



SQL Server 2005/2008: Performance Tuning & Optimization

Student Lab Book

Version 1.3a

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. These materials are intended for distribution to and use only by Microsoft Premier Customers. Use or distribution of these materials by any other persons is prohibited without the express written permission of Microsoft Corporation. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

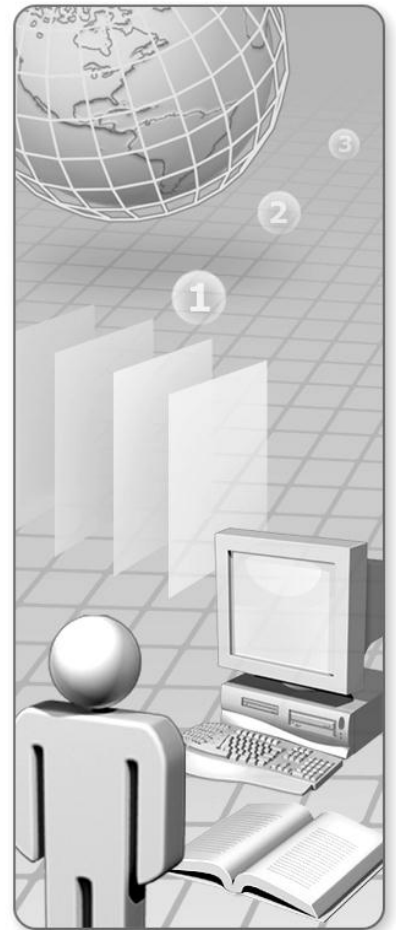
©2010 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, Windows, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

MODULE 1: ARCHITECTURE	3
LAB 1.....	5
EXERCISE 1: MEMORY.....	5
MODULE 2: TABLE AND INDEX STRUCTURE	11
LAB 1.....	13
EXERCISE 1: SCHEMA	13
EXERCISE 2: CLUSTERED INDEXES AND PAGE SPLITS	21
EXERCISE 3: USING DMV'S.....	27
EXERCISE 4: USING MISSING INDEXES XML SHOWPLAN.....	31
MODULE 3: PERFORMANCE TOOLS & MONITORING	35
LAB 1.....	37
EXERCISE 1: SERVER SIDE TRACING.....	37
EXERCISE 2: BLOCKED PROCESS REPORT	41
EXERCISE 3: GRAPHICAL SHOWPLAN XML	45
EXERCISE 4: SQLDIAG	47
LAB 2.....	49
EXERCISE 1: HIGH CPU REPORT	49
EXERCISE 2: MISSING INDEX REPORT	51
MODULE 4: LOCKING AND CONCURRENCY.....	53
LAB 1.....	55
EXERCISE 1: DMVs	55
EXERCISE 2: ISOLATION LEVEL.....	59
EXERCISE 3: DEADLOCKS.....	63
MODULE 5: QUERY OPTIMIZATION	65
LAB 1.....	67
EXERCISE 1: ASYNCHRONOUS UPDATE STATISTICS	67
EXERCISE 2: IDENTIFYING AND TUNING EXPENSIVE QUERIES IN THE CACHE.....	69
MODULE 7: RESOURCE GOVERNOR	75
LAB 1.....	77
EXERCISE 1: SETUP RESOURCE GOVERNOR FOR WORKLOADS	77

Module 1: Architecture



Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. These materials are intended for distribution to and use only by Microsoft Premier Customers. Use or distribution of these materials by any other persons is prohibited without the express written permission of Microsoft Corporation. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

©2010 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, Windows, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Lab 1

Exercise 1: Memory

Upon completing this exercise, you will be able to:

- Use DMVs and DBCC commands and performance monitor counters to analyze SQL Server memory.

Note: All scripts to run are in the c:\labs subdirectory with directories there for each module.

Run Scripts to query Memory Related DMV's

1. Execute bpool_by_db.sql.

Question A: What is the value returned for AdventureWorksPTO? Do any other databases consume more memory?

2. Execute memory_bpool.sql.

Question B: What is the value returned?

3. Execute bpool_vs_reserved_cache store.sql.

Question C: What is the value returned?

4. Execute memory_non_bpool.sql.

Question D: How much memory is in use besides the buffer pool?

5. Open a command prompt and navigate to C:\Program Files\Microsoft SQL Server\100\Tools\Binn . Run sqldiag -Isd_general.xml. Wait until you see the collection started message before running the next step.

6. In Management Studio, execute batch_one.sql, batch_two.sql, and batch_three.sql. Wait until all queries have completed and then hit Ctrl-C in the command window to stop the SQLDiag. It will take a few minutes. Maybe time to grab a drink.
7. Execute steps 1-4 and see if there has been any change to the buffer pool or reserved memory.

Question E: Which database consumes the majority of the buffer pool?

8. Click start run and type perfmon to launch Performance Monitor. Click on the View Log Data icon (disk looking icon) and navigate to C:\Program Files\Microsoft SQL Server\100\Tools\Binn\SQLDIAG and select the SQLDIAG.BLG file.
9. Click on the + and add the following counters and note the values:

Object	Instance	Counter
Process	sqlservr	Private Bytes
	sqlservr	Virtual Bytes
Memory		Available Mbytes
SQL Server Memory Manager		Total Server Memory
SQL Server Memory Manager		Target Server Memory

Question F: Is there any memory pressure on the server (Less than 100 for available mbytes)?

Question G: Is SQL Server using a good percentage of the memory on the server?

