



Ask Why My Query So Slow?

Jason Wong

<http://usa.redirectme.net>

Sr. DBA

IT Applications Manager

DBA Developer

Programmer, App Admin

M.S. Rice '88, MBA U.H. '94 (MIS)



#57 HOUSTON

Houston Area SQL Server User Group – houston.sqlpass.org

Jason Wong

- ▶ Jason Wong has worked in database related IT over 20+ years. The first database he mastered was Ashton DBIV on IBM386. He have held developer, IT Applications Manager, Sr. DBA positions in various industries.
- ▶ Jason Wong is always dedicated to advanced technology, taking up mostly challenging tasks. He has two master's degrees and has presented a paper to NASA JSC after grad-school about robotics collision avoidance.
- ▶ Jason Wong is currently a Sr. DBA in Houston area and a SQLPASS volunteer. Jason Wong has a web site since 1995. URL: <http://usa.redirectme.net>
- ▶ At leisure time, Jason likes to travel with his family around the world to see different cultures. His hobbies include tennis, computer and help out his wife in the garden.

Ask Why My Query So Slow?

- ▶ ~~Application slow~~
- ▶ ~~Network slow~~
- ▶ ~~User slow~~



- ▶ Database slow
- ▶RAM, Disk IO, CPU
- ▶Blocking.....

What you don't know won't hurt you.



Database slow

- ▶ code inefficiency,
- ▶ poor execution plan,
- ▶ disk fragmentation,
- ▶ index fragmentation,
- ▶ insufficient RAM,
- ▶ poor IO performance.

RAM

- ▶ Buffer pool, Procedure cache, Memtoleave, SSRS etc.
- ▶ 64-bit VS 32-bit
- ▶ (VAS limitation) $2^{31} = 2147483648$
- ▶ AWE, lock page in memory
- ▶ Max, min memory parameters
- ▶ DDR2 800MHz 64-bits $\rightarrow = 51.2$ Gbps
- ▶ FusionIO SSD \rightarrow PCIe (2) = 5 Gbps
- ▶ Hard-disk IO on RAID
- ▶ \rightarrow magnetic harddisk invented 60's
- ▶ \rightarrow scale down of x20 - x50

FusionIO – Case Study (from FusionIO web site)

METRIC	LEGACY	POST ioDRIVE	IMPROVEMENT FACTOR	CUSTOMER IMPACT
Average duration of an SQL transaction	345ms	88ms	4 times	Web pages load faster, improving the customer experience, which leads to higher sales conversion rates.
Time for full backup of the largest database	2 hours	6 minutes	20 times	There is no perceived impact of competition for I/O resources during backup procedures. The window in which this competition occurs in is much smaller.
Time to restore a full backup of the largest database	3 hours	15 minutes	12 times	Faster recovery time means less loss exposure in the event of a major outage.
Average number of read/write operations waiting in a queue to complete	0.4	0.008	50 times	Customers compete with other customers for resources much less frequently.
Number of transactions in one hour window that took more than 500 milliseconds	3011	163	18 times	Web pages load faster, improving the customer experience, which leads to higher sales conversion rates. Also, the system can support more cart transactions per second.

Perfmon counters:

cntr_value	counter_name	object_name
15	Free list stalls/sec	SQLServer:Buffer Manager
69428	Free pages	SQLServer:Buffer Manager
1272213	Stolen pages	SQLServer:Buffer Manager
68690087	Page reads/sec	SQLServer:Buffer Manager
35300865	Page writes/sec	SQLServer:Buffer Manager
26477768	Checkpoint pages/sec	SQLServer:Buffer Manager
40436	Page life expectancy	SQLServer:Buffer Manager
0	Memory Grants Pending	SQLServer:Memory Manager
100212736	Target Server Memory (KB)	SQLServer:Memory Manager
100212736	Total Server Memory (KB)	SQLServer:Memory Manager

- Buffer Hit Ratio
- Avg. Disk S/Read
- Avg. Disk S/Write
- Page Faults /S
- Lazy Writes
- Processor Queue Length
- Compilations/sec
- Re-Compilations/sec
- Cache Hit Ratio
- % Processor Time
- Context Switches/sec

```
CREATE VIEW [dbo].[View_Memory_Counters]
AS
SELECT TOP (100) PERCENT cntr_value, counter_name, object_name
FROM sys.dm_os_performance_counters
WHERE (object_name = 'SQLServer:Memory Manager') AND (counter_name = 'Memory Grants
Pending'
    or counter_name = 'Total Server Memory (KB)'
    or counter_name = 'Target Server Memory (KB)'
)
OR
(object_name = 'SQLServer:Buffer Manager') AND (counter_name = 'Free list stalls/sec'
OR counter_name = 'Page life expectancy' OR
    counter_name = 'Page reads/sec' OR counter_name = 'Page writes/sec' OR
    counter_name = 'Checkpoint pages/sec' OR counter_name = 'Free pages')
ORDER BY counter_name
GO
```

```

WITH SortedCounters AS
(
    SELECT a.RID AS ROWID, a.object_name AS ObjName, a.counter_name AS CntName, a.HostDBInstance AS DBIntName,
           a.cnt_value AS cntvalue, a.CreatedDateTime,
           ROW_NUMBER() OVER (ORDER BY a.CreatedDateTime) AS 'RowNumber'
    FROM [PerfmonHistory].[dbo].[PerfmonCounterHistory] AS a
    where a.counter_name = 'Checkpoint pages/sec'
           AND a.HostDBInstance = 'DataCenterHost'
)
Select top 100 a.ROWID, a.ObjName, a.CntName, a.DBIntName, (a.cntvalue-b.cntvalue)/360 AS TimedValuePerSec,
a.CreatedDateTime, a.RowNumber, b.RowNumber
from SortedCounters a, SortedCounters b where b.RowNumber = a.RowNumber-1 order by a.RowNumber desc

```

Results Messages

ROWID	ObjName	CntName	DB...	TimedValuePerSec	CreatedDateTime	RowNumber	RowNumber
844...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	40	2011-01-10 14:48:00.817	20596	20595
023...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	3	2011-01-10 14:42:00.423	20595	20594
0A1...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	21	2011-01-10 14:36:00.200	20594	20593
4D5...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	2	2011-01-10 14:30:01.060	20593	20592
38A...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	188	2011-01-10 14:24:00.890	20592	20591
E3B...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	34	2011-01-10 14:18:00.763	20591	20590
AD8...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	0	2011-01-10 14:12:00.593	20590	20589
7AD...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	51	2011-01-10 14:06:01.427	20589	20588
381...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	62	2011-01-10 14:00:00.270	20588	20587
EE0...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	3	2011-01-10 13:54:01.163	20587	20586
893...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	40	2011-01-10 13:48:01.000	20586	20585
2F1...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	3	2011-01-10 13:42:00.773	20585	20584
B36...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	5	2011-01-10 13:36:00.813	20584	20583
ED5...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	93	2011-01-10 13:30:00.660	20583	20582
700...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	4	2011-01-10 13:24:00.340	20582	20581
70A...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	13	2011-01-10 13:18:00.217	20581	20580
5C6...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	55	2011-01-10 13:12:01.003	20580	20579
C20...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	18	2011-01-10 13:06:00.960	20579	20578
B02...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	27	2011-01-10 13:00:00.780	20578	20577
F1A...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	5	2011-01-10 12:54:00.607	20577	20576
DB6...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	36	2011-01-10 12:48:00.677	20576	20575
7B8...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	3	2011-01-10 12:42:00.317	20575	20574
7748...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	10	2011-01-10 12:36:00.377	20574	20573
875...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	18	2011-01-10 12:30:01.070	20573	20572
9CE...	SQLServer:Buffer Manager	Checkpoint pages/sec	D...	13	2011-01-10 12:24:00.887	20572	20571

Powershell Script Monitoring CPU

```
$server = "ServerHost"
$namespace = "root\CIMV2"
    trap [System.Data.SqlClient.SqlException] { break; } #cannot reach the db!

    $processor = Get-WmiObject -class Win32_PerfFormattedData_PerfOS_Processor -Property Name,PercentProcessorTime -
computerName $server -namespace $namespace
    if($processor -ne $null)
    {
        $conn = new-object System.Data.SqlClient.SqlConnection("data source=SQL2K8;initial
catalog=PerfmonHistory;integrated security=SSPI")
        $conn.open()
        # if the system only has one processor, we don't get an array of objects but a single object
        if($processor -is [array])
        {
            for($i = 0; $i -lt $processor.Count; $i++)
            {
                $query = "INSERT INTO [PerfmonHistory].[dbo].[PerfmonCounterHistory]
([object_name],[counter_name],[instance_name],[cntr_value],[cntr_type],[HostDBInstance]) VALUES ('System', 'Processor', '' +
$processor[$i].Name + "', " + [decimal]$processor[$i].PercentProcessorTime + ", '0', '' + $server + '')"
                $cmd = new-object "System.Data.SqlClient.SqlCommand" ($query, $conn)
                $cmd.executenonquery()
            }
        }
        else
        {
            $query = "INSERT INTO [PerfmonHistory].[dbo].[PerfmonCounterHistory]
([object_name],[counter_name],[instance_name],[cntr_value],[cntr_type],[HostDBInstance]) VALUES ('System', 'Processor', '' +
$processor.Name + "', " + [decimal]$processor.PercentProcessorTime + ", '0', '' + $computer + '')"
            $cmd = new-object "System.Data.SqlClient.SqlCommand" ($query, $conn)
            $cmd.executenonquery()
        }
        $processor = $null
        $conn.close()
    }
}
```

