 char(24),
@no_ol_dist_info char(24),
@no_s_data char(50),
@x int,
@rbk int


BEGIN TRANSACTION
BEGIN TRY

SET @no_o_all_local = 0

SELECT
  @no_c_discount = c_discount,
  @no_c_last = c_last,
  @no_c_credit = c_credit
FROM dbo.customer
WHERE
  c_w_id = @no_w_id AND
  c_d_id = @no_d_id AND
  c_id = @no_c_id

UPDATE dbo.district
SET
  @no_d_tax = d_tax,
  @o_id = d_next_o_id,
  @no_d_next_o_id = d_next_o_id = district.d_next_o_id + 1
WHERE district.d_id = @no_d_id
AND district.d_w_id = @no_w_id


SET @rbk = CAST(100 * RAND() + 1 AS INT)
DECLARE
@loop_counter int
SET @loop_counter = 1
DECLARE
@loop$bound int
SET @loop$bound = @no_o_ol_cnt
WHILE @loop_counter <= @loop$bound
BEGIN
IF ((@loop_counter = @no_o_ol_cnt) AND (@rbk = 1))
SET @no_ol_i_id = 100001
ELSE
SET @no_ol_i_id =  CAST(1000000 * RAND() + 1 AS INT)
SET @x = CAST(100 * RAND() + 1 AS INT)
IF (@x > 1)
SET @no_ol_supply_w_id = @no_w_id
ELSE
BEGIN
SET @no_ol_supply_w_id = @no_w_id
SET @no_o_all_local = 0
WHILE ((@no_ol_supply_w_id = @no_w_id) AND (@no_max_w_id != 1))
BEGIN
```

```
SET @no_ol_supply_w_id = CAST(@no_max_w_id * RAND() + 1 AS INT)
DECLARE
@db_null_statement$2 int
END
END
SET @no_ol_quantity = CAST(10 * RAND() + 1 AS INT)

SELECT @no_i_price = item.i_price
, @no_i_name = item.i_name
, @no_i_data = item.i_data
FROM dbo.item
WHERE item.i_id = @no_ol_i_id

UPDATE dbo.stock
SET
  s_quantity = s_quantity - @no_ol_quantity + CASE WHEN (s_quantity > @no_ol_quantity)
    THEN 0 ELSE 91 END,
  @no_s_data = s_data,
  @no_ol_dist_info =
    CASE @no_d_id
      WHEN 1 THEN s_dist_01
      WHEN 2 THEN s_dist_02
      WHEN 3 THEN s_dist_03
      WHEN 4 THEN s_dist_04
      WHEN 5 THEN s_dist_05
      WHEN 6 THEN s_dist_06
      WHEN 7 THEN s_dist_07
      WHEN 8 THEN s_dist_08
      WHEN 9 THEN s_dist_09
      WHEN 10 THEN s_dist_10
    END
OUTPUT
  @o_id,
  @no_d_id,
  @no_w_id,
  @loop_counter,
  @no_ol_i_id,
  NULL,
  (@no_ol_quantity * @no_i_price),
  @no_ol_supply_w_id,
  @no_ol_quantity,
  CASE @no_d_id
      WHEN 1 THEN inserted.s_dist_01
      WHEN 2 THEN inserted.s_dist_02
      WHEN 3 THEN inserted.s_dist_03
      WHEN 4 THEN inserted.s_dist_04
      WHEN 5 THEN inserted.s_dist_05
      WHEN 6 THEN inserted.s_dist_06
      WHEN 7 THEN inserted.s_dist_07
      WHEN 8 THEN inserted.s_dist_08
      WHEN 9 THEN inserted.s_dist_09
      WHEN 10 THEN inserted.s_dist_10
    END
INTO dbo.order_line
WHERE
  stock.s_i_id = @no_ol_i_id AND
  stock.s_w_id = @no_ol_supply_w_id
```

```
SET @loop_counter = @loop_counter + 1
END

INSERT dbo.orders( o_id, o_d_id, o_w_id, o_c_id, o_entry_d, o_ol_cnt, o_all_local)
VALUES ( @o_id, @no_d_id, @no_w_id, @no_c_id, @TIMESTAMP, @no_o_ol_cnt, @no_o_all_local)

INSERT dbo.new_order(no_o_id, no_d_id, no_w_id)
VALUES (@o_id, @no_d_id, @no_w_id)

IF (@rbk = 1)
ROLLBACK TRANSACTION

SELECT @no_w_tax = warehouse.w_tax
FROM dbo.warehouse
WHERE warehouse.w_id = @no_w_id

SELECT convert(char(8), @no_c_discount) as N'@no_c_discount', @no_c_last as N'@no_c_last', @no_c_
    credit as N'@no_c_credit', convert(char(8),@no_d_tax) as N'@no_d_tax', convert(char(8),@no_w_tax)
    as N'@no_w_tax', @no_d_next_o_id as N'@no_d_next_o_id'

END TRY
BEGIN CATCH
SELECT
ERROR_NUMBER() AS ErrorNumber
,ERROR_SEVERITY() AS ErrorSeverity
,ERROR_STATE() AS ErrorState
,ERROR_PROCEDURE() AS ErrorProcedure
,ERROR_LINE() AS ErrorLine
,ERROR_MESSAGE() AS ErrorMessage;
IF @@TRANCOUNT > 0
ROLLBACK TRANSACTION;
END CATCH;
IF @@TRANCOUNT > 0
COMMIT TRANSACTION;

END
GO
PRINT N'Altering SqlProcedure [dbo].[slev]...';


GO

ALTER PROCEDURE [dbo].[slev]
@st_w_id int,
@st_d_id int,
@threshold int
AS
BEGIN
DECLARE
@st_o_id_low int,
@st_o_id_high int
BEGIN TRANSACTION
BEGIN TRY

SELECT
@st_o_id_low = district.d_next_o_id - 20,
```

```
@st_o_id_high = district.d_next_o_id - 1
FROM dbo.district
WHERE district.d_w_id = @st_w_id AND district.d_id = @st_d_id