10. Remove the counters from the previous step and add the following.

Object	Counter
SQL Server:Buffer Manager	Database Pages
	Checkpoint pages/sec
	Lazy Writes/sec
	Page Life Expectancy
	Free Pages
	Stolen Pages
	Target Pages
	Total Pages

Question H: Is there memory pressure inside SQL Server? Do you have low counts for Free Pages (less than 640) and Page Life Expectancy (less than 300) or Lazy writes/sec >0?

Question I: What is the value for Database Pages vs. Target Pages?

Question J: The database pages comprise the buffer pool. Do they account for most of the target pages?

Question K: Is stolen pages a high percentage of the target pages (stolen/target)?

Stolen pages cannot be flushed from the buffer pool. So a high percentage of stolen pages could cause memory pressure inside SQL Server.

11. Remove the counters from the previous step and add the following:

Object	Counter
SQL Server:Memory Manager	Connection Memory (kb)
	Granted Workspace Memory (Kb)
	Lock Memory (kb)
	Maximum Workspace Memory (kb)
	SQL Cache Memory (kb)
	Target Server Memory (kb)
	Total Server Memory (kb)

Compare the individuals vs the Total Server Memory. Usually each category should be a small percentage of total server memory.

12. Remove the counters from the previous step and add the following:

Object	Counter	Instance
SQL Server:Plan Cache	Cache Pages	_total
		Bound Trees
		Extended Stored Procedures
		Object Plans
		SQL Plans
		Temporary Tables and Table Variables

The plan cache is a section in the buffer pool use for query plans. Compare each counter to the _total so see which comprises the majority of the procedure cache. If there are a lot of ad-hoc queries, SQL Plans may consume most of the procedure cache. If the application issues a lot of stored procedures, then Object Plans may consume most of the procedure cache.

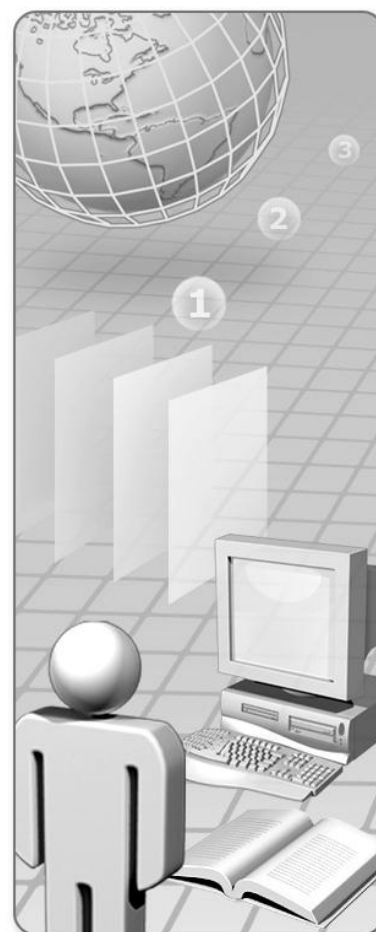
13. Open a Query Window and execute dbcc memormystatus;

Question L: Under the memory node id=0 what are the values for MultiPage Allocator and SinglePage Allocator?

Question M: Under the Buffer Pool section what is the value for Target (this results set is towards the end)?

Question N: Under the Procedure Cache what are the values for TotalProcs and TotalPages?

Module 2: Table and Index Structure



Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. These materials are intended for distribution to and use only by Microsoft Premier Customers. Use or distribution of these materials by any other persons is prohibited without the express written permission of Microsoft Corporation. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

©2010 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, Windows, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Lab 1

Exercise 1: Schema

Upon completing this exercise, you will be able to:

- Learn about the relationship between sys.objects, sys.syspartitions, and sys.indexes.
- Observe INDID = 0 is a heap, INDID=1 table has a clustered index
- Observe simple ShowPlans.

Observe indexes

1. Open and execute CreateNewContact.sql
2. Compare the differences between the indexes for the original Person.Contact table and the NewContact table by running the code below

(The queries to run for this exercise are located in the Exercise1.sql file)

```
Use AdventureWorksPTO

go

select * from sys.objects where name= 'contact'

select * from sys.indexes where object_id =
object_id('person.contact')

select * from sys.partitions where object_id =
object_id('person.contact')

select * From sys.stats where object_id =
object_id('person.contact')

go

select * from sys.objects where name= 'newcontact'

select * from sys.indexes where object_id =
object_id('person.newcontact')

select * from sys.partitions where object_id =
object_id('person.newcontact')

select * From sys.stats where object_id =
object_id('person.newcontact')

go
```

Notice that the object_id in sys.objects is the object_id in sys.indexes.

Question A: How many indexes are in Person.Contact?

Question B: Which one has INDEX_ID = 1?

Question C: Is this a clustered or nonclustered index?

Question D: How many indexes are in NewContact?

Question E: The script did not create any indexes for this new table. Considering this, what does INDEX_ID=0 mean for NewContact?

3. Execute a query against a table with no indexes (a HEAP table) and click the hyperlink for showplan at the end of your result set to see the execution plan. Also click the messages tab and note the logical and physical reads that are necessary for this.

Note: That in SQL 2005 when clicking this link you will be taken to the XML code view of the execution plan. In SQL 2008 this changed so that it takes you to the graphical view by default.

```
SET statistics xml on  
  
SET statistics time on  
  
SET statistics io on  
  
Go
```



```
SELECT lastname FROM Person.NewContact WHERE lastname between 'Devon'
and 'Mendel'
```

```
Go
```

```
SET statistics xml off
```

```
SET statistics time off
```

```
SET statistics io off
```

The important point to note here is that a Table Scan is required to find the matching rows, because there are no indexes on the NewContact table.