RID	object_na...	counter_name	instance_name	cntr_value	cntr_type	Host...	CreatedDateTime
4A...	System	Processor	_Total	32	0	DA...	2011-01-20 07:48:02.237
49...	System	Processor	7	28	0	DA...	2011-01-20 07:48:02.223
48...	System	Processor	6	75	0	DA...	2011-01-20 07:48:02.217
47...	System	Processor	5	12	0	DA...	2011-01-20 07:48:02.210
46...	System	Processor	4	15	0	DA...	2011-01-20 07:48:02.200
45...	System	Processor	3	22	0	DA...	2011-01-20 07:48:02.193
44...	System	Processor	2	84	0	DA...	2011-01-20 07:48:02.187
43...	System	Processor	1	3	0	DA...	2011-01-20 07:48:02.170
42...	System	Processor	0	22	0	DA...	2011-01-20 07:48:02.150
28...	System	Processor	_Total	6	0	DA...	2011-01-20 07:42:04.303
27...	System	Processor	7	0	0	DA...	2011-01-20 07:42:04.287
26...	System	Processor	6	3	0	DA...	2011-01-20 07:42:04.263
25...	System	Processor	5	31	0	DA...	2011-01-20 07:42:04.257
24...	System	Processor	4	0	0	DA...	2011-01-20 07:42:04.247
23...	System	Processor	3	0	0	DA...	2011-01-20 07:42:04.240
22...	System	Processor	2	0	0	DA...	2011-01-20 07:42:04.230
21...	System	Processor	1	15	0	DA...	2011-01-20 07:42:04.223
20...	System	Processor	0	0	0	DA...	2011-01-20 07:42:04.203
16...	System	Processor	_Total	50	0	DA...	2011-01-20 07:36:02.090
15...	System	Processor	7	44	0	DA...	2011-01-20 07:36:02.070
14...	System	Processor	6	38	0	DA...	2011-01-20 07:36:02.063
13...	System	Processor	5	53	0	DA...	2011-01-20 07:36:02.050
12...	System	Processor	4	50	0	DA...	2011-01-20 07:36:02.040
11...	System	Processor	3	38	0	DA...	2011-01-20 07:36:02.033
10...	System	Processor	2	25	0	DA...	2011-01-20 07:36:02.027
0F...	System	Processor	1	84	0	DA...	2011-01-20 07:36:02.020
0E...	System	Processor	0	66	0	DA...	2011-01-20 07:36:02.000
D6...	System	Processor	_Total	45	0	DA...	2011-01-20 07:30:02.837
D5...	System	Processor	7	78	0	DA...	2011-01-20 07:30:02.823
D4...	System	Processor	6	41	0	DA...	2011-01-20 07:30:02.807
D3...	System	Processor	5	48	0	DA...	2011-01-20 07:30:02.800
D2...	System	Processor	4	84	0	DA...	2011-01-20 07:30:02.790
D1...	System	Processor	3	8	0	DA...	2011-01-20 07:30:02.783
D0...	System	Processor	2	14	0	DA...	2011-01-20 07:30:02.777
CF...	System	Processor	1	41	0	DA...	2011-01-20 07:30:02.770
CE...	System	Processor	0	44	0	DA...	2011-01-20 07:30:02.747
B8...	System	Processor	_Total	15	0	DA...	2011-01-20 07:24:02.560
B7...	System	Processor	7	10	0	DA...	2011-01-20 07:24:02.543
B6...	System	Processor	6	80	0	DA...	2011-01-20 07:24:02.530
B5...	System	Processor	5	1	0	DA...	2011-01-20 07:24:02.520

	Metric	Value	Reference Range	State	Category	Warning Threshold	Critical Threshold
<input type="checkbox"/>	FreePages	8,956.00	N/A	Critical	Custom	15,000	10,000
<input type="checkbox"/>	PageLifeExpectancy	134.00	N/A	Critical	Custom	600	300
<input type="checkbox"/>	PageReadPerSec	9,100.88	N/A	Critical	Custom	90	110
<input type="checkbox"/>	System Threads	1,785.00	N/A	Critical	Custom	500	555
<input checked="" type="checkbox"/>	LazyWrites	22.51	N/A	Warning	Custom	20	40
<input type="checkbox"/>	PageWritePerSec	96.47	N/A	Warning	Custom	90	110
<input type="checkbox"/>	Cache Faults/sec	0.00	N/A	OK	Custom	500	1,000
<input type="checkbox"/>	CheckpointPages	10.50	N/A	OK	Custom	300	600
<input type="checkbox"/>	Context Switch	6,208.00	N/A	OK	Custom	7,000	8,000
<input type="checkbox"/>	CPU0	2.00	N/A	OK	Custom	80	99
<input type="checkbox"/>	CPU1	0.00	N/A	OK	Custom	80	99
<input type="checkbox"/>	CPU2	2.00	N/A	OK	Custom	80	99
<input type="checkbox"/>	CPU3	0.00	N/A	OK	Custom	80	99
<input type="checkbox"/>	CPU4	0.00	N/A	OK	Custom	80	99
<input type="checkbox"/>	CPU5	2.00	N/A	OK	Custom	80	99
<input type="checkbox"/>	CPU6	2.00	N/A	OK	Custom	80	99
<input type="checkbox"/>	CPU7	0.00	N/A	OK	Custom	80	99
<input type="checkbox"/>	FreeListStalls	0.00	N/A	OK	Custom	2	4
<input type="checkbox"/>	MemoryGrantPending	0.00	N/A	OK	Custom	1	2
<input type="checkbox"/>	Page Reads/sec	0.00	N/A	OK	Custom	500	1,000
<input type="checkbox"/>	Percent Privileged Time	0.00	N/A	OK	Custom	20	25
<input type="checkbox"/>	Percent Disk Time	0.00	N/A	OK	Custom	50	55
<input type="checkbox"/>	SDriveSecPerRead	0.00	N/A	OK	Custom	10	20
<input type="checkbox"/>	SDriveSecPerWrite	0.00	N/A	OK	Custom	10	20
<input type="checkbox"/>	Stolen Pages per sec	-0.01	N/A	OK	Custom	300	600
<input type="checkbox"/>	TDriveSecPerRead	0.00	N/A	OK	Custom	5	10
<input type="checkbox"/>	TDriveSecPerWrite	0.00	N/A	OK	Custom	5	10
<input type="checkbox"/>	Page Faults/sec	60.00	N/A	N/A	Custom		
<input type="checkbox"/>	TargetServerMemory	100.21	N/A	N/A	Custom		
<input type="checkbox"/>	TotalServerMemory	100.21	N/A	N/A	Custom		

Appropriate RAID Configuration

- ▶ Raid 0: I/Os per disk = (reads + writes) / number of disks
- ▶
- ▶ Raid 1: I/Os per disk = [reads + (2 * writes)] / 2
- ▶
- ▶ Raid 5: I/Os per disk = [reads + (4 * writes)] / number of disks
- ▶
- ▶ Raid 10: I/Os per disk = [reads + (2 * writes)] / number of disks
- ▶
- ▶ Raid 6: I/Os per disk = [reads + (6 * writes)] / number of disks
- ▶
- ▶

RAID 1+0 VS 0+1

RAID 0+1 (Mirrored Stripes)

RAID Level 0+1 is a **mirror** (RAID 1) array whose segments are striped (RAID 0) arrays. It is a great alternative for users that like the security of RAID 1 but need some additional **performance boost**.


- Minimum number of drives required: 4


Advantages

- Fault tolerant
- Very High I/O rates

Disadvantages

- Very expensive
- High overhead
- Very limited scalability

performance  good

reliability  good

efficiency  low

RAID 1+0 (Striped Mirrors)

RAID Level 10 is a **striped** (RAID 0) array whose segments are **mirrored** (RAID 1). It is similar in performance to RAID 0+1, but with better fault tolerance and rebuild performance.


- Minimum number of drives required: 4


Advantages

- High fault tolerance
- High I/O rates
- Faster rebuild performance than RAID 0+1
- Under certain circumstances, RAID 10 array can sustain multiple simultaneous drive failures