SELECT
COUNT(DISTINCT stock.s_i_id)
FROM dbo.order_line
, dbo.stock
WHERE order_line.ol_w_id = @st_w_id
AND order_line.ol_d_id = @st_d_id
AND order_line.ol_o_id BETWEEN @st_o_id_low AND @st_o_id_high
AND stock.s_w_id = order_line.ol_w_id
AND stock.s_i_id = order_line.ol_i_id
AND stock.s_quantity < @threshold
OPTION (ORDER GROUP)

END TRY
BEGIN CATCH
SELECT
ERROR_NUMBER() AS ErrorNumber
,ERROR_SEVERITY() AS ErrorSeverity
,ERROR_STATE() AS ErrorState
,ERROR_PROCEDURE() AS ErrorProcedure
,ERROR_LINE() AS ErrorLine
,ERROR_MESSAGE() AS ErrorMessage;
IF @@TRANCOUNT > 0
ROLLBACK TRANSACTION;
END CATCH;
IF @@TRANCOUNT > 0
COMMIT TRANSACTION;
END
GO
PRINT N'Update complete.';


GO
```

How we tested the Azure VM

Creating a database backup

This section contains steps for creating a TPC-C-like database, generating data and loading the data onto the database, and making a backup of the database.

Creating the TPC-C-like database

1. Open SQL Server Management Studio.
2. Right-click Databases, and choose New Database…
3. Name the database.
4. Click the Filegroups tab, and click Add Filegroup.
5. Name the new filegroup, and set it to Default.
6. In the general tab, to create one additional data file, click Add.
7. Name each file. Set the initial size for the new file to 200GB. Make sure the file is on the new Filegroup you created.
8. If necessary, change all data and log file paths to point to the data and log drives on the system.
9. On the Options tab, set the Recovery Model to simple.
10. To start the build, click OK. This may take a while.

Generating the data and loading It into the database

1. Open HammerDB.
2. Select Options→Benchmark.
3. Choose MSSQL Server and TPC-C.
4. Expand SQl Server→TPC-C→Schema Build.
5. Double-click Options, and perform the following actions:
6. Enter the IP of the SQL Server.
7. Change the driver to ODBC Driver 17 for SQL Server.
8. Enter the SQL Server Database name.
9. Set Number of Warehouses to 1000.
10. Select an appropriate number of Virtual Users to Build Schema.
11. Click OK.
12. To begin the build, double-click Build.

Backing up the database

1. Open Microsoft SQL Management Studio, and right-click the TPC-C database.
2. Navigate to Tasks→Backup.
3. On the General tab, to set the name of your backup file and the location to store the file, click Add….
4. In the left-hand pane, click Backup Options.
5. Under Compression, select Compress backup.
6. Click Ok.

Creating the Microsoft Windows Server 2019 baseline image

Creating the baseline image VM

1. Log into the Azure Portal, and navigate to the Virtual Machines service.
2. To open the Add VM wizard, click Add.
3. On the Basics tab, set the following:
4. Choose your Subscription from the dropdown menu.
5. Choose your Resource group from the dropdown menu.
6. Name the Virtual Machine.
7. Choose your Region from the dropdown menu.
8. Leave the Availability options set to No infrastructure redundancy required.
9. Choose Windows Server 2019 Datacenter from the Image dropdown menu.
10. Leave Azure Spot instance set to No.
11. Select the instance size you wish to use. We used Standard B4ms.

12. Choose a new username and password for the Administrator account.
13. Leave Public inbound ports set to Allow selected ports.
14. For Select inbound ports, choose SSH (22).
15. On the Disks tab, set the following:
16. For the OS disk type, choose Standard HDD from the dropdown menu.
17. Leave the default Encryption type.
18. On the Networking tab, set the following:
19. Choose your Virtual network from the dropdown menu.
20. To create a new Public IP, choose Create new.
21. Leave the rest of the settings at defaults.
22. On the Management tab, set the following:
23. Choose your Diagnostics storage account from the dropdown menu.
24. Leave the rest set to defaults.
25. On the Advanced tab, leave all defaults.
26. On the Tags tab, add any tags you wish to use.
27. On the Review + create tab, review your settings, and click Create.

Configuring Windows Server 2019

1. Open Server Manager, and click on Local Server.
2. Disable IE Enhanced Security Configuration.
3. Change the time zone to your local time zone.
4. Change the name of your server, and reboot when prompted.
5. Open Server Manager again, and click on Local Server.
6. Click to run updates.
7. Run updates, rebooting when prompted, until the server shows no new updates to install.

Creating a snapshot of your baseline client VM

1. In your Azure portal, navigate to the Snapshots service.
2. To open the Snapshot wizard, click Add.
3. On the Basics tab, set the following:
    a. Choose your Subscription from the dropdown menu.
    b. Choose your Resource group from the dropdown menu.
    c. Enter a name for your snapshot.
    d. Choose your Region from the dropdown menu.
    e. Select Full - make a complete read-only copy of the selected disk for the Snapshot type.
    f. Choose the OS disk from your baseline VM.
    g. Choose Standard SSD for the Storage type.
4. On the Encryption tab, leave all defaults.
5. On the Tags tab, add any tags you wish to use.
6. On the Review + create tab, review your settings, and click Create.

Installing Microsoft SQL Server 2019 Enterprise

1. Download or copy the ISO to the server, and unzip it.
2. Double-click the Setup application.
3. Click Installation→New SQL Server Standalone installation or add features to an existing installation.
4. Choose the trial version, and click Next.
5. Check the I accept the license terms and Privacy Statement box, and click Next.
6. Check the Use Microsoft Update to check for updates (recommended) box, and click Next.
7. On the Install Rules page, click Next.

8. Check the boxes for the following features, and click Next:
    a. Database Engine Services
    b. Full-Test and Semantic Extractions for Search
    c. Client Tools Connectivity
    d. Client Tools Backwards Compatibility
9. Leave the Default instance, and click Next.
10. Leave the default Service Accounts, and click Next.
11. On the Server Configuration tab, choose Mixed Mode and enter and confirm a Password for the SQL Server system administrator (sa) account.
12. Click Add Current User to Specify the SQL Server administrators.
13. Click Next.
14. Once you've passed the rule check, click Next.
15. Click Install.
16. When the installation is complete, go back to the SQL Server Installation Center, and click Install SQL Server Management Tools.
17. Download the SSMS file, and install with defaults.
18. When prompted, reboot the server.
19. Run Windows Update one more time to ensure there aren't any new updates for SQL (make sure Windows Updates are set to get updates for other Microsoft products).
20. Once you've installed all available updates, disable Windows Update service:
21. Click Start.