4. Execute the following and compare the entries for NewContact in the system tables:

```
select * from sys.objects where name= 'newcontact'
```

```
select * from sys.indexes where object_id =
object_id('person.newcontact')
```

```
select * from sys.partitions where object_id =
object_id('person.newcontact')
```

```
select * From sys.stats where object_id =
object_id('person.newcontact')
```

Note sys.stats now contains a row. In SQL Server 2000 statistics were created in the sysindexes table.

5. Contrast the execution plan when an index is added to the NewContact table.

```
CREATE index idx_last_name on person.newcontact (lastname)
```

```
go
```

```
SET statistics xml on
```

```
SET statistics time on
```

```
SET statistics io on
```

```
GO
```

```
SELECT lastname FROM Person.NewContact WHERE lastname between 'Devon'
and 'Mendel'
```

```
Go
```

```
SET statistics xml off
```

```
SET statistics time off
```

```
SET statistics io off
```

Note the change in the execution plan. Also notice the number of logical and physical reads versus the logical and physical reads for a table scan. You find these on the messages tab of your result set.

6. Execute the query below and look at the execution plan

```
SET showplan_XML on
```

```
go
```

```
SELECT firstname, lastname FROM Person.NewContact WHERE firstname
='Devon'
```

Question F: Is there any difference between this query and the previous one?

Question G: Explain why or why not:

Execute the query below:

```
-- composite index
if exists (select * from sys.indexes where name = 'idx_first_name'
and object_id = object_id('Person.NewContact'))
```

```
drop index person.newcontact.idx_first_name
go

create index idx_first_name on person.newcontact(firstname, lastname)
go

SET showplan_XML on
go

SELECT firstname, lastname
FROM Person.NewContact WHERE firstname = 'Devon'
go

SET showplan_XML off
go
```

Question H: What is different about this query from the previous one?

```
-- covered query using the include clause
DROP index person.newcontact.idx_first_name
Go

CREATE index idx_first_name on person.newcontact(firstname)
include (lastname)
Go

SET showplan_XML on
Go
```

```
SELECT firstname, lastname FROM Person.NewContact WHERE firstname  
='Devon'
```

Go

```
SET showplan_XML off
```

Go

Question I: What is different about this query from the previous one?

7. Now create a clustered index with the nonclustered index and see which index the optimizer chooses.

```
drop index person.newcontact.idx_first_name
```

go

```
drop index person.newcontact.idx_last_name
```

go

```
create index idx_first_name on person.newcontact(firstname)
```

go

```
create clustered index idx_last_name on person.newcontact(lastname)
```

go

```
SET showplan_XML on
```

go

```
SELECT firstname, lastname FROM Person.NewContact WHERE firstname  
='Devon'
```

go

```
SET showplan_XML off
```

go

Question J: Was the last query covered or not?

Exercise 2: Clustered Indexes and Page Splits

Upon completing this exercise, you will be able to:

- Observe page splitting and its causes
- Use `dm_db_index_physical_stats` to determine if an index is fragmented
- If Fragmentation is a concern

Create a page Split scenario

1. Execute `PageSplit1.sql`

```
-- Create Table with Clustered Index

USE AdventureworksPT0

GO

IF object_id('PgSplit') is not null

BEGIN

    DROP TABLE PgSplit

END

GO

CREATE TABLE PgSplit (col1 int ,col2 char(3950))

GO

CREATE UNIQUE CLUSTERED INDEX PgSplit_ind on PgSplit(col1)

GO
```

Note the clustered index on `col1`. This means the rows will be physically ordered on the data pages using this column.

Also, the `col2` field is a fixed character column with a width of 3950. Considering that each page in SQL Server is roughly 8K, only two rows will fit on each data page. This was done to more easily observe page splits.

Start Performance Monitor and monitor page splits/sec

2. On the Start menu, click Run.
3. Type `PERFMON.EXE` in the edit box and press ENTER.
4. Highlight any counters in the bottom pane of `perfmon` and press the DELETE key to remove them.

5. Click the PLUS button on the toolbar to add counters.
 6. Under “Performance Object” select SQL Server:Access Methods.
 7. Then select the “Page Splits/sec” performance counter.
 8. Click the ADD button to add this counter.
 9. Click the CLOSE button to return to perfmon.
 10. Click Ok.
-
11. Watch the Page Splits/Sec counter in perfmon, then execute the code in InsertPagesplit.sql. Notice the loop is appending 100 rows to the table out of order for the clustered index on col1 (0, 100, 1, 99, 2, 98, 3, 97, 4, 96...49,50).

Inserting the rows out of order for the clustered index forces SQL Server to periodically split a data page in order to keep the rows physically ordered by the clustered index. Large rows like the ones in this example make for more frequent page splits.

Question A: What was the max page splits/sec in perfmon? Note: Your result may be different than other students doing this and other exercises

12. Select all the rows from the table and write down the number of logical reads. Learn to use the STATISTICS IO option. Execute the following in a New Query Window. (The queries to run for this step are located in the Exercise2.sql file)

```
Use AdventureworksPT0
```

```
Go
```

```
SET STATISTICS IO ON
```

```
SELECT * FROM PgSplit
```

```
SET STATISTICS IO OFF
```

```
--NOTE: if you do this on multiproc box, the data may not be in order.
```


Question B: At the end of the query results (text results) or on the Messages tab (table results), what is the number of logical reads needed to service this query?