Disadvantages

- Very expensive
- High overhead
- Very limited scalability

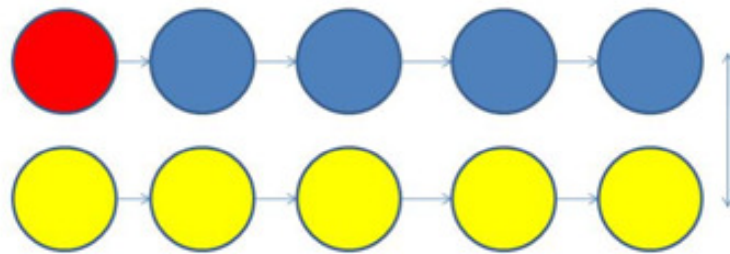
performance  good

reliability  good

efficiency  low

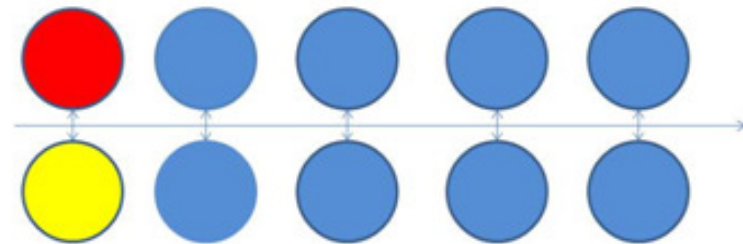
RAID 1+0 VS 0+1

RAID 0+1



Strip (0), then Mirror (1)
 $1/10 * 5/9 = 5/90$

RAID 1+0

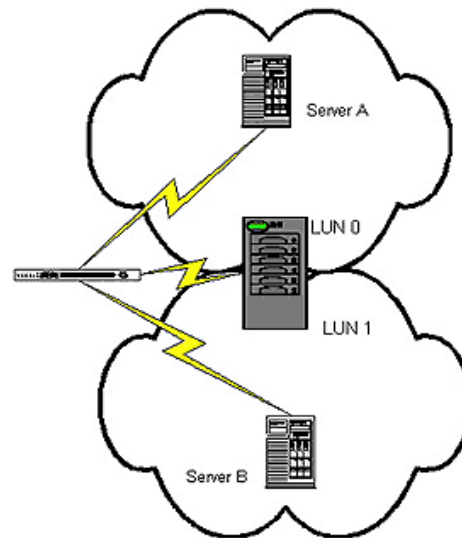


- Mirror (1), then Strip (0)
- $1/10 * 1/9 = 1/90$

Appropriate RAID Configuration

LUN Masking

LUN masking, like zoning, prevents servers from seeing all but specific storage resources, but it is more efficient and granular since it can be used to control LUNs within a storage device. LUNs are either individual disks, groups of disks, or individual parts of multiple disks defined by a RAID controller. LUNs, which most people commonly refer to as partitions or logical disks, are granular storage entities that are carved out of a single storage system (be it a RAID or JBOD or even a tape library). Because multiple LUNs can reside on a single storage system, multiple computers can access the LUNs through a single wire connection to a storage system with LUN masking, a situation that is far more scalable than zoning, which is hampered by its 1:1 setup (one port: one connection).



[click on image for full view](#)

With LUN Masking you can use a single Fibre Channel link to split up a RAID unit into multiple logical parts.

Database blocking

Isolation level:

Isolation level	Dirty read	Nonrepeatable read	Phantom
Read uncommitted	Yes	Yes	Yes
Read committed	No	Yes	Yes
Repeatable read	No	No	Yes
Snapshot	No	No	No
Serializable	No	No	No

ACID (transactions), Locking/Blocking

```
ALTER DATABASE .... SET ALLOW_SNAPSHOT_ISOLATION on (Snapshot)
```

```
ALTER DATABASE .... SET READ_COMMITTED_SNAPSHOT on (Read committed)
```


Isolation side-effect

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read
Read Uncommitted	Yes	Yes	Yes
Read Committed (locking)	No	Yes	Yes
Read Committed_S napshot (RowVersioning)	No	Yes	Yes
Repeatable Read	No	No	Yes
Snapshot	No	No	No
Serializable	No	No	No

(Kalen Delany, SQLServer)

(Thomas Kyte, Oracle)

Non-repeatable read – Users

```
Microsoft SQL Server 2008 (SP2) - 10.0.3798.0 (Intel X86) Jun 18 2010 16:55:53 Copyright (c) 1988-2008
Microsoft Corporation Standard Edition on Windows NT 5.1 <X86> (Build 2600: Service Pack 3)
```

```
DBCC FREEPROCCACHE
DBCC DROPCLEANBUFFERS
--ALTER DATABASE AdventureWorks SET ALLOW_SNAPSHOT_ISOLATION on
ALTER DATABASE AdventureWorks SET READ_COMMITTED_SNAPSHOT on
SELECT @@VERSION
SELECT @@SPID
--User 1
USE AdventureWorks;
SET NOCOUNT ON;|
--SET TRANSACTION ISOLATION LEVEL SNAPSHOT
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
begin tran
SELECT rate, PayFrequency, [EmployeeID] FROM [AdventureWorks].[HumanResources].[EmployeePayHistory] where
EmployeeID = 1
    waitfor delay '00:00:30'
SELECT rate, PayFrequency, [EmployeeID] FROM [AdventureWorks].[HumanResources].[EmployeePayHistory] where
EmployeeID = 1
commit tran --rollback tran

SELECT @@VERSION
SELECT @@SPID

--User 2
USE AdventureWorks;
--SET TRANSACTION ISOLATION LEVEL SNAPSHOT
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
--User 2
begin tran
update [AdventureWorks].[HumanResources].[EmployeePayHistory] set Rate = 13.45
where EmployeeID = 1
commit tran --rollback tran
(1 row(s) affected)
```

Non-repeatable read – Result

```
SQLQuery6.sql - P9661VF1\.....5))*  SQLQuery5.sql - P9661VF1\.....4))*  SQLQuery4.sql - P9661VF1\.....3))*
-- DBCC FREEPROCCACHE
-- DBCC DROPCLEANBUFFERS
-- SELECT @@VERSION
-- SELECT @@SPID

--User 1
begin tran
-- SELECT rate, PayFrequency,
--     [EmployeeID]
-- FROM [AdventureWorks].[HumanResources].[EmployeePayHistory]
-- where EmployeeID = 1
-- waitfor delay '00:00:30'
-- SELECT rate, PayFrequency,
--     [EmployeeID]
-- FROM [AdventureWorks].[HumanResources].[EmployeePayHistory]
-- where EmployeeID = 1
commit tran
--rollback tran
```

Results Messages

	rate	PayFrequen...	EmployeeID
1	12.45	1	1

	rate	PayFrequen...	EmployeeID
1	13.45	1	1

Phantom read User1

```
DBCC FREEPROCCACHE
DBCC DROPCLEANBUFFERS
SELECT @@VERSION
SELECT @@SPID
--ALTER DATABASE AdventureWorks SET ALLOW_SNAPSHOT_ISOLATION on
ALTER DATABASE AdventureWorks SET READ_COMMITTED_SNAPSHOT on

--User 1
USE AdventureWorks;
SET NOCOUNT ON;
--SET TRANSACTION ISOLATION LEVEL SNAPSHOT
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
begin tran
SELECT s.CustomerID, s.OrderDate, s.SalesOrderID, s.TotalDue, SUM(s2.TotalDue) AS RunningTotal
FROM Sales.SalesOrderHeader AS s INNER JOIN Sales.SalesOrderHeader AS s2
ON s2.CustomerID = s.CustomerID AND( s2.OrderDate < s.OrderDate OR( s2.OrderDate = s.OrderDate AND
s2.SalesOrderID <= s.SalesOrderID))
GROUP BY s.CustomerID, s.OrderDate, s.SalesOrderID, s.TotalDue
ORDER BY s.CustomerID, s.OrderDate, s.SalesOrderID;
    waitfor delay '00:00:30'
SELECT s.CustomerID, s.OrderDate, s.SalesOrderID, s.TotalDue, SUM(s2.TotalDue) AS RunningTotal
FROM Sales.SalesOrderHeader AS s INNER JOIN Sales.SalesOrderHeader AS s2
ON s2.CustomerID = s.CustomerID AND( s2.OrderDate < s.OrderDate OR( s2.OrderDate = s.OrderDate AND
s2.SalesOrderID <= s.SalesOrderID))
GROUP BY s.CustomerID, s.OrderDate, s.SalesOrderID, s.TotalDue
ORDER BY s.CustomerID, s.OrderDate, s.SalesOrderID;
commit tran
--rollback tran
```

Phantom read User2

```
SELECT @@VERSION
SELECT @@SPID

--User 2
USE AdventureWorks;
--SET TRANSACTION ISOLATION LEVEL SNAPSHOT
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
begin tran
INSERT INTO [AdventureWorks].[Sales].[SalesOrderHeader]
([RevisionNumber],[OrderDate],[DueDate],[ShipDate],[Status],[OnlineOrderFlag],[PurchaseOrderNumber]
,[AccountNumber],[CustomerID],[ContactID],[SalesPersonID],[TerritoryID],[BillToAddressID],[ShipToAddressID]
,[ShipMethodID],[CreditCardID],[CreditCardApprovalCode],[CurrencyRateID],[SubTotal],[TaxAmt],[Freight],[Com
ment]
)
SELECT
    [RevisionNumber],[OrderDate],[DueDate],[ShipDate],[Status]
,[OnlineOrderFlag],[PurchaseOrderNumber],[AccountNumber],[CustomerID],[ContactID],[SalesPersonID],[Territor
yID],[BillToAddressID]
,[ShipToAddressID],[ShipMethodID],[CreditCardID],[CreditCardApprovalCode],[CurrencyRateID],[SubTotal],[TaxA
mt],[Freight],[Comment]
FROM [AdventureWorks].[Sales].[SalesOrderHeader]
    where CustomerID = 1
commit tran
--rollback tran

(4 row(s) affected)
```

Phantom read Results

```

USE AdventureWorks;
SET NOCOUNT ON;
begin tran
SELECT s.CustomerID, s.OrderDate, s.SalesOrderID, s.TotalDue, SUM(s2.TotalDue) AS RunningTotal
FROM Sales.SalesOrderHeader AS s INNER JOIN Sales.SalesOrderHeader AS s2
ON s2.CustomerID = s.CustomerID AND( s2.OrderDate < s.OrderDate OR( s2.OrderDate = s.OrderDate AND s2.SalesOrderID <= s.SalesOrderID))
GROUP BY s.CustomerID, s.OrderDate, s.SalesOrderID, s.TotalDue
--ORDER BY s.CustomerID, s.OrderDate, s.SalesOrderID;
waitfor delay '00:05:00'
SELECT s.CustomerID, s.OrderDate, s.SalesOrderID, s.TotalDue, SUM(s2.TotalDue) AS RunningTotal
FROM Sales.SalesOrderHeader AS s INNER JOIN Sales.SalesOrderHeader AS s2
ON s2.CustomerID = s.CustomerID AND( s2.OrderDate < s.OrderDate OR( s2.OrderDate = s.OrderDate AND s2.SalesOrderID <= s.SalesOrderID))
GROUP BY s.CustomerID, s.OrderDate, s.SalesOrderID, s.TotalDue
--ORDER BY s.CustomerID, s.OrderDate, s.SalesOrderID;
commit tran
    
```