22. To open the Services list, type `services`
23. Disable the Windows Update service.

Locking pages in memory

1. Click Start, and type `Local Security Policy`
2. Open the Security program.Expand Local Policies, and click User Rights Assignment.
3. In the right-hand pane, scroll down and double-click Lock pages in memory.
4. Click Add User or Group, type `NT Service\MSSQLSERVER`, and click OK.
5. To close the Properties window, click OK. Close the Local Security Policy window.

Installing HammerDB 3.3

1. Download HammerDB from https://hammerdb.com/download.html.
2. Double-click the .exe file, choose English, and click OK.
3. Click Yes.
4. Click Next.
5. Chose a destination location, and click Next.
6. Click Next.
7. Click Finish.

Creating a snapshot of your baseline SQL Server VM

1. In the Azure portal, navigate to the Snapshots service.
2. To open the Snapshot wizard, click Add.On the Basics tab, set the following:
    a. Choose your Subscription from the dropdown menu.
    b. Choose your Resource group from the dropdown menu.
    c. Enter a name for your snapshot.
    d. Choose your Region from the dropdown menu.
    e. For the Snapshot type, celect Full - make a complete read-only copy of the selected disk.
    f. Choose the OS disk from your baseline VM.
    g. For the Storage type, choose Standard SSD.On the Encryption tab, leave all defaults.
3. On the Tags tab, add any tags you wish to use.
4. On the Review + create tab, review your settings, and click Create.

Creating images with the baseline snapshots

To create an image, you must first have a Shared Image Gallery. The steps below will walk you through the creation of the gallery as well as the image creation steps. Once you have created your gallery, you will not need to do so again to add new images.

1. In the Azure portal, navigate to the Shared image galleries service.
2. To open the Add gallery wizard, click Add.On the Basics tab, set the following:
    a. Choose your Subscription from the dropdown menu.
    b. Choose your Resource from the dropdown menu.
    c. Name your gallery.
    d. Choose your Region from the dropdown menu.
    e. Enter a Description, if you'd like.
3. On the Tags tab, add any tags you wish to use.
4. On the Review + create tab, review your settings, and click Create.
5. Click on your new image gallery. To open the wizard, click Add new image definition.
6. On the Basics tab, set the following:
    a. Set the Operating System to Windows.
    b. Set the Operation system state to Specialized.
    c. Enter whatever you wish for the Publisher, Offer, and SKU entries.
7. Skip the Version tab.
8. Skip the Publishing options tab.
9. On the Tags tab, add any tags you wish to use.
10. On the Review + create tab, review your settings, and click Create.
11. Click on the image definition you've created, and click Add version to open the wizard.
12. On the Basics tab, set the following:
    a. Enter a version number, such as `1.0.0`Choose the OS disk snapshot of the baseline client VM you created from the dropdown menu.
    b. Leave the rest as defaults.
13. On the Encryption tab, leave defaults.
14. On the Tags tab, add any tags you wish to use.
15. On the Review + create tab, review your settings, and click Create.
16. Repeat steps 5 through 15 for the SQL Server VM, using the OS disk snapshot of the baseline SQL VM in step 12.

Creating the VMs under test

In this section we list the steps required to create a VM from the image we created previously.

Creating VMs from the specialized Images

1. Open the Azure Portal, and navigate to the Shared image galleries service.
2. Click on the Shared image gallery you created.
3. Navigate to the image version you created (we used 1.0.0), and click Create VM.
4. On the Basics tab, set the following:
    a. Choose your Subscription from the dropdown menu.
    b. Choose your Resource group from the dropdown menu.
    c. Enter a Virtual machine name.
    d. Choose Availability Zone and set the Zone you desire.
    e. Select the instance size.
    f. Under Licensing, select Windows server.
    g. Leave the rest as defaults.
5. On the Disks tab, set the following:
    a. Change the OS disk type to Standard HDD.
    b. Click Create and attach a new disk.
        i. In the Create a new disk wizard, click Change size, and pick the size of Premium SSD that matches your instance type.
        ii. Leave the rest as defaults, and click OK.

6. Skip the Networking, Management, and Advanced tabs.
7. On the Tags tab, assign any tags you wish to use.
8. On the Review + create tab, review your settings, and click Create.
9. Once the VM creation is finished, click Go to resource (or, navigate to the virtual machine service and, click the new VM).
10. Click ConnectàRDP, and download the RDP file.
11. Double-click the RDP file, and log in with the user and password you set previously.
12. Right-click the Windows Start button, and click Disk Management.
13. On the GPT partition popup window, click OK.
14. Right-click the Premium SSD you added, and follow the prompts to create a new NTFS volume for the database.

Configuring SQL Server on the VMs under test

In this section, we list the various SQL Server settings that we changed and the steps for changing them.

Setting the SQL Server memory reserve and max degree of parallelism (MAXDOP)

1. Open the SQL Server Management Studio.
2. Right-click on the SQL Instance, and click Properties.
3. Click Advanced node. Scroll down to Max Degree of Parallelism, and change the value to 1Click OK.
4. Right-click the SQL Instance, and go to Memory.
5. Set the Max Memory to 90% of the total memory in the system.
6. Click OK, and close the Properties window.
7. Right-click the SQL Instance, and restart the service.
8. When prompted, click Yes.

Configuring the tempdb database

1. Open the SQL Server Management Studio
2. Expand Databases and System databases, and right-click tempdb.
3. Add files, and change the starting size as necessary. We used 8x 1GB files.
4. Right-click the SQL Server instance, and restart the service.
5. When prompted, click Yes.

Creating, attaching, and formatting data disks

Creating and attaching data disks

1. Open the Azure Portal, and navigate to Virtual machines.
2. Select your VM under test, and click Disks.
3. Under Data disks, click Create and attach a new disk.
4. Name the disk.
5. Leave the Storage type as Premium SSD.
6. Set the Size to 8192 GiB.
7. Leave default Encryption (SSE with PMK) and Host caching (None).
8. Repeat steps 3 through 7 until you have 6 data disks.
9. Click Save.

Partitioning and formatting the drives

1. Log into the VM under test.
2. Open a Powershell instance with Administrator privileges.
3. Run the following command to pool, partition, and format the disks as a single striped volume:

```
$PhysicalDisks = Get-PhysicalDisk | Where-Object {$_.CanPool -eq $true}
New-StoragePool -FriendlyName "DataFiles" -StorageSubsystemFriendlyName "Windows Storage*" `
-PhysicalDisks $PhysicalDisks | New-VirtualDisk -FriendlyName "Datafiles" `
-Interleave 65536 -NumberOfColumns $PhysicalDisks.Count -ResiliencySettingName simple `
-UseMaximumSize |Initialize-Disk -PartitionStyle GPT -PassThru |New-Partition -AssignDriveLetter `
-UseMaximumSize |Format-Volume -FileSystem NTFS -NewFileSystemLabel "DataDisks" `
-AllocationUnitSize 65536 -Confirm:$false
```

Optimizing the database

AWS provided us with a script that modified the base TPC-C-like database. The script altered certain row and page locking settings, and modified certain stored procedures with the aim of improving OPM results for any solution running the benchmark.

Running the optimization script

1. Log into the SUT instance.
2. Open SQL Server Management Studio (SSMS).
3. Restore the tpcc database backup.
4. Run the tpcc_optimze.sql query (available at the end of this document).
5. Back up the optimized database.

Running the tests

In this section, we list the steps required to run the OLTP workload driven by a client VM against our in VM under test. We use HammerDB's CLI, and include the scripts to be used below. We tested on a 1,000 warehouse

Running the TPC-C-like OLTP workload

1. Log into the client instance.
2. Navigate to the HammerDB folder, and edit load_parameters.tcl to point to the IP of your VM under test.
3. On the instance under test, start Perfmon.
4. On the client instance, open a command prompt, and navigate to the HammerDB folder.
5. Type the following command:

```
hammerdbcli.bat auto execture_test.tcl
```

6. When the run finishes, stop Perfmon, and save the HammerDB results file and Perfmon output.
7. On the SQL Server VM, delete and restore the TPC-C database.
8. Reboot the VMs.
9. Repeat the test two more times for a total of three runs, and record the median run.

execute_test.tcl

```
#!/usr/bin/tclsh

proc runtimer { seconds } {

set x 0

set timerstop 0

while {!$timerstop} {

 incr x

 after 1000

  if { ![ expr {$x % 60} ] } {

  set y [ expr $x / 60 ]

  puts "Timer: $y minutes elapsed"

  }

 update

 if {  [ vucomplete ] || $x eq $seconds } { set timerstop 1 }

    }

return
```

```
}

vuset logtotemp 1

source load_common_parameters.tcl

source load_parameters.tcl

loadscript

set intra_run_script ""

print dict

puts "*** TEST SEQUENCE STARTED ***"

foreach z { 144 } {

puts "***+ VU:$z"

if {$intra_run_script ne ""} {

  set log_time [clock seconds]

  puts "***+ tcl_timing +***{\"event_description\":\"Executing $intra_run_
script\",\"timestamp\":$log_time}"

       if { [catch { exec bash $intra_run_script <<db_username>> <<db_password>> <<db_host>> <<db_name>>
2>> "results/intra_run_script_output.txt" } msg] } {

    puts "Intra run script wrote to stdout, but that's OK"

  }

  set log_time [clock seconds]

  puts "***+ tcl_timing +***{\"event_description\":\"Execution of $intra_run_script is
complete\",\"timestamp\":$log_time}"

}

set start_time [clock seconds]

puts "***+ Start Test:$start_time"

vuset vu $z

vuset showoutput 1

vuset delay 2000

vucreate

vurun

runtimer 1680

set end_time [clock seconds]

puts "***+ End Test:$end_time"

set elapsed [ expr { $end_time - $start_time } ]

puts "***+ tcl_timing +***{\"event_description\":\"Executed test VU$z\",\"timestamp\":$end_
time,\"elapsed_time\":$elapsed}"

vudestroy

after 5000
```

```
}

puts "*** TEST SEQUENCE COMPLETE ***"

exit
```

load_common_parametes.tcl

```
#!/usr/bin/tclsh

puts "***+ Run Set ID:20200917211908"

puts "***+ Run ID:20200917211908-0"

puts "***+ Run Sequence:0"

puts "***+ Warehouse Count:1000"

puts "*** LOADING COMMON PARAMETERS ***"

dbset db mssqls

dbset bm TPC-C

diset tpcc mssqls_count_ware 1000

diset tpcc mssqls_rampup 3

diset tpcc mssqls_duration 20

diset tpcc mssqls_driver timed

diset tpcc mssqls_allwarehouse true

diset tpcc mssqls_raiseerror true

diset tpcc mssqls_timeprofile true

diset tpcc mssqls_total_iterations 2147483648

puts "*** COMMON PARAMETERS LOADED ***"
```

load_parameters.tcl

```
#!/usr/bin/tclsh
diset connection mssqls_linux_server [SQL VM IP]
diset connection mssqls_server [SQL VM IP]
diset connection mssqls_uid sa
diset connection mssqls_pass [password]
diset connection mssqls_authentication sql
```

tpcc_optimize.sql

```
USE [master]
GO
PRINT N'Changing database settings...';
GO
ALTER DATABASE [tpcc] SET AUTO_CREATE_STATISTICS OFF
GO
ALTER DATABASE [tpcc] SET AUTO_UPDATE_STATISTICS OFF WITH NO_WAIT
GO
ALTER DATABASE [tpcc] SET TARGET_RECOVERY_TIME = 300 SECONDS WITH NO_WAIT
GO
ALTER DATABASE [tpcc] SET DELAYED_DURABILITY = DISABLED WITH NO_WAIT
GO
```