Observe the amount of fragmentation

13. Execute ExamineFragmentation.sql

Question C: What are the values for avg_fragmentation_in_percent and page_count?

Contrast the behavior to rows inserted in clustered –index-order

14. Execute PageSplit2.sql

Question D: What was the max page splits/sec in perfmon?

Question E: What was the difference between this value and the value seen in Step 3?

Question F: Even though we insert the rows in clustered-index-order, we still see page splits in perfmon. What does this tell you about this counter?

Question G: Will you see a page split only when you insert data?

Compare the fragmentation between PgSplit and NoPgSplit

15. Modify the code in ExamineFragementation.sql and change the table name to NoPgSplit.
16. Again note the values for avg_fragmentation_in_percent and page_count columns.

Compare the IO differences between simple queries on both tables

17. In a New Query window execute the following

```
Use AdventureworksPT0

go

SET STATISTICS IO ON

PRINT 'PgSplit Performance:'

PRINT '===== '

SELECT * FROM PgSplit WHERE col2='a'

PRINT 'NoPgSplit Performance:'

PRINT '===== '

SELECT * FROM NoPgSplit WHERE col2='a'

SET STATISTICS IO OFF
```

Question H: What are the number of logical reads for the same query on the NoPgSplit table?

Question I: Why would there be less logical reads? (Comparing the output from dm_db_index_physical_stats on both tables should provide the answer.) Rerun ExamineFragmentation.sql if you need help.

Use Alter Index to remove fragmentation

18. Run AlterIdx_PgSplit.sql.

19. Modify ExamineFragmentation.sql to use PgSplit table and execute the code.

Question J: What are the values for avg_fragmentation_in_percent and page_count?

Run the query below in a new query window and compare the logical reads vs. the number of logical reads for PgSplit in step 4.

```
Use AdventureworksPT0

go

SET STATISTICS IO ON

PRINT 'PgSplit Performance:'

PRINT '===== '

SELECT * FROM PgSplit WHERE col2='a'
```


Exercise 3: Using DMV's

Upon completing this exercise, you will be able to:

- Use DMV's to identify indexes which are not used or rarely used indexes
- Specifically use sys.dm_db_index_usage_stats to view index usage.

Run a Script to Create Unused Index Scenario

1. Restart the SQL Server Service.
2. Execute UnusedIndexes.sql.

Question A: How many rows are returned?

Question B: For table Address is the clustered index PK_Address_AddressID in the list?

3. Execute the code below in a New Query Window.

(The queries to run for this exercise are located in the Exercise3.sql file)

```
USE AdventureworksPT0
go
set statistics profile on
go
SET statistics XML on
go
```

```
SELECT * from Person.Address
```

Question C: What index is used?

4. Re-execute UnusedIndexes.sql (step 2).

Question D: How many rows are returned?

Question E: Is the index PK_address_AddressID in the list?

5. Execute RarelyUsedIndexes.sql

Question F: What index on the Address table is listed?

6. Re-execute the batch in step 3.

7. Run RarelyUsedIndexes.sql again.

Question G: Did the user_scans column increase? If so, by how much?

8. Execute the code below in a New Query Window.

```
USE AdventureworksPT0
go
set statistics profile on
go
SET statistics XML on
go

SELECT * from person.address where addressid =10
```

9. Run RarelyUsedIndexes.sql again.

Question H: Did the user_seeks column increase?

Exercise 4: Using missing Indexes XML showplan

Upon completing this exercise, you will be able to:

- Use the MissingIndexes XML showplan element to create an index to improve query performance.
- Use STATISTICS XML ON to view xml showplan

Create a Missing Index Scenario

1. Use the below query to drop the IX_SalesOrderHeader_TerritoryID Index.

Drop index IX_SalesOrderHeader_TerritoryID

on Sales.SalesOrderHeader

2. Execute MissingIndexesSQL.sql

Note number of reads and how long it took to run the query (see the Message Tab)

Question A: What were the values from logical, physical, and read-ahead reads?

Question B: What were the values for CPU and elapsed time?

3. Click the XML Showplan results set from step 1. In SQL 2008 it opens the graphic query plan, right click the band at the top that says “Query Plan Cost:” and choose the “Show Execution Plan XML” option that opens up.

Note: that in SQL 2005 the tools automatically open you up in the XML view. Also be aware that in SQL 2008 instead of going to the XML view you could just right click on the Query Plan Cost and select “Missing Index Details” and it will script the missing index to a new query window for you.

4. Locate the MissingIndexes tag

Question C: What are the columns identified in the EQUALITY column group?

Question D: What are the columns identified in the INEQUALITY column group?

Question E: What are the columns identified in the INCLUDE column group?