	CustomerID	OrderDate	SalesOrderID	TotalDue	RunningTotal
1	1	2001-08-01 00:00:00.000	43860	14603.7393	14603.7393
2	1	2001-11-01 00:00:00.000	44501	26128.8674	40732.6067
3	1	2002-02-01 00:00:00.000	45283	37643.1378	78375.7445
4	1	2002-05-01 00:00:00.000	46042	34722.9906	113098.7351
5	2	2002-08-01 00:00:00.000	46976	10184.0774	10184.0774
6	2	2002-11-01 00:00:00.000	47997	5469.5941	15653.6715
7	2	2003-02-01 00:00:00.000	49054	1739.4078	17393.0793
8	2	2003-05-01 00:00:00.000	50216	1935.5166	19328.5959
9	2	2003-08-01 00:00:00.000	51728	3905.2547	23233.8506
10	2	2003-11-01 00:00:00.000	57044	4537.8484	27771.699
11	2	2004-02-01 00:00:00.000	63198	4053.9506	31825.6496

	CustomerID	OrderDate	SalesOrder...	TotalDue	RunningTotal
1	1	2001-08-01 00:00:00.000	43860	14603.7393	14603.7393
2	1	2001-08-01 00:00:00.000	75136	14603.7393	29207.4786
3	1	2001-11-01 00:00:00.000	44501	26128.8674	55336.346
4	1	2001-11-01 00:00:00.000	75137	26128.8674	81465.2134
5	1	2002-02-01 00:00:00.000	45283	37643.1378	119108.3512
6	1	2002-02-01 00:00:00.000	75138	37643.1378	156751.489
7	1	2002-05-01 00:00:00.000	46042	34722.9906	191474.4796
8	1	2002-05-01 00:00:00.000	75139	34722.9906	226197.4702
9	2	2002-08-01 00:00:00.000	46976	10184.0774	10184.0774
10	2	2002-11-01 00:00:00.000	47997	5469.5941	15653.6715
11	2	2003-02-01 00:00:00.000	49054	1739.4078	17393.0793

Phantom read User3

```
--User 3 Admin
SELECT @@VERSION
SELECT @@SPID
use master
select distinct c.name, a.* from sysprocesses a, sysprocesses b, sysdatabases c where ( a.blocked <>0 or
(a.blocked = 0 and a.spid = b.blocked) ) and a.dbid = c.dbid order by a.spid
select * from sysprocesses where blocked <> 0

/*
use AdventureWorks
begin tran
delete from Sales.SalesOrderHeader
where SalesOrderID in
(
SELECT TOP (4) SalesOrderID
FROM Sales.SalesOrderHeader
WHERE (CustomerID = 1) order by SalesOrderID desc
)
--rollback tran
commit tran

SELECT CustomerID, * FROM [AdventureWorks].[Sales].[SalesOrderHeader] where CustomerID = 1
*/
```

Select blocking update User1

```
Microsoft SQL Server 2005 - 9.00.4266.00 (X64) Oct 7 2009 17:38:17 Copyright (c) 1988-2005 Microsoft Corporation Enterprise Edition (64-bit) on Windows NT 5.2 (Build 3790: Service Pack 2)
```

```
dbcc dropcleanbuffers
```

```
dbcc freeproccache
```

```
--User 1 select @@spid
```

```
Use AdventureWorks
```

```
SELECT SUM([StatusOid]-[PreviousStatusOid]) AS
```

```
ChangeHands, [Oid], [Type], [StatusOid], [PreviousStatusOid], [Date] FROM
```

```
[AdventureWorks].[Operation].[XX_StatusChangeHistory] WITH (TABLOCK)
```

```
Group by [Oid], [Type], [StatusOid], [PreviousStatusOid], [Date] order by [Oid], [Type]
```

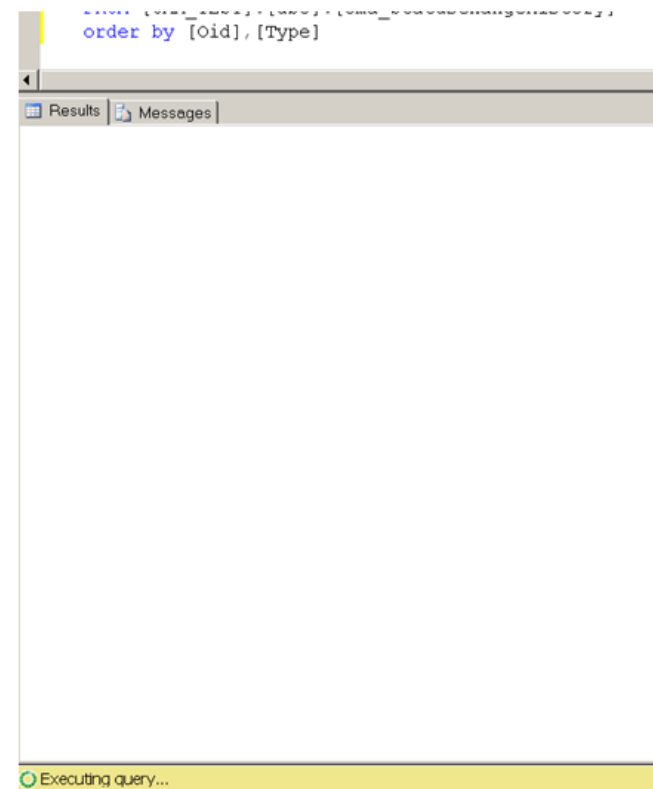
	ChangeHands	Oid	Ty...	Status...	PreviousStatus...	Date
1	0	1	2	7	7	2010-05-25 17:06:59.460
2	0	1	2	8	8	2006-09-12 13:48:13.210
3	1	1	2	9	8	2007-03-30 11:12:19.013
4	-2	101	1	1	3	2007-02-03 01:04:38.513
5	-1	103	1	1	2	2007-07-17 01:40:08.717
6	-4	103	1	2	6	2007-02-03 01:04:38.623
7	3	103	1	6	3	2006-10-12 11:19:51.223
8	3	108	1	6	3	2006-10-12 11:20:09.907
9	0	110	1	3	3	2006-08-07 00:32:18.737
10	2	110	1	5	3	2006-06-07 00:33:17.097
11	-2	112	1	3	5	2006-08-02 12:06:30.480
12	-1	112	1	5	6	2006-10-12 13:47:56.840
13	3	112	1	6	3	2006-10-12 11:23:14.647
14	0	118	1	3	3	2006-06-02 21:35:22.087
15	2	118	1	5	3	2006-06-15 18:48:39.890
16	1	118	1	6	5	2006-10-11 13:36:18.740
17	3	119	1	6	3	2006-10-12 11:20:10.110
18	3	121	1	6	3	2006-10-18 13:17:45.917
19	3	122	1	6	3	2006-10-12 11:20:10.300
20	-2	127	1	3	5	2006-05-31 21:01:57.110
21	2	127	1	5	3	2006-06-02 18:47:42.527
22	0	131	1	3	3	2006-05-17 01:12:41.723
23	3	131	1	6	3	2006-10-12 11:23:14.833
24	0	133	1	3	3	2006-08-01 21:57:57.503
25	0	135	1	3	3	2006-07-12 13:40:30.910

Executing query...

Select blocking update User2

```
--User2
select @@spid
update [AdventureWorks].[Operation].[xx_StatusChangeHistory]
set [StatusOid] =4 where [Oid]=101 and [Type]=1

SELECT [Oid],[Type],[StatusOid],[PreviousStatusOid],[Date] FROM
[AdventureWorks].[Operation].[xx_StatusChangeHistory] order by [Oid],[Type]
```



Select blocking update User3

```
--User3→admin
select distinct c.name, a.* from master.sys.sysprocesses a, master.sys.sysprocesses b,
master.sys.sysdatabases c where ( a.blocked <>0 or (a.blocked = 0 and a.spid = b.blocked) ) and a.dbid =
c.dbid order by a.spid
select * from master.sys.sysprocesses where blocked <> 0
exec sp_whoisactive
```

name	spid	kpid	block...	waittype	waittime	lastwaittype	waitresource	dbid	uid	cpu	physical...	memusage	login_time
ADMINDB	85	2420	0	0x0063	15	ASYNC_NETWORK_IO		18	1	11203	861	3	2010-11-12 14:25:47.410
ADMINDB	90	3952	85	0x0008	3312	LCK_M_IX	TAB: 43:1453248232:0	18	1	4891	13	3	2010-11-12 14:25:48.370

spid	kpid	block...	waittype	waittime	lastwaittype	waitresource	dbid	uid	cpu	physical...	memusage	login_time	last_batch
90	3952	85	0x0008	3515	LCK_M_IX	TAB: 43:1453248232:0	18	1	4891	13	3	2010-11-12 14:25:48.370	2010-11-12 14:30:44.35

dd hh:mm:ss.mss	session...	sql_text	login_name	wait_info	CPU	tempdb_writ...	tempdb_cur
00 00:00:07.246	85	<?query -- SELECT SUM([StatusOid]-[PreviousStatu...	NA02>wongj	(16ms)ASYNC_NETWORK_IO	500	0	0
00 00:00:03.970	90	<?query -- update [< Test].[dbo].[< StatusC...	NA02>wongj	(3907ms)LCK_M_IX		0	0

name	spid	kpid	block...	waittype	waittime
ADMINDB	85	2420	0	0x0063	15
ADMINDB	90	3952	85	0x0008	3312

(Enlarged view)

spid	kpid	block...	waittype	waittime	lastwaittype
90	3952	85	0x0008	3515	LCK_M_IX

- ▶ WITH (TabLockx) Microsoft SQL Server 2008 (SP1) – 10.0.2789.0 (X64)
- ▶ Microsoft SQL Server 2008 (SP2) – 10.0.3798.0 (Intel X86)

Select blocking update – Quiz

```
--User 1|
begin tran
SELECT rate, PayFrequency,
      [EmployeeID]
  FROM [AdventureWorks].[HumanResources].[EmployeePayHistory] with (ROWLOCK)
 where EmployeeID = 1
  waitfor delay '00:05:00'
SELECT rate, PayFrequency,
      [EmployeeID]
  FROM [AdventureWorks].[HumanResources].[EmployeePayHistory] with (ROWLOCK)
 where EmployeeID = 1
commit tran
--rollback tran

--User 2
select @@version

begin tran
update [AdventureWorks].[HumanResources].[EmployeePayHistory]
set Rate = 12.45
where EmployeeID = 1
commit tran
--rollback tran
```

- ▶ with (Rowlock) Microsoft SQL Server 2008 ?