```
USE [tpcc]
GO

PRINT N'Altering indexes on [dbo].[customer]...';
GO
DROP INDEX dbo.customer.customer_i2
GO
CREATE NONCLUSTERED INDEX [customer_c_last]
 ON [dbo].[customer]([c_w_id] ASC, [c_d_id] ASC, [c_last] ASC, [c_first] ASC, [c_id] ASC) WITH (ALLOW_
    PAGE_LOCKS = OFF);
 GO


PRINT N'Altering indexes on [dbo].[district]...';
GO
DROP INDEX [dbo].[district].[d_details]
GO
CREATE NONCLUSTERED INDEX [district_d_id]
 ON [dbo].[district]([d_id] ASC) WITH (ALLOW_PAGE_LOCKS = OFF);
 GO
ALTER INDEX [PK_DISTRICT] ON [dbo].[district] SET (ALLOW_PAGE_LOCKS = OFF)
GO

PRINT N'Altering indexes on [dbo].[history]...';
GO
CREATE CLUSTERED INDEX [history_i1]
 ON [dbo].[history]([h_c_w_id] ASC, [h_date] ASC, [h_c_d_id] ASC, [h_c_id] ASC, [h_amount] ASC);
GO

PRINT N'Altering indexes on [dbo].[item]...';
GO
ALTER INDEX [PK_ITEM] ON [dbo].[item] SET (ALLOW_ROW_LOCKS = OFF)
GO

PRINT N'Altering indexes on [dbo].[new_order]...';
GO
ALTER INDEX [new_order_i1] ON [dbo].[new_order] SET (ALLOW_ROW_LOCKS = OFF)
GO
ALTER INDEX [new_order_i1] ON [dbo].[new_order] SET (ALLOW_PAGE_LOCKS = ON)
GO

PRINT N'Altering indexes on [dbo].[order_line]...';
GO
ALTER INDEX [order_line_i1] ON [dbo].[order_line] SET (ALLOW_ROW_LOCKS = OFF)
GO
ALTER INDEX [order_line_i1] ON [dbo].[order_line] SET (ALLOW_PAGE_LOCKS = ON)
GO

PRINT N'Altering indexes on [dbo].[orders]...';
GO
ALTER INDEX [orders_i1] ON [dbo].[orders] SET (ALLOW_ROW_LOCKS = OFF)
GO
ALTER INDEX [orders_i1] ON [dbo].[orders] SET (ALLOW_PAGE_LOCKS = ON)
GO
ALTER INDEX [orders_i2] ON [dbo].[orders] SET (ALLOW_ROW_LOCKS = OFF)
GO
```

```
ALTER INDEX [orders_i2] ON [dbo].[orders] SET (ALLOW_PAGE_LOCKS = ON)
GO

PRINT N'Altering indexes on [dbo].[stock]...';
GO
ALTER INDEX [PK_STOCK] ON [dbo].[stock] SET (ALLOW_PAGE_LOCKS = OFF)
GO

PRINT N'Altering indexes on [dbo].[warehouse]...';
GO
ALTER INDEX [PK_WAREHOUSE] ON [dbo].[warehouse] SET (ALLOW_PAGE_LOCKS = OFF)
GO
DROP INDEX [dbo].[warehouse].[w_details]
GO

GO
PRINT N'Altering SqlProcedure [dbo].[ostat]...';


GO



ALTER PROCEDURE [dbo].[ostat]
@os_w_id int,
@os_d_id int,
@os_c_id int,
@byname int,
@os_c_last char(20)
AS
BEGIN
SET ANSI_WARNINGS OFF
DECLARE
@os_c_first char(16),
@os_c_middle char(2),
@os_c_balance money,
@os_o_id int,
@os_entdate datetime2(0),
@os_o_carrier_id int,
@os_ol_i_id        INT,
@os_ol_supply_w_id INT,
@os_ol_quantity INT,
@os_ol_amount      INT,
@os_ol_delivery_d DATE,
@namecnt int,
@i int
BEGIN TRANSACTION
BEGIN TRY

IF (@byname = 1)
BEGIN
  SELECT TOP 1
    @os_c_id = c_id,
    @os_c_balance = c_balance,
    @os_c_first = c_first,
    @os_c_middle = c_middle,
```

```
    @os_c_last = c_last
  FROM (
    SELECT TOP 50 PERCENT c_id, c_balance, c_first, c_middle, c_last
    FROM dbo.customer WITH (repeatableread)
    WHERE
      c_last = @os_c_last AND
      c_w_id = @os_w_id AND
      c_d_id = @os_d_id
    ORDER BY c_first) X
  ORDER BY c_first desc
END

ELSE
BEGIN
  SELECT @os_c_balance = customer.c_balance, @os_c_first = customer.c_first
  , @os_c_middle = customer.c_middle, @os_c_last = customer.c_last
  FROM dbo.customer WITH (repeatableread)
  WHERE customer.c_id = @os_c_id AND customer.c_d_id = @os_d_id AND customer.c_w_id = @os_w_id
END


BEGIN
SELECT TOP (1) @os_o_id = o_id, @os_o_carrier_id = o_carrier_id, @os_entdate = o_entry_d
FROM dbo.orders WITH (serializable)
WHERE orders.o_d_id = @os_d_id
AND orders.o_w_id = @os_w_id
AND orders.o_c_id = @os_c_id
ORDER BY orders.o_id DESC

IF @@ROWCOUNT = 0
PRINT 'No orders for customer';
END

SELECT order_line.ol_i_id
, order_line.ol_supply_w_id
, order_line.ol_quantity
, order_line.ol_amount
, order_line.ol_delivery_d
FROM dbo.order_line WITH (repeatableread)
WHERE order_line.ol_o_id = @os_o_id
AND order_line.ol_d_id = @os_d_id
AND order_line.ol_w_id = @os_w_id

SELECT@os_c_id as N'@os_c_id', @os_c_last as N'@os_c_last', @os_c_first as N'@os_c_first', @os_c_middle
    as N'@os_c_middle', @os_c_balance as N'@os_c_balance', @os_o_id as N'@os_o_id', @os_entdate as
    N'@os_entdate', @os_o_carrier_id as N'@os_o_carrier_id'
END TRY
BEGIN CATCH
SELECT
ERROR_NUMBER() AS ErrorNumber
,ERROR_SEVERITY() AS ErrorSeverity
,ERROR_STATE() AS ErrorState
,ERROR_PROCEDURE() AS ErrorProcedure
,ERROR_LINE() AS ErrorLine
,ERROR_MESSAGE() AS ErrorMessage;
IF @@TRANCOUNT > 0
ROLLBACK TRANSACTION;
```