5. Generate the index statement based on the MissingIndexes tag in step 3 and create the index.

(The queries to run for this exercise are located in the Exercise4.sql file)

```
USE AdventureWorksPT0;

GO

IF EXISTS (SELECT name FROM sys.indexes
           WHERE name = N'IX_SalesOrderHeader_TerritoryID')
    DROP INDEX IX_SalesOrderHeader_TerritoryID ON
    Sales.SalesOrderHeader;

GO

CREATE NONCLUSTERED INDEX IX_SalesOrderHeader_TerritoryID
ON Sales.SalesOrderHeader (TerritoryID, ShipMethodID, SubTotal,
Freight) INCLUDE (SalesOrderNumber, CustomerID);

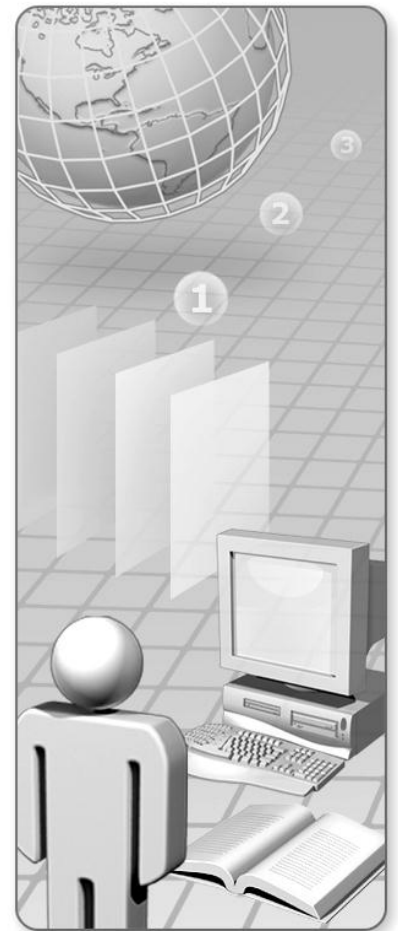
GO
```

6. Execute MissingIndexesSQL.sql again and note the number of reads and execution time.

Question F: Did the new index improve query performance? Explain.



Module 3: Performance Tools & Monitoring



Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. These materials are intended for distribution to and use only by Microsoft Premier Customers. Use or distribution of these materials by any other persons is prohibited without the express written permission of Microsoft Corporation. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

©2010 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, Windows, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Lab 1

Upon completing this Lab you will be able to explore SQL Server tools to troubleshooting SQL performance issues.

The path to files used in the virtual machine for this lab is:

C:\Labs\Module_3_Performance_Tools_SQLPTO_08

Exercise 1: Server Side Tracing

Upon completing this exercise, you will be able to:

- Explore the new trace features and definitions in profiler.
- Script the trace definitions into SQL scripts.
- Use new catalog views and functions to obtain information from trace.

Create a New Trace in profiler

1. Open Profiler by clicking START->All Programs-> SQL Server 2008 -> Performance Tools->SQL Server Profiler.
2. In Profiler, use menu option File – New Trace to create a new trace. Connect to the local instance of SQL Server.
3. Choose the "Trace to File" option and specify a file name of lab1.trc.
4. Configure the trace to capture the following events at a minimum:
 - SQL:BatchStarting
 - SQL:BatchCompleted
 - RPC:Starting
 - RPC:Completed

Capture ALL columns for the SQL:BatchStarting/SQL:BatchCompleted events
Click OK to start the trace, then immediately stop the trace by clicking the red square on the toolbar.

Note: These steps to create a server side trace can be performed on any server in an enterprise. It is not necessary to perform it on your production server.

Generate a T-SQL script of the trace settings

5. Use menu option File → Export → Script Trace Definition → For SQL Server 2005 – 2008.
6. Save the script to a file named tracescript.sql in C:\temp.

Load the trace script into Management Studio

7. Open a query window in Management Studio
8. Load the .SQL file generated by Profiler in the previous step
9. Edit the script to use a file name of lab1a.trc. Do this by locating the line that begins “exec @rc = sp_trace_create” and replacing the “InsertFileNameHere” with “c:\temp\lab1a”
10. Configure the trace file size to rollover at 1 MB. Refer to SQL Server Books Online, sp_trace_create, for the correct syntax.
11. Run the script to start tracing.
12. Close this Management Studio query window.

Generate SQL Commands to Capture

13. Open two additional query windows in Management Studio but do not execute either query.
14. In one query window load the file lab1.sql.
15. In the other query window type the query "SELECT 1".
16. Start running the query in the lab1.sql window.

This query will take ~ 30 seconds to compile.

While it is executing switch to the other window and execute the SELECT 1 query several times, a few seconds apart.

17. Execute execproc.cmd

Use Trace Functions to Read Events from trace

18. Join the results of fn_trace_gettable with sys.trace_events to retrieve the event name as well (example shown below. Note that the path and filename might have to be modified).

```
select b.name, a.*
```



```

from ::fn_trace_gettable('c:\temp\lab1a.trc', default) a join
sys.trace_events b on a.EventClass = b.trace_event_id

where b.name like 'RPC:%'

```

19. From a new query window, locate the traceid of the trace you created by running the following query:

```
select * from sys.traces
```

This will probably be traceid 2. Use the “Path” column to verify the trace writing to c:\temp\lab1a.trc.

20. Stop this trace by issuing the following commands one at a time:

```
EXEC sp_trace_setstatus 2, 0
```

```
EXEC sp_trace_setstatus 2, 2
```

Use Trace Function to Read Trace Events

21. Go back to a query window
22. use ::fn_trace_gettable to query the events, ordering by the EventSequence column. You can use the query in step 18 and remove the where clause.

```

select b.name, a.*

from ::fn_trace_gettable('c:\temp\lab1a.trc', default) a join
sys.trace_events b on a.EventClass = b.trace_event_id

order by EventSequence

```

23. Close all Query Windows. If prompted cancel any running queries, rollback any open transactions, and choose not to save changes to the files.

Exercise 2: Blocked Process Report

The key focus of this exercise is start SQL profiler trace to view Blocked Process Report event

Upon completing this exercise, you will be able to:

- Start SQL Profiler trace to view the Blocked Process Report event

Configuring the blocked process threshold

1. From a query window run

```
sp_configure 'show advanced options', 1
```

Go

Reconfigure with override

Go

```
sp_configure
```

Note: If you cut and paste the query from above you may need to replace the single quotes in order for the query to run.