Improve Performance

70% - 80% on code efficiency

20% - 30% on server configuration

Today's topic:

Why my query so slow?

Help Optimizer to help your query

Execution Plan, Query Tuning

Indexes

Statistics

CTE, Temp Table, ~~Cursor~~

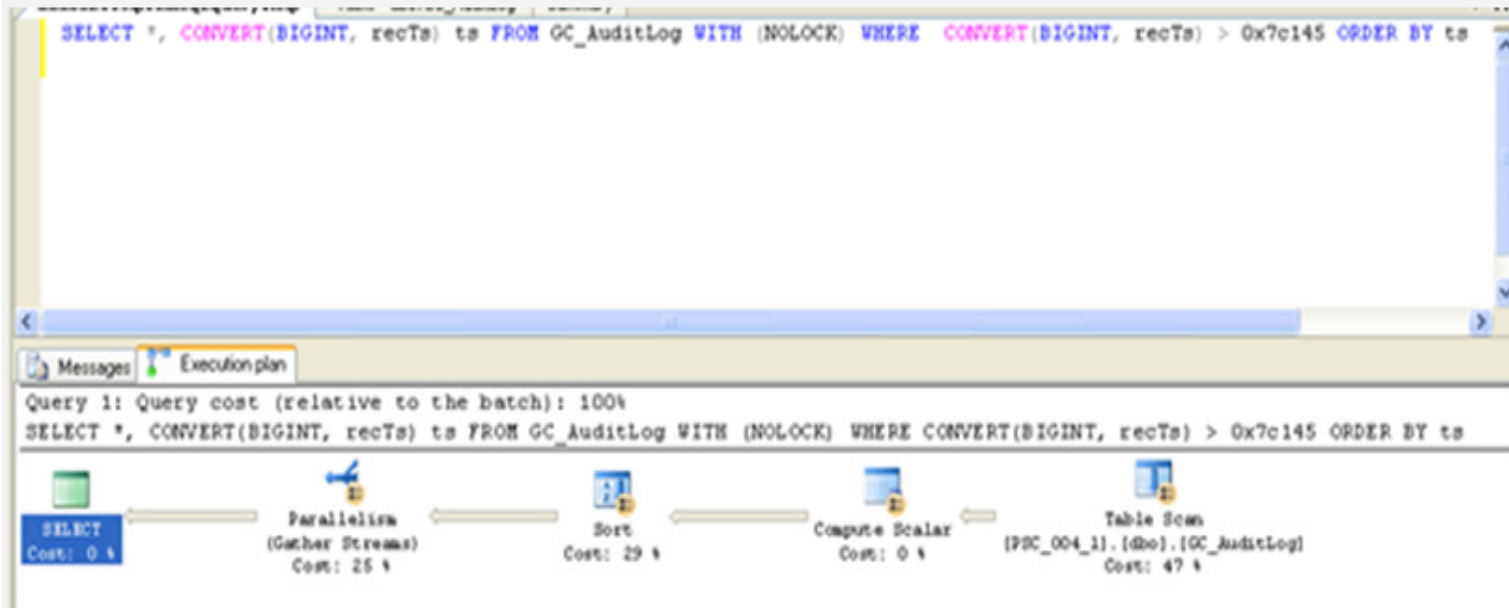
Next

»» Code Efficiency
Query Tuning
....Take a deep breath first

Query performance tuning

Query tuning

```
>SELECT *, CONVERT(BIGINT, recTs) ts FROM AuditLog WITH (NOLOCK) WHERE CONVERT(BIGINT, recTs) > 0x7c145 ORDER BY ts
```



- [-] [-] **dbo.GC_AuditLog**
 - [-] Columns
 - [-] auditId (bigint, null)
 - [-] auditDateTime (datetime, not null)
 - [-] rxclasstype (varchar(250), not null)
 - [-] kind (int, not null)
 - [-] recKey (int, null)
 - [-] data1 (int, null)
 - [-] data2 (varchar(250), null)
 - [-] data3 (float, null)
 - [-] data4 (float, null)
 - [-] userName (varchar(250), not null)
 - [-] modifiedInfo (varchar(250), null)
 - [-] recTs (timestamp, not null)
 - [-] Keys
 - [-] Constraints
 - [-] CK__GC_AuditLo__kind__09DE7BCC
 - [-] DF__GC_AuditL__audit__08EA5793
 - [-] DF__GC_AuditL__userN__0AD2A005
 - [-] Triggers
 - [-] Indexes
 - [-] IDX_GC_AuditLog_auditDateTime (Non-Unique, Non-Clustered)
 - [-] IDX_GC_AuditLog_rxclasstype (Non-Unique, Non-Clustered)
 - [-] Statistics
 - [-] _WA_Sys_00000004_07F6335A
 - [-] _WA_Sys_00000005_07F6335A
 - [-] _WA_Sys_0000000A_07F6335A
 - [-] _WA_Sys_0000000C_07F6335A
 - [-] IDX_GC_AuditLog_auditDateTime
 - [-] IDX_GC_AuditLog_rxclasstype

```
SELECT *, CONVERT(BIGINT, recTs) ts FROM AuditLog WITH (NOLOCK) WHERE CONVERT(BIGINT, recTs) > 0x7c145 ORDER BY ts
```

```
SELECT *, CONVERT(BIGINT, recTs) FROM dbo.AuditLog WITH (NOLOCK) WHERE recTs > CONVERT(BIGINT, 0x7c145) ORDER BY recTs
```

```
SELECT recTs FROM dbo.AuditLog WITH (NOLOCK) WHERE recTs > CONVERT(BIGINT, 0x7c145) ORDER BY recTs
```

```
SELECT recTs FROM dbo.AuditLog WITH (NOLOCK) WHERE recTs > CONVERT(BIGINT, 0x7c145)
```

```
SELECT recTs FROM dbo.GC_AuditLog WITH (NOLOCK, index(IDX_GC_AuditLog_recTs)) WHERE recTs = 508229
```

```

SELECT *, CONVERT(BIGINT, recTs) ts FROM dbo.GC_AuditLog WITH (NOLOCK) WHERE CONVERT(BIGINT, recTs) > 0x7c145 ORDER BY ts
SELECT *, CONVERT(BIGINT, recTs) ts FROM dbo.GC_AuditLog WITH (NOLOCK) WHERE recTs > CONVERT(BIGINT, 0x7c145) ORDER BY ts
SELECT *, CONVERT(BIGINT, recTs) FROM dbo.GC_AuditLog WITH (NOLOCK) WHERE recTs > CONVERT(BIGINT, 0x7c145) ORDER BY recTs
SELECT recTs FROM dbo.GC_AuditLog WITH (NOLOCK, index(IDX_GC_AuditLog_recTs)) WHERE recTs = 508229
SELECT recTs FROM dbo.GC_AuditLog WITH (NOLOCK, index(IDX_GC_AuditLog_recTs)) WHERE recTs = 0x7c145

```

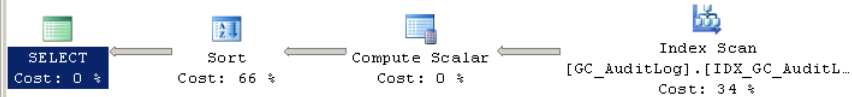
Results Messages Execution plan

Query 1: Query cost (relative to the batch): 50%

```

SELECT *,CONVERT([bigint],[recTs],0) [ts] FROM [dbo].[GC_AuditLog] WITH(nolock,readuncommitted) WHERE CONVERT([bigint],[recTs],0)>@1 ORDER BY [ts] ASC

```

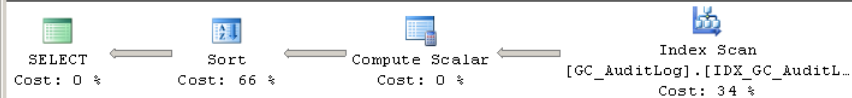


Query 2: Query cost (relative to the batch): 50%

```

SELECT *,CONVERT([bigint],[recTs],0) [ts] FROM [dbo].[GC_AuditLog] WITH(nolock,readuncommitted) WHERE [recTs]>CONVERT([bigint],@1,0) ORDER BY [ts] ASC

```

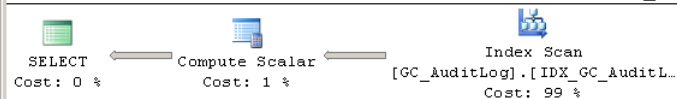


Query 3: Query cost (relative to the batch): 15%

```

SELECT *,CONVERT([bigint],[recTs],0) FROM [dbo].[GC_AuditLog] WITH(nolock,readuncommitted) WHERE [recTs]>CONVERT([bigint],@1,0) ORDER BY [recTs] ASC