```
END CATCH;
IF @@TRANCOUNT > 0
COMMIT TRANSACTION;
END
GO
PRINT N'Altering SqlProcedure [dbo].[delivery]...';


GO


ALTER PROCEDURE [dbo].[delivery]
@d_w_id int,
@d_o_carrier_id int,
@timestamp datetime2(0)
AS
BEGIN
SET ANSI_WARNINGS OFF
DECLARE
@d_no_o_id int,
@d_d_id int,
@d_c_id int,
@d_ol_total int
BEGIN TRANSACTION
BEGIN TRY
DECLARE
@loop_counter int
SET @loop_counter = 1
WHILE @loop_counter <= 10
BEGIN
SET @d_d_id = @loop_counter


SELECT TOP 1
@d_no_o_id = no_o_id
FROM dbo.new_order WITH (serializable updlock)
WHERE no_w_id = @d_w_id AND
no_d_id = @d_d_id
ORDER BY no_o_id ASC

IF (@@rowcount <> 0)
BEGIN
    -- claim the order for this district
    DELETE dbo.new_order
    WHERE no_w_id = @d_w_id AND
    no_d_id = @d_d_id AND
    no_o_id = @d_no_o_id

    UPDATE dbo.orders
    SET o_carrier_id = @d_o_carrier_id
    , @d_c_id = orders.o_c_id
    WHERE orders.o_id = @d_no_o_id
    AND orders.o_d_id = @d_d_id
    AND orders.o_w_id = @d_w_id

    SET @d_ol_total = 0
```

```
    UPDATE dbo.order_line
    SET ol_delivery_d = @timestamp
        , @d_ol_total = @d_ol_total + ol_amount
    WHERE order_line.ol_o_id = @d_no_o_id
    AND order_line.ol_d_id = @d_d_id
    AND order_line.ol_w_id = @d_w_id

    UPDATE dbo.customer SET c_balance = customer.c_balance + @d_ol_total
    WHERE customer.c_id = @d_c_id
    AND customer.c_d_id = @d_d_id
    AND customer.c_w_id = @d_w_id
END



PRINT
'D: '
+
ISNULL(CAST(@d_d_id AS nvarchar(4000)), '')
+
'O: '
+
ISNULL(CAST(@d_no_o_id AS nvarchar(4000)), '')
+
'time '
+
ISNULL(CAST(@timestamp AS nvarchar(4000)), '')
SET @loop_counter = @loop_counter + 1
END
SELECT@d_w_id as N'@d_w_id', @d_o_carrier_id as N'@d_o_carrier_id', @timestamp as N'@TIMESTAMP'
END TRY
BEGIN CATCH
SELECT
ERROR_NUMBER() AS ErrorNumber
,ERROR_SEVERITY() AS ErrorSeverity
,ERROR_STATE() AS ErrorState
,ERROR_PROCEDURE() AS ErrorProcedure
,ERROR_LINE() AS ErrorLine
,ERROR_MESSAGE() AS ErrorMessage;
IF @@TRANCOUNT > 0
ROLLBACK TRANSACTION;
END CATCH;
IF @@TRANCOUNT > 0
COMMIT TRANSACTION;
END
GO
PRINT N'Altering SqlProcedure [dbo].[payment]...';


GO

ALTER PROCEDURE [dbo].[payment]
@p_w_id int,
@p_d_id int,
@p_c_w_id int,
```

```
@p_c_d_id int,
@p_c_id int,
@byname int,
@p_h_amount numeric(6,2),
@p_c_last char(16),
@TIMESTAMP datetime2(0)
AS
BEGIN
SET ANSI_WARNINGS OFF
DECLARE
@p_w_street_1 char(20),
@p_w_street_2 char(20),
@p_w_city char(20),
@p_w_state char(2),
@p_w_zip char(10),
@p_d_street_1 char(20),
@p_d_street_2 char(20),
@p_d_city char(20),
@p_d_state char(20),
@p_d_zip char(10),
@p_c_first char(16),
@p_c_middle char(2),
@p_c_street_1 char(20),
@p_c_street_2 char(20),
@p_c_city char(20),
@p_c_state char(20),
@p_c_zip char(9),
@p_c_phone char(16),
@p_c_since datetime2(0),
@p_c_credit char(32),
@p_c_credit_lim  numeric(12,2),
@p_c_discount  numeric(4,4),
@p_c_balance money,       --numeric(12,2),
@p_c_data varchar(500),
@namecnt int,
@p_d_name char(11),
@p_w_name char(11),
@p_c_new_data varchar(500),
@h_data varchar(30)

BEGIN TRY

IF (@byname = 1)
BEGIN
SELECT TOP 1
  @p_c_id = c_id
FROM (
  SELECT TOP 50 PERCENT c_id, c_first
  FROM dbo.customer WITH (repeatableread)
  WHERE
    c_last = @p_c_last AND
    c_w_id = @p_c_w_id AND
    c_d_id = @p_c_d_id
  ORDER BY c_first) X
ORDER BY c_first desc
END
```

```
BEGIN TRANSACTION

-- get customer info and update balances
UPDATE dbo.customer
SET
  @p_c_balance = c_balance = c_balance - @p_h_amount,
  c_data =
    CASE
      WHEN c_credit <> 'BC' THEN c_credit
      ELSE LEFT(
           ISNULL(CAST(@p_c_id AS char), '') +   ' '    +
           ISNULL(CAST(@p_c_d_id AS char), '') + ' ' +
           ISNULL(CAST(@p_c_w_id AS char), '') + ' ' +
           ISNULL(CAST(@p_d_id AS char), '') + ' ' +
           ISNULL(CAST(@p_w_id AS char), '') + ' ' +
           ISNULL(CAST(@p_h_amount AS CHAR(8)), '') + ' ' +
           ISNULL(CAST(@TIMESTAMP AS char), '') + ' ' +
           ISNULL(@p_w_name, '') + ' ' +
           ISNULL(@p_d_name, '') + ' ' +
           c_data,
           500)
    END,
  @p_c_first = c_first,
  @p_c_middle = c_middle,
  @p_c_last = c_last,
  @p_c_street_1 = c_street_1,
  @p_c_street_2 = c_street_2,
  @p_c_city = c_city,
  @p_c_state = c_state,
  @p_c_zip = c_zip,
  @p_c_phone = c_phone,
  @p_c_credit = c_credit,
  @p_c_credit_lim = c_credit_lim,
  @p_c_discount = c_discount,
  @p_c_since = c_since
WHERE
  c_id = @p_c_id AND
  c_w_id = @p_c_w_id AND
  c_d_id = @p_c_d_id