Question A: What is the value for blocked process threshold?

2. Configure the threshold for 5 seconds by running

```
sp_configure 'blocked process threshold', 5
```

Go

Reconfigure with override

Note: If you cut and paste the query from above you may need to replace the single quotes in order for the query to run.

Create a new SQL Profiler trace

3. Launch SQL Profiler and choose File → New Trace.
4. Name the trace ToolsEx1

5. Enable Save to file; choose c:\temp for directory and file name ToolsEx1
6. Set Maximum File Size to 200 MB
7. On the Events tab enable 'Show all events'
8. Expand Errors and Warnings and enable the Blocked process report event.
9. Start the trace

Execute Batches to Cause Blocking

10. Execute BlockingQuery1.sql . If prompted specify connection credentials
11. Execute BlockingQuery2.sql. If prompted specify connection credentials
12. Execute BlockingQuery3.sql. If prompted specify connection credentials
13. In SQL Management Studio in the Object Explorer window select connect and make a connection to the server.
14. Right click the server name in Object Explorer and then choose Reports->Standard Reports->Activity – All Blocking Transactions.

Question B: How many transactions are blocked?

Question C: What are the session id(s) involved?

15. Stop the SQL Profiler trace.
16. In profiler search for the Blocked process report event.

Question D: How often does the Blocked Process report get triggered?

Question E: What SPIDs are involved in blocking? Is there more than 1 SPID being blocked?

Question F: What are the input buffers of the SPID involved?

Question G: What is the transaction count for each SPID?

Question H: What is the transaction isolation level for each SPID?

17. In SQL Management Studio switch back to BlockingQuery1.sql window and issue
Rollback tran

18. Close Profiler Window.

19. Close BlockingQuery1

20. Close BlockingQuery2

21. Close BlockingQuery3

When closing BlockingQuery3 you will be prompted if you want to save the transaction. Select NO.

Exercise 3: Graphical Showplan XML

Upon completing this exercise, you will be able to:

- Use SQL Profiler to view graphical query plan
- Extract graphical query plan into a separate file

Clean the Procedure Cache

1. From SQL Server Management Studio Open a new query window. If prompted specify connection credentials
2. Execute DBCC freeproccache .

Create a new SQL Profiler trace.

3. Launch SQL Profiler and choose File → New Trace.
4. Name the trace ToolsEx1
5. Enable Save to file;
6. From a command prompt, choose c:\temp\ for directory and file name ShowPlanEx1
7. Set Maximum File Size to 200 MB
8. On the Events tab enable 'Show all events'
9. Expand Performance and enable ShowPlan XML for Query Compile and ShowPlan XML.
10. Start the trace

Execute Queries to Get Showplan Information

11. Execute PerfQuery1.sql . If prompted specify connection credentials
12. Execute PerfQuery1.sql a second time. If prompted specify connection credentials.

Question A: In Profiler how many ShowPlan XML for Query Compile are reported?

13. Select one of the ShowPlan XML events and view the graphical display.
14. Right mouse click and choose Extract Event Data.
15. Save the event in c:\temp\ as PerfQuery1_plan.SQLPlan. Double click the file.

What application is launched?

16. Close all Query Windows. If prompted cancel any running queries, rollback any open transactions, and choose not to save changes to the files.

Exercise 4: SQLDiag

Upon completing this exercise, you will be able to:

- Run SQLDiag to capture performance data
- Review the captured data in SQL Profiler

Run SQLDiag

1. Open a command prompt and navigate to C:\Program Files\Microsoft SQL Server\100\Tools\Binn
2. Make a copy of SQLDIAG.XML and rename it to a SQLDIAG_Lab1.XML in the same folder.
3. Open SQLDIAG_Lab1.XML using notepad
4. Locate the PerfmonCollector,BlockingCollector,ProfilerCollector tag and set enabled="true"
5. Choose File-> Save.
6. Run sqldiag /I SQLDIAG_Lab1.XML /E +00:05:00. Once you see a 'Collection Started' message go to next step.

Note: If you are prompted that the directory already exists and would you like to overwrite. Type "Y" to overwrite.

7. Switch to SQL Management Studio and run PerfQuery1 and PerfQuery2 several times.
8. Stop SQLDIAG by navigating to the command window opened in step 6 and enter CTRL + C. Once SQLDiag has completed you should have a C:\Program Files\Microsoft SQL Server\10\Tools\Binn\SQLDiag directory.
9. Use SQL Profiler and choose File -> Open Trace to open the trace (.trc) in step 8. Once the trace is loaded choose File -> Import Performance Data and open the performance monitor log (.blg) from step 9. In the Performance Counter Limit Dialog box expand Processor and choose %User Time ->_Total and expand SQL Server:SQL Statistics and select SQL Compilations/sec. The more counters you import the longer it will take.
10. Notice as you navigate through the Profiler events the Perf Mon changes. If there are peaks in Perfmon select the peak and notice Profiler events have changed.
11. Close all Query Windows. If prompted cancel any running queries, rollback any open transactions, and choose not to save changes to the files.

Lab 2

Exercise 1: High CPU report

Upon completing this exercise, you will be able to:

- Use Standard built in reports

Load Custom Reports

1. Right click on the server name from SQL Server Management Studio and select Reports-> Standard Reports->Performance – Top Queries By Avg. CPU Time.