```

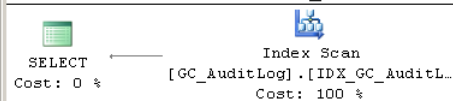


Query 4: Query cost (relative to the batch): 13%

```

SELECT recTs FROM dbo.GC_AuditLog WITH (NOLOCK, index(IDX_GC_AuditLog_recTs)) WHERE recTs = 508229

```

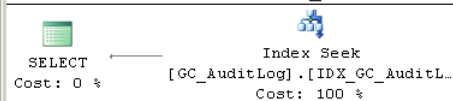


Query 5: Query cost (relative to the batch): 0%

```

SELECT recTs FROM dbo.GC_AuditLog WITH (NOLOCK, index(IDX_GC_AuditLog_recTs)) WHERE recTs = 0x7c145

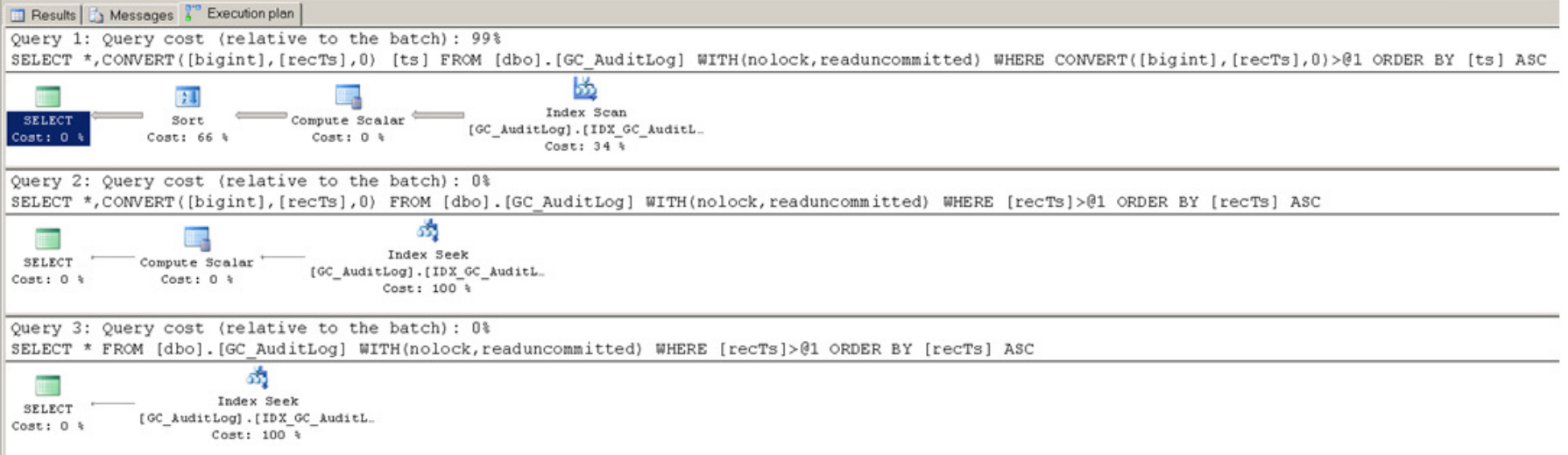
```




```

SELECT *, CONVERT(BIGINT, recTs) ts FROM dbo.GC_AuditLog WITH (NOLOCK) WHERE CONVERT(BIGINT, recTs) > 0x7c145 ORDER BY ts
SELECT *, CONVERT(BIGINT, recTs) FROM dbo.GC_AuditLog WITH (NOLOCK) WHERE recTs > 0x7c145 ORDER BY recTs
SELECT * FROM dbo.GC_AuditLog WITH (NOLOCK) WHERE recTs > 0x7c145 ORDER BY recTs

```



From 11-12 seconds to <1 seconds.

Re-write:

```

>SELECT *, CONVERT(BIGINT, recTs) ts FROM dbo.AuditLog WITH (NOLOCK) WHERE CONVERT(BIGINT, recTs) > 0x7c145
ORDER BY ts OPTION (FORCE ORDER)

```

```

>SELECT *, CONVERT(BIGINT, recTs) ts FROM dbo.AuditLog WITH (NOLOCK) WHERE COALESCE(CONVERT(BIGINT, recTs),
CONVERT(BIGINT, recTs)) > 0x7c145 ORDER BY ts OPTION (FORCE ORDER)

```

```

>SELECT *, CONVERT(BIGINT, recTs) ts FROM dbo.AuditLog WITH (NOLOCK) WHERE CONVERT(BIGINT, recTs) >
COALESCE(0x7c145, 0x7c145) ORDER BY ts

```

CASE 1 - inline function, expression

A Search predicate contains a column expression in a function.

It is recommended that you remove the encapsulating function, allowing the column expression to be considered by the query optimizer without affecting the result.

The query optimizer will not be able to perform an index seek on search predicates that include the column in a function, resulting in a less optimal execution plan.

Example,

Change:

```
SELECT OrderID FROM myDB.dbo.Orders WHERE DATEADD(day, 15, OrderDate) = '07/23/1996'
```

To:

```
SELECT OrderID FROM myDB.dbo.Orders WHERE OrderDate = DATEADD(day, -15, '07/23/1996')
```

Example,

```
INSERT INTO @Invoices
SELECT pid.InvoiceNumber, SUM(TotCost) Sump, Value - SUM(TotCost) Pickup, Value
FROM ABC PreviewInvoiceDetail pid INNER JOIN ABC PreviewInvoices pi on pi.InvoiceNumber = pid.InvoiceNumber
LEFT OUTER JOIN ABC ServiceOrders so ON so.ServiceOrderNumber = pid.SOrder
LEFT OUTER JOIN ABC CorporateServiceTypes cst ON cst.Oid = so.ServiceTypeOid
WHERE
(
    (Profile LIKE 'homesump%' OR Profile LIKE 'homeflash%') OR ISNULL(cst.ServiceType, '') = 'Sump-
Vacuum'
)
```

The use of the LEFT function in the search predicate may prevent the query optimizer from generated execution plans that use index seeks as well as from effectively evaluating string statistics.

The function or expression "substring" in a query can cause index suppression resulting in poor performance due to a scan being performed instead of a seek.

Hard-coded date is bad unless there is a reason that cannot be overcome.

Example,

```
From ABC.dbo.Node Where iParentNodeId = 10011 And IsNumeric(SubString(sdescription,0,4)) = 1
```

Example,

```
select count(*) from dbo.[pv_Profile] where customer=c.customer and datediff(d,cast(begdate as datetime),'06/30/2010')<31  
and beguser in (select userid from dbo.pv_User where [site]='PRO')  
and left(prftype,2)='LP'
```

The query optimizer will not be able to perform an index seek on search predicates that include the column in a function, resulting in a less optimal execution plan.

Example,

```
Case  
WHEN isdate(c.begdate)=1 THEN  
    Case  
    When datediff(d,cast(c.begdate as datetime),'06/30/2010')<31 Then 'Y'  
    Else 'N'  
    End  
Else 'N'
```

Example,

```
[C_Date_Opened]=  
CASE  
WHEN isdate(c.begdate)=1 THEN c.begdate  
ELSE NULL  
END
```

Example,

```
SELECT @V = MIN(SickLeaveHours)+1 FROM HumanResources.Employee  
SELECT @V = MIN(SickLeaveHours)+1 FROM HumanResources.Employee  
IF ISNULL(@WorkOrder,"") <> ""
```

Quiz (people knows how to waste computer time)

Example1: what is wrong with this code?

```
SET NOCOUNT ON
```

```
DECLARE @rowname varchar(255), @rc int
```

```
SET @rc = 1
```

```
SELECT TOP 1 @rowname = CONVERT(varchar(255), OID)
```

```
FROM dbo.WO_WorkOrderDetail order by OID
```

```
WHILE @rc <> 0
```

```
BEGIN
```

```
    SELECT TOP 1 @rowname = CONVERT(varchar(255), OID)
```

```
    FROM dbo.WO_WorkOrderDetail
```

```
    WHERE OID > @rowname
```

```
    ORDER BY OID
```

```
    SET @rc = @@ROWCOUNT
```

```
SELECT count(*), OID from wo_LatestFile latest (nolock) inner join wo_WorkOrderDetail  
(nolock)
```

```
on LatestFile = FileHistoryID WHERE OID = @rowname GROUP by OID
```

```
END
```

SQL
saturday!



Example2

```
INSERT INTO #tmpMinMaxSDs4SO
SELECT B.Oid, NULL, NULL FROM Bills B WITH (NOLOCK)
WHERE B.Oid NOT IN(SELECT isnull(BillOid,0) FROM #tmpMinMaxSDs4SO)
AND ISNULL(B.IsDeleted,0) = 0
```

Example3

```
@@IDENTITY
IDENT_CURRENT
SCOPE_IDENTITY
```

Example4

```
select * from (
select count(*) as TotalInstancesOfThisSourceSystemItemID, SourceSystemItemID from tktit
where sourcesystemitemid > 0 and sourcesystemid is not null and SourceSystemItemID <> '922389'
group by sourcesystemitemid
) as tbl
where TotalInstancesOfThisSourceSystemItemID > 1
```

CASE 2 - select *, Order by, Group by

Using SELECT * can result in significant performance overhead if the associated database application does not require all columns from tables and views used in the where clause of the query.