SET @h_data = (ISNULL(@p_w_name, '') + ' ' + ISNULL(@p_d_name, ''))

INSERT dbo.history( h_c_d_id, h_c_w_id, h_c_id, h_d_id, h_w_id, h_date, h_amount, h_data)
VALUES ( @p_c_d_id, @p_c_w_id, @p_c_id, @p_d_id, @p_w_id, @TIMESTAMP, @p_h_amount, @h_data)

-- get district data and update year-to-date
UPDATE dbo.district
SET
  d_ytd = d_ytd + @p_h_amount,
  @p_d_street_1 = d_street_1,
  @p_d_street_2 = d_street_2,
  @p_d_city = d_city,
  @p_d_state = d_state,
  @p_d_zip = d_zip,
  @p_d_name = d_name
WHERE
```

```
  d_w_id = @p_w_id AND
  d_id = @p_d_id

-- get warehouse data and update year-to-date
UPDATE dbo.warehouse
SET
  w_ytd = w_ytd + @p_h_amount,
  @p_w_street_1 = w_street_1,
  @p_w_street_2 = w_street_2,
  @p_w_city = w_city,
  @p_w_state = w_state,
  @p_w_zip = w_zip,
  @p_w_name = w_name
WHERE
  w_id = @p_w_id

SELECT@p_c_id as N'@p_c_id', @p_c_last as N'@p_c_last', @p_w_street_1 as N'@p_w_street_1'
, @p_w_street_2 as N'@p_w_street_2', @p_w_city as N'@p_w_city'
, @p_w_state as N'@p_w_state', @p_w_zip as N'@p_w_zip'
, @p_d_street_1 as N'@p_d_street_1', @p_d_street_2 as N'@p_d_street_2'
, @p_d_city as N'@p_d_city', @p_d_state as N'@p_d_state'
, @p_d_zip as N'@p_d_zip', @p_c_first as N'@p_c_first'
, @p_c_middle as N'@p_c_middle', @p_c_street_1 as N'@p_c_street_1'
, @p_c_street_2 as N'@p_c_street_2'
, @p_c_city as N'@p_c_city', @p_c_state as N'@p_c_state', @p_c_zip as N'@p_c_zip'
, @p_c_phone as N'@p_c_phone', @p_c_since as N'@p_c_since', @p_c_credit as N'@p_c_credit'
, @p_c_credit_lim as N'@p_c_credit_lim', @p_c_discount as N'@p_c_discount', @p_c_balance as
    N'@p_c_balance'
, @p_c_data as N'@p_c_data'


END TRY
BEGIN CATCH
SELECT
ERROR_NUMBER() AS ErrorNumber
,ERROR_SEVERITY() AS ErrorSeverity
,ERROR_STATE() AS ErrorState
,ERROR_PROCEDURE() AS ErrorProcedure
,ERROR_LINE() AS ErrorLine
,ERROR_MESSAGE() AS ErrorMessage;
IF @@TRANCOUNT > 0
ROLLBACK TRANSACTION;
END CATCH;
IF @@TRANCOUNT > 0
COMMIT TRANSACTION;
END
GO
PRINT N'Altering SqlProcedure [dbo].[neword]...';


GO

ALTER PROCEDURE [dbo].[neword]
@no_w_id int,
@no_max_w_id int,
@no_d_id int,
```

```
@no_c_id int,
@no_o_ol_cnt int,
@TIMESTAMP datetime2(0)
AS
BEGIN
SET ANSI_WARNINGS OFF
DECLARE
@no_c_discount smallmoney,
@no_c_last char(16),
@no_c_credit char(2),
@no_d_tax smallmoney,
@no_w_tax smallmoney,
@no_d_next_o_id int,
@no_ol_supply_w_id int,
@no_ol_i_id int,
@no_ol_quantity int,
@no_o_all_local int,
@o_id int,
@no_i_name char(24),
@no_i_price smallmoney,
@no_i_data char(50),
@no_s_quantity int,
@no_ol_amount int,
@no_s_dist_01 char(24),
@no_s_dist_02 char(24),
@no_s_dist_03 char(24),
@no_s_dist_04 char(24),
@no_s_dist_05 char(24),
@no_s_dist_06 char(24),
@no_s_dist_07 char(24),
@no_s_dist_08 char(24),
@no_s_dist_09 char(24),
@no_s_dist_10 char(24),
@no_ol_dist_info char(24),
@no_s_data char(50),
@x int,
@rbk int


BEGIN TRANSACTION
BEGIN TRY

SET @no_o_all_local = 0

SELECT
  @no_c_discount = c_discount,
  @no_c_last = c_last,
  @no_c_credit = c_credit
FROM dbo.customer
WHERE
  c_w_id = @no_w_id AND
  c_d_id = @no_d_id AND
  c_id = @no_c_id

UPDATE dbo.district
SET
```

```
  @no_d_tax = d_tax,
  @o_id = d_next_o_id,
  @no_d_next_o_id = d_next_o_id = district.d_next_o_id + 1
WHERE district.d_id = @no_d_id
AND district.d_w_id = @no_w_id