Question A: What is the total CPU used by the top query

2. Run a high CPU Scenario by executing
c:\ labs\Module_3_Performance_Tools_SQLPTO_08
\lab2\Scenarios\HighCPU\repro.cmd
3. Refresh the page and observe the report

Question B: What is the total CPU used by the top query?

Question C: Has the query with the highest average CPU changed?

Exercise 2: Missing Index Report

Upon completing this exercise, you will be able to:

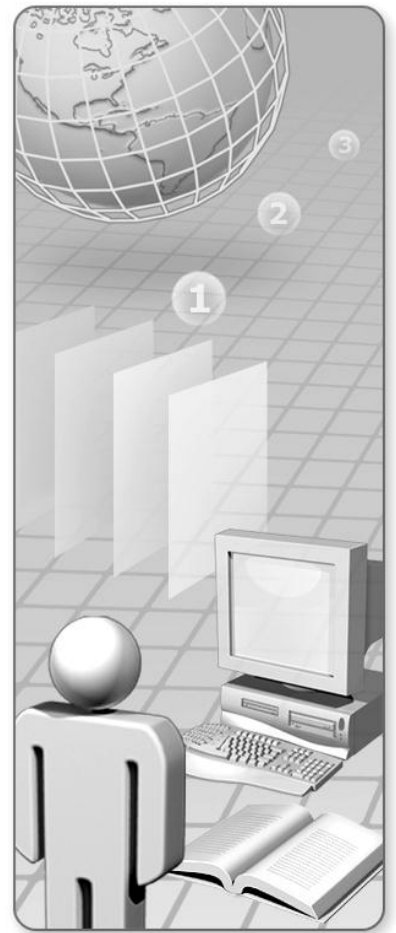
- Load custom reports to view missing index issues
- The impact of missing indexes on IO.

Create a Missing Index scenario

1. Open the query located at
C:\labs\Module_3_Performance_Tools_SQLPTO_08\lab2\reports\setup.sql in Management Studio and execute it.
2. Open the query located at
C:\Labs\Module_3_Performance_Tools_SQLPTO_08\lab2\Scenarios\MissingIndex\query.sql in Management Studio and execute it.
3. Right click on the server name from SQL Server Management Studio and select Reports->Custom Reports.
4. When prompted with a file open dialog, browse and select
C:\Labs\Module_3_Performance_Tools_SQLPTO_08\lab2\Reports\ missing_indexes.rdl
5. Examine the details of the missing index and its impact.



Module 4: Locking and Concurrency



Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. These materials are intended for distribution to and use only by Microsoft Premier Customers. Use or distribution of these materials by any other persons is prohibited without the express written permission of Microsoft Corporation. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

©2010 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, Windows, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Lab 1

Introduction

Use SQL Server DMVs to troubleshoot Blocking and Locking in SQL Server

Exercise 1: DMVs

Upon completing this exercise, you will be able to:

- Use DMV's to troubleshoot blocking and locking in SQL Server

Create a Blocking Scenario then run DMV to determine Blocking

1. Open a command prompt. Navigate to
C:\Labs\Module_4_Locking_and_Concurrency_SQLPTO_08 and run blocker1.cmd
2. Open a command prompt. Navigate to
C:\Labs\Module_4_Locking_and_Concurrency_SQLPTO_08 and run blocker2.cmd
3. Open a New Query Window and run the following select statements to determine if any blocking is occurring.

```
select * from sys.dm_exec_requests
```

Question A: What session_id has blocking_session_id > 0?

Question B: What are the values for wait_type and wait_resource?

Notice dm_exec_requests does not show any rows for the session at the head of the blocking chain. This because the session at the head of the blocking chain has entered a sleeping state due to waiting command.

4. To verify run the following:

```
Select * from sys.dm_exec_sessions
```

```
select * from sys.dm_os_waiting_tasks
```

Question C: Find the row with blocking_session_id > 0. What is the value for the columns session_id, wait_type, and resource_description?

5. Check what locks are in place and use the locks to determine if blocking is occurring.

```
select * from sys.dm_tran_locks
```

Question D: What row has request_status = Wait?

Question E: Is the request_session_id value the same session_id that was listed as blocked in step 2?

Question F: What are the values for resource_type, request mode, resource_database_id, resource_associated_entity_id, resource_description, request_owner_type, and request_owner_id for the blocked row?

Notice request_owner_id is not a session_id but is instead a transaction id.

Find the row with the same resource_description, resource_associated_entity_id, and resource_database_id. Notice the differences between request_mode between this row and the one identified earlier.

6. To determine what object is being locked take the value from the resource_database_id column and run the following queries substituting the appropriate value for <resource_database_id>

```
Select db_name(<resource_database_id >)
```

Switch to the database identified in the query above and run the following substituting the appropriate value for <resource_associated_entity_id>

```
select OBJECT_NAME(object_id) as [Object Name], * from sys.partitions  
where hobt_id=(<resource_associated_entity_id>)
```

7. Execute Create_sp_block_info.sql
8. In a New Query window execute sp_block_info
9. Close all query windows and navigate to the command windows opened in step 1 and step 2. ctrl+c to end task and close each command window.

Exercise 2: Isolation Level

Upon completing this exercise, you will be able to:

- Illustrate how snapshot isolation level can effect blocking and locking
- Illustrate how read_committed_snapshot database option can change locks requested for select statements.