Example,

```
SELECT SR.* FROM SurveyResponse As SR ...
```

Example,

```
exec [sys].sp_bcp_dbcmptlevel [ABC_Staging] set fmtonly on  
select * from [dbo].[pvusermf] set fmtonly off
```

Example,

- Insert #tempTbl Select ... Order by

CASE 3 - compiled incorrect plan

Inefficient cached plan selected due to abnormal parameters.

It is recommended using the RECOMPILE query hint so the execution plans are not cached.

The significant difference between execution times and consumed resources likely indicates that an unexpected set of parameters was used resulting in an inefficient cached plan being selected.

Query hint Option:

```
Example, CREATE PROC usp_city_search
          @OrderID
AS
          ..... -- code to validate @OrderID

CASE
--When @ OrderID is not null and 9 digits,
Where OrderID = @OrderID ....
Where OrderID like @OrderID
--When @ OrderID is null,
Where OrderID like '%' ....
Where OrderID = '%' ....
--When @ OrderID is not null and less than 9 digits,
Where OrderID like @OrderID + '%' ....
          12%

--User
EXEC usp_city_search '123456789'
EXEC usp_city_search ''
EXEC usp_city_search '12%'
```

1)
OPTION (RECOMPILE);
OPTION (OPTIMIZE FOR (@OrderID = '123456789',));

2)

EXEC sp_create_plan_guide
@name = N'Guide7',
@stmt = N'SELECT c.LastName, c.FirstName, e.Title
FROM HumanResources.Employee AS e
WITH (NOLOCK, INDEX (PK_Employee_EmployeeID))
JOIN Person.Contact AS c ON e.ContactID = c.ContactID
WHERE e.ManagerID = 2;',
@type = N'SQL',
@module_or_batch = NULL,
@params = NULL,
@hints = N'OPTION (TABLE HINT (e, NOLOCK))';
GO

3)

EXEC sp_create_plan_guide
@name = N'Guide6',
@stmt = N'SELECT c.LastName, c.FirstName, e.Title
FROM HumanResources.Employee AS e
WITH (NOLOCK, INDEX (PK_Employee_EmployeeID))
JOIN Person.Contact AS c ON e.ContactID = c.ContactID
WHERE e.ManagerID = 3;',
@type = N'SQL',
@module_or_batch = NULL,
@params = NULL,
@hints = N'OPTION (TABLE HINT (e, INDEX (IX_Employee_ManagerID) , NOLOCK, FORCESEEK))';
GO

4)

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'optimize for ad hoc workloads', 1;
GO
RECONFIGURE;
GO
```

Example,

```
exec [ABC].[dbo].[pValidateUser] @OrgRefId1 uniqueidentifier, @CustomerName2 varchar(11), @OrgRefId3
uniqueidentifier, @orgrefid=@p4 output, @userrefid=@p5 output, @isMobileOnly=@p6 output
select @p4, @p5, @p6
```

...

...

```
SELECT ... FROM [ABC].[dbo].[FuelOrder] WHERE ( ( [ABC].[dbo].[FuelOrder].[OrgRefID] = @OrgRefId1 AND
[ABC].[dbo].[FuelOrder].[RTA] IS NOT NULL AND [ABC].[dbo].[FuelOrder].[TripFromToDesc] IS NULL AND
[ABC].[dbo].[FuelOrder].[CustomerName] LIKE @CustomerName2 AND [ABC].[dbo].[FuelOrder].[OrgRefID] =
@OrgRefId3))
```

CASE 4 - recompile plan

Saved settings options may prohibit further query optimizations.

It is recommended that you evaluate these options settings. If the behavior associated with these settings isn't specifically required for the function of this procedure, the procedure should be recreated using a session that has both these options set to ON.

Although these settings may not relate to current performance problems, these settings can prohibit further performance optimizations, such as indexed (materialized) views.

```
SET ANSI_NULLS ON
SET QUOTED_IDENTIFIER ON
```

Example,

```
USE [ABC]
GO
SET ANSI_NULLS OFF
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE PROCEDURE dbo.TktCustom_GetByRowguid
( @Rowguid uniqueidentifier )
AS
```

Example,

```
USE [ABC]
GO
SET ANSI_NULLS OFF
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE PROCEDURE dbo.TktPriceList_Get
( @WhereClause varchar (2000), @OrderBy varchar (2000) )
AS
```

CASE 5 - union

If the likelihood of duplicate rows in the result set is low, the query should be rewritten to replace the UNION operator with UNION ALL.

```
From : -----
INSERT INTO #tmpMinMaxSDs4SO
SELECT DISTINCT S.BillOid, MIN(S.ServiceDate), MAX(S.ServiceDate)
FROM ShiftOrders S WITH (NOLOCK)
Where
    Old in (SELECT DISTINCT ShiftorderOid FROM BillItems BI WITH (NOLOCK) WHERE BI.IsDeleted=0) AND
    S.IsDeleted = 0 AND S.Oid > 100
GROUP BY S.BillOid
/*****/
INSERT INTO #tmpMinMaxSDs4SO
Select
    *
From
(
    SELECT DISTINCT S.BillOid, MIN(S.ServiceDate) as MinServiceDate, MAX(S.ServiceDate) as MaxServiceDate
    FROM ShiftOrders S WITH (NOLOCK)
    Left Outer Join Bills B WITH (NOLOCK) on B.Oid=S.BillOid
    WHERE B.IsDeleted = 0 AND B.BillTypeOid = 5
    GROUP BY S.BillOid
)X
Where
    BillOid NOT IN (SELECT IsNull(BillOid,0) FROM #tmpMinMaxSDs4SO)
```

```

Change to: /*****/
INSERT INTO #tmpMinMaxSDs4SO
SELECT DISTINCT S.BillOid, MIN(S.ServiceDate), MAX(S.ServiceDate)
FROM oma_ShiftOrders S WITH (NOLOCK)
Where
    Old in (SELECT DISTINCT ShiftorderOid FROM BillItems BI WITH (NOLOCK) WHERE BI.IsDeleted=0) AND
    S.IsDeleted = 0 AND S.Oid > 100
GROUP BY S.BillOid
/*****/
UNION --INSERT INTO #tmpMinMaxSDs4SO
--Select * From
--(
SELECT DISTINCT
    S.BillOid, MIN(S.ServiceDate) as MinServiceDate, MAX(S.ServiceDate) as MaxServiceDate
FROM ShiftOrders S WITH (NOLOCK)
Left Outer Join oma_Bills B WITH (NOLOCK) on B.Oid=S.BillOid
WHERE
    B.IsDeleted = 0 AND B.BillTypeOid = 5
GROUP BY S.BillOid
--)X
--Where BillOid NOT IN (SELECT IsNull(BillOid,0) FROM #tmpMinMaxSDs4SO)
/*****/

```

Results: from 6 minutes 49 seconds to 9 seconds, that is "4544%" better.

CASE 6 - missing join

It is recommended that you include all appropriate join columns in the underlying tables in the join predicate.

The query is missing a join predicate that defines the relationship between the tables used in the query. Without a join predicate, the result will include the Cartesian product of all rows, resulting in significant performance overhead.

Example,

```
-- Update the existing ABC Invoice Record
UPDATE ABC_invoices
  SET modifieddate = Getdate(), modifiedby = 'System',
  --InvoiceNumber = pi.InvoiceNumber,
  invoiceamt = pi.VALUE, invoicedate = pi.invoicedate
FROM   ABC_previewinvoices pi --INNER JOIN ABC InvoiceHeader rih ON
  --pi.InvoiceNumber = rih.PreviewMainInvoiceNumber
INNER JOIN ABC_woheader woh
  ON pi.ordernumber = woh.wonumber
WHERE  workorderoid = @WorkOrderOid
AND    pi.invoicenumber = @InvoiceNumber
```

CASE 7 - type implicit conversion, nested view

An implicit conversion on column may be causing index suppression.
It is recommended that you consider redesigning this WHERE clause to prevent index suppression and performance degradation.

Example,

```
SELECT ... FROM [dbo].[ABC] AS [Extent1]  
WHERE [Extent1].[Status] = @Status
```

Example,

```
SELECT ... FROM ... Inner join on Tab1.ColumnA = Tab2.ColumnB
```

CASE 8 - unfiltered delete

An unfiltered delete, it is recommended that full truncations of this type should be re-written to use the much more efficient TRUNCATE statement instead of DELETE. Note: delete keeps the value of identity column, truncate renews it. TRUNCATE TABLE is faster and uses fewer system and transaction log resources.

Example,

```
DELETE FROM ABC_previewinvoiceheaderstage
```

CASE 9 – like operator, index selectivity

LIKE is used for string comparison with no wild cards.

The use of the LIKE operator in the search predicate with no wild card is equivalent to using the equal string comparison operator. Using the string equal operator will likely result in a more optimal execution plan.

It is recommended that you use the string equal operator in place of the LIKE operator.

Example,

```
SELECT @Path = Path FROM ABC_serverPath
WHERE
    ServerName=@@Servername AND
    module like 'Preview-Generator' AND Isdeleted=0
```

The function "like" may be causing a table scan.

It is recommended that you consider redesigning this WHERE clause to prevent index suppression and performance degradation.

The function or expression "like" in a query can cause index suppression resulting in poor performance due to a scan being performed instead of a seek.

Example,

```
SELECT ... FROM [ABC].[dbo].[FuelOrder]
WHERE (( [ABC].[dbo].[FuelOrder].[OrgRefID] = @OrgRefId1 AND [ABC].[dbo].[FuelOrder].[RTA] IS NOT NULL
AND [ABC].[dbo].[FuelOrder].[TripFromToDesc] IS NULL
AND [ABC].[dbo].[FuelOrder].[CustomerName] LIKE @CustomerName2
AND [ABC].[dbo].[FuelOrder].[OrgRefID] = @OrgRefId3))
```

Example of using multi-status code:

```
SELECT name FROM .... WHERE name = 'xxxx'  
  
AND (status & 32) = 0 -- Do not include  
  
AND (status & 64) = 0 -- Do not include  
  
AND (status & 128) = 0 -- Do not include  
  
AND (status & 256) = 0 -- Do not include  
  
AND (status & 512) = 0 -- Do not include  
  
AND (status & 32768) = 0 -- Do not include  
  
AND (status & 1073741824) = 0 -- Do not include
```


CASE 10 – missing, unused, duplicate indexes

Execution plan

Set statistics IO on
Set statistics Time on
Set SHOWPLAN_TEXT on
Set SHOWPLAN_ALL on

To find missing, unused, duplicate indexes:

<http://usa.redirectme.net/repriser/sqlserverpub.html>

Missing Indexes	Duplicate Indexes	Unused Indexes
sys.dm_db_missing_index_group_stats	sys.index_columns	sys.dm_db_index_usage_stats
sys.dm_db_missing_index_groups	sys.indexes	sys.indexes
sys.dm_db_missing_index_details	sys.dm_db_index_usage_stats	sys.objects
	sys.foreign_keys	sys.schemas

Other useful DMVs, DMFs

[sys.dm_exec_cached_plans](#)
[sys.dm_exec_requests](#)
[sys.dm_exec_query_memory_grants](#)
[sys.dm_exec_query_stats](#)
[sys.dm_exec_cursors](#)
[sys.dm_exec_xml_handles](#)

CASE 11 - query hint

Query hint abuse has been detected.

It is recommended to evaluate the use query hints periodically for performance benefit to determine if the hint still provides the original performance benefit.

The following query hints were detected: (TabLockX). Query hints are used to affect the execution plan or enforce a locking method. Over time, the underlying reason for using the hint may change, resulting in the hint causing performance problems rather than improving performance.

Example,

```
DELETE TOP (@PermanentSnapshotCount) SnapshotData
output deleted.SnapshotDataID into @cleanedSnapshots (SnapshotDataId)
FROM SnapshotData with(readpast)
WHERE SnapshotData.PermanentRefCount = 0 AND
       SnapshotData.TransientRefCount = 0 ;
SET @SnapshotsCleaned = @@ROWCOUNT ;
```

Example,

```
declare @BatchID uniqueidentifier
set @BatchID = NEWID()
UPDATE [Event] WITH (TABLOCKX)
SET [BatchID] = @BatchID, [ProcessStart] = GETUTCDATE(), [ProcessHeartbeat] = GETUTCDATE()
FROM ( SELECT TOP 8 [EventID] FROM [Event] WITH (TABLOCKX)
       WHERE [ProcessStart] is NULL ORDER BY [TimeEntered]
) AS t1 WHERE [Event].[EventID] = t1.[EventID]
```

CASE 12 - locking, blocking

The resource locking results in blocking, with shared lock, exclusive lock etc.

ACID

Isolation level:

Default: Read committed

Read uncommitted, A.K.A. "With (nolock)"

Snapshot

.....

Isolation level	Dirty read	Nonrepeatable read	Phantom
Read uncommitted	Yes	Yes	Yes
Read committed	No	Yes	Yes
Repeatable read	No	No	Yes
Snapshot	No	No	No
Serializable	No	No	No

CASE 13 - in, not in, exists, not exists

Not in can be accomplished with outer join.

Example,

Change this block : `INSERT INTO #tmpMinMax SELECT DISTINCT S.BillOid, MIN(S.servicedate),MAX(S.servicedate) FROM orders S WITH (NOLOCK) left outer join Bills B WITH (NOLOCK) on B.Oid=S.BillOid WHERE B.Oid NOT IN (SELECT isnull(BillOid,0) FROM #tmpMinMax) AND ISNULL(B.IsDeleted,0) = 0 AND (B.billtypeid=5) GROUP BY S.BillOid`

To the following block : `INSERT INTO #tmpMinMax SELECT DISTINCT S.BillOid, MIN(S.servicedate),MAX(S.servicedate) FROM orders S WITH (NOLOCK) left outer join Bills B WITH (NOLOCK) on B.Oid=S.BillOid left outer join #tmpMinMax tem WITH (NOLOCK) on S.BillOid = tem.BillOid WHERE ISNULL(B.IsDeleted,0) = 0 AND (B.billtypeid=5) AND tem.BillOid is null GROUP BY S.BillOid`

This statement inside a stored procedure is reduced from 10-11 minutes to 6-8 seconds execution, i.e. improvement by 8250%.

Example,

```
--UPDATE ABC_Invoices SET PickupAmount = InvoiceAmt - SumpAmount WHERE CorporateCustomerOid = 103 and WorkOrderOid IS NOT NULL
update ABC_invoices set pickupamount = invoiceamt where oid in (
select oid from ABC_invoices
where corporatecustomeroid = 103
and invoiceamt <> 0
and pickupamount = 0
and sumpamount = 0
and isdeleted = 0
and invoicedate > '10/31/2009')
```

Example,

• Change the WHERE clause predicate on line 118 from:

`AND (Select Count(Oid) from ShiftReports WITH (NOLOCK) Where ShiftOrderOid = SO.Oid and ShiftReports.isdeleted=0) = 0`

• To:

```
AND ( NOT EXISTS ( SELECT [ShiftOrderOid] FROM [dbo].[ShiftReports] AS [ShiftReports] WHERE  
                ( ([ShiftReports].[ShiftOrderOid] = [SO].[Oid]) AND ([ShiftReports].[IsDeleted] = 0) ) ) ) )
```

CASE 14 – Object Naming Convention

4 part naming convention: Node.database.schema.object

Ambiguous naming, calling execution, may result to an engine initiated search, a recompile, which is just waste of time.

Naming objects with a word QA, Test, Dev, etc. in the name may cause human confusion.

Use of schema, schema-owner.

Example,

DBHost1.ABC.dbo.isp_runsomething

ABC.tiger.usp_dosomething – Database: ABC, Schema: tiger, stored procedure: usp_dosomething

Do not name with “sp_” prefix

Using Fully Qualify Database Objects, You minimize overhead for name resolution, and avoid potential schema locks and execution plan recompiles.

Foreign Keys, Indexes, objects should be named meaningfully.

FK example: FK_PKTable_FKTable_Col1Col2Col3

Index example: UIX_Table_Key1Key2_Inc1Inc2

CASE 15 - database version compatibility

The compatibility mode of database (80) is lower than the SQL Server version (90).

If you are unsure of the reasons this database is running in a lower compatibility mode, it is recommended that you test the impact of setting the compatibility mode to 90.

General Rules:

1. Create a **clustered index** (note that if you set the primary key in Enterprise Manager it will cluster it by default) on each table you create and unless you are really knowledgeable enough to figure out a better plan.
2. Create an index on any **column that is a foreign key**. If you **know it will be unique**, set the flag to force the index to be unique. **Index joined columns plus where-clause columns as key**, **include selected columns**.
3. Unless you need a different behavior, always **owner qualify your objects** when you reference them in TSQL. Use `dbo.sysdatabases` instead of just `sysdatabases`.
4. Use set **`nocount`** on at the top of each stored procedure (and set `nocount off` at the bottom).
5. Think hard about locking. If you're not writing banking software, would it matter that you take a chance on a dirty read? You can use the NOLOCK hint, but it's often easier to use SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED at the top of the procedure, then reset to READ COMMITTED at the bottom. Use **Snapshot isolation instead**.
6. I know you've heard it a million times, but **only return the columns and the rows you need**.
7. Use transactions when appropriate, but **allow zero user interaction while the transaction is in progress**. I try to do all my transactions inside a stored procedure.
8. Avoid temp tables as much as you can, but if you need a temp table, create it explicitly using **Create Table #temp**. Use CTE **instead**.
9. **Avoid NOT IN**, instead use a left outer join - even though it's often easier to visualize the NOT IN.
10. If you insist on using dynamic `sql` (executing a concatenated string), use named parameters and `sp_executesql` (rather than EXEC) so you have a chance of **reusing the query plan**. While it's simplistic to say that stored procedures are always the right answer, it's also close enough that you won't go wrong using them.
11. Get in the habit of **profiling your code** before and after each change. While you should keep in mind the depth of the change, if you see more than a 10-15% increase in CPU, Reads, or Writes it probably needs to be reviewed.
12. Look for every possible way to **reduce the number of round trips** to the server. Returning multiple result-sets is one way to do this. Reduce recursive looping.
13. **Avoid forced query hints**.
14. When you're done coding, set Profiler to monitor statements from your machine only, then run through the application from start to finish once. Take a **look at the number of reads and writes**, and the number of calls to the server. See anything that looks unusual? It's not uncommon to see calls to procedures that are no longer used, or to see duplicate calls. Impress your DBA by asking him to review those results with you.

SQLSaturday #57 Session Feedback

Please provide feedback on this session,

“Ask Why My Query So Slow?”

on SpeakerRate at: <http://spkr8.com/e/715>

or <http://SpeakerRate.com>

Search Events: SQL Saturday #57





Thank You, Sponsors!



CTREC
HILTON
IT ACADEMY



Texas Memory Systems, Inc.
The World's Fastest Storage®



redgate®
sponsored by Red Gate



SQL Saturday #57 Houston

»» Ask Why My Query So Slow?

Jason Wong

URL: <http://usa.redirectme.net>

Feedback on SpeakerRate:

<http://spkr8.com/e/715> or

<http://SpeakerRate.com>

Search Events: SQL Saturday #57