SET @rbk = CAST(100 * RAND() + 1 AS INT)
DECLARE
@loop_counter int
SET @loop_counter = 1
DECLARE
@loop$bound int
SET @loop$bound = @no_o_ol_cnt
WHILE @loop_counter <= @loop$bound
BEGIN
IF ((@loop_counter = @no_o_ol_cnt) AND (@rbk = 1))
SET @no_ol_i_id = 100001
ELSE
SET @no_ol_i_id =  CAST(1000000 * RAND() + 1 AS INT)
SET @x = CAST(100 * RAND() + 1 AS INT)
IF (@x > 1)
SET @no_ol_supply_w_id = @no_w_id
ELSE
BEGIN
SET @no_ol_supply_w_id = @no_w_id
SET @no_o_all_local = 0
WHILE ((@no_ol_supply_w_id = @no_w_id) AND (@no_max_w_id != 1))
BEGIN
SET @no_ol_supply_w_id = CAST(@no_max_w_id * RAND() + 1 AS INT)
DECLARE
@db_null_statement$2 int
END
END
SET @no_ol_quantity = CAST(10 * RAND() + 1 AS INT)

SELECT @no_i_price = item.i_price
, @no_i_name = item.i_name
, @no_i_data = item.i_data
FROM dbo.item
WHERE item.i_id = @no_ol_i_id

UPDATE dbo.stock
SET
  s_quantity = s_quantity - @no_ol_quantity + CASE WHEN (s_quantity > @no_ol_quantity)
    THEN 0 ELSE 91 END,
  @no_s_data = s_data,
  @no_ol_dist_info =
    CASE @no_d_id
      WHEN 1 THEN s_dist_01
      WHEN 2 THEN s_dist_02
      WHEN 3 THEN s_dist_03
      WHEN 4 THEN s_dist_04
      WHEN 5 THEN s_dist_05
      WHEN 6 THEN s_dist_06
      WHEN 7 THEN s_dist_07
```

```
      WHEN 8 THEN s_dist_08
      WHEN 9 THEN s_dist_09
      WHEN 10 THEN s_dist_10
    END
OUTPUT
  @o_id,
  @no_d_id,
  @no_w_id,
  @loop_counter,
  @no_ol_i_id,
  NULL,
  (@no_ol_quantity * @no_i_price),
  @no_ol_supply_w_id,
  @no_ol_quantity,
  CASE @no_d_id
      WHEN 1 THEN inserted.s_dist_01
      WHEN 2 THEN inserted.s_dist_02
      WHEN 3 THEN inserted.s_dist_03
      WHEN 4 THEN inserted.s_dist_04
      WHEN 5 THEN inserted.s_dist_05
      WHEN 6 THEN inserted.s_dist_06
      WHEN 7 THEN inserted.s_dist_07
      WHEN 8 THEN inserted.s_dist_08
      WHEN 9 THEN inserted.s_dist_09
      WHEN 10 THEN inserted.s_dist_10
    END
INTO dbo.order_line
WHERE
  stock.s_i_id = @no_ol_i_id AND
  stock.s_w_id = @no_ol_supply_w_id

SET @loop_counter = @loop_counter + 1
END

INSERT dbo.orders( o_id, o_d_id, o_w_id, o_c_id, o_entry_d, o_ol_cnt, o_all_local)
VALUES ( @o_id, @no_d_id, @no_w_id, @no_c_id, @TIMESTAMP, @no_o_ol_cnt, @no_o_all_local)

INSERT dbo.new_order(no_o_id, no_d_id, no_w_id)
VALUES (@o_id, @no_d_id, @no_w_id)

IF (@rbk = 1)
ROLLBACK TRANSACTION

SELECT @no_w_tax = warehouse.w_tax
FROM dbo.warehouse
WHERE warehouse.w_id = @no_w_id

SELECT convert(char(8), @no_c_discount) as N'@no_c_discount', @no_c_last as N'@no_c_last', @no_c_
    credit as N'@no_c_credit', convert(char(8),@no_d_tax) as N'@no_d_tax', convert(char(8),@no_w_tax)
    as N'@no_w_tax', @no_d_next_o_id as N'@no_d_next_o_id'

END TRY
BEGIN CATCH
SELECT
ERROR_NUMBER() AS ErrorNumber
,ERROR_SEVERITY() AS ErrorSeverity
,ERROR_STATE() AS ErrorState
```

```
,ERROR_PROCEDURE() AS ErrorProcedure
,ERROR_LINE() AS ErrorLine
,ERROR_MESSAGE() AS ErrorMessage;
IF @@TRANCOUNT > 0
ROLLBACK TRANSACTION;
END CATCH;
IF @@TRANCOUNT > 0
COMMIT TRANSACTION;

END
GO
PRINT N'Altering SqlProcedure [dbo].[slev]...';


GO

ALTER PROCEDURE [dbo].[slev]
@st_w_id int,
@st_d_id int,
@threshold int
AS
BEGIN
DECLARE
@st_o_id_low int,
@st_o_id_high int
BEGIN TRANSACTION
BEGIN TRY

SELECT
@st_o_id_low = district.d_next_o_id - 20,
@st_o_id_high = district.d_next_o_id - 1
FROM dbo.district
WHERE district.d_w_id = @st_w_id AND district.d_id = @st_d_id

SELECT
COUNT(DISTINCT stock.s_i_id)
FROM dbo.order_line
, dbo.stock
WHERE order_line.ol_w_id = @st_w_id
AND order_line.ol_d_id = @st_d_id
AND order_line.ol_o_id BETWEEN @st_o_id_low AND @st_o_id_high
AND stock.s_w_id = order_line.ol_w_id
AND stock.s_i_id = order_line.ol_i_id
AND stock.s_quantity < @threshold
OPTION (ORDER GROUP)

END TRY
BEGIN CATCH
SELECT
ERROR_NUMBER() AS ErrorNumber
,ERROR_SEVERITY() AS ErrorSeverity
,ERROR_STATE() AS ErrorState
,ERROR_PROCEDURE() AS ErrorProcedure
,ERROR_LINE() AS ErrorLine
,ERROR_MESSAGE() AS ErrorMessage;
IF @@TRANCOUNT > 0
```

```
ROLLBACK TRANSACTION;
END CATCH;
IF @@TRANCOUNT > 0
COMMIT TRANSACTION;
END
GO
PRINT N'Update complete.';


GO
```

Read the report at http://facts.pt/5ro20rg ▶

This project was commissioned by AWS.