Create a Blocking Scenario with READ_COMMITTED_SNAPSHOT

1. Open BlockingQuery1.sql and execute this.
2. Open BlockingQuery2 in another window and execute this. This session should be blocked.
3. Open a third query window, execute sp_block_info (installed in Exercise 1) and examine the blocking that is created by this combination.
4. Close all the query windows. If you are prompted to commit transactions before closing, choose “no.” If you are prompted to cancel the transactions, choose “yes.”
5. Enable the READ_COMMITTED_SNAPSHOT database option by opening a new query window and issuing the following statement:

```
ALTER DATABASE AdventureWorksPTO
    SET READ_COMMITTED_SNAPSHOT ON;
```

Make sure all other query windows have been closed when that statement is executed.

6. Execute BlockingQuery1.sql again.
7. Execute BlockingQuery2.sql Did you get results set from this query this time? Under the read committed transaction isolation level BlockingQuery2 was blocked. But once the READ_COMMITTED_SNAPSHOT database option is enabled the select statement is not blocked because it no longer takes shared locks. Analyze the locks

```
Select * from sys.dm_tran_locks
```

8. Close query windows opened in both steps 2 and 3. If prompted cancel the query and rollback any open transactions.

9. Disable READ_COMMITTED_SNAPSHOT database option.

```
ALTER DATABASE AdventureWorksPTO
    SET READ_COMMITTED_SNAPSHOT OFF;
```

10. Enable ALLOW_SNAPSHOT_ISOLATION database option.

```
ALTER DATABASE AdventureWorksPTO
    SET ALLOW_SNAPSHOT_ISOLATION ON;
```

11. Execute BlockingQuery1.sql

12. Execute BlockingQuery2.sql .

Question A: Was a results set returned? Is this query being blocked? Analyze the locks.

```
Select * from sys.dm_tran_locks
```

13. Open a New Query Window and execute the batch below.

```
use adventureworkspto
go
set transaction isolation level snapshot
go
select * from person.address
```

Question B: Is a results set returned? What type of locks did this session grab? Analyze the locks.

```
Select * from sys.dm_tran_locks
```

14. Close Query windows used in step 7, 8, and 9.
15. Turn off ALLOW_SNAPSHOT_ISOLATION database option.

```
ALTER DATABASE AdventureWorksPTO  
SET ALLOW_SNAPSHOT_ISOLATION OFF;
```


Exercise 3: Deadlocks

Upon completing this exercise, you will be able to:

- Analyze complex deadlocks using Deadlock graph Profiler event

Create a new SQL Profiler trace.

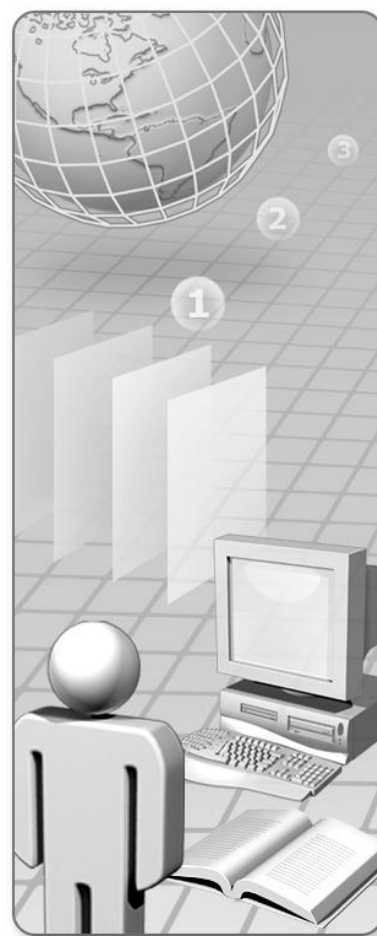
1. Launch SQL Profiler and choose File → New Trace.
2. Name the trace ToolsEx1
3. Enable Save to file; choose c:\temp for directory
4. Set Maximum File Size to 200 MB
5. On the Events tab enable 'Show all events'
6. Expand Locks and enable the Deadlock graph event.
7. Start the trace by clicking the Run button.

Create a deadlock scenario

8. Open a Query Window and execute CreateDeadlockTables.sql
9. Execute the following command file Deadlock1.cmd
10. Go to SQL profiler trace and verify a deadlock occurred, then pause the trace.
11. Analyze the deadlock detection.
12. Stop the trace and close SQL Profiler.
13. Open a Query window and execute DropDeadlockTables.sql
14. Close all query windows.



Module 5: Query Optimization



Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. These materials are intended for distribution to and use only by Microsoft Premier Customers. Use or distribution of these materials by any other persons is prohibited without the express written permission of Microsoft Corporation. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

©2010 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, Windows, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Lab 1

Exercise 1: Asynchronous Update Statistics

Upon completing this exercise, you will be able to:

- Understand the difference between synchronous or asynchronous updates
- Use DMV's to query background jobs
- Use Trace Events to view update statistics events

Create a Scenario to Generate AutoStats

1. Configure and start a Profiler trace that captures, at a minimum, the SQL:BatchStarting/Completed events and Performance: AutoStats events. Capture ALL columns for these events.
2. Open the file scenario1.sql in Management Studio and run it.
3. Stop the Profiler trace. You should have three AutoStats events in the trace.

Question A: Why are there three trace events?

Question B: The TextData column is blank. Which column do you have to look at to determine what statistics/index is being maintained?

Question C: From the data in the trace events themselves can you confirm that the stats were updated asynchronously? Hint: Refer to SQL Server BOL in the Auto Stats Event topic for a detailed view of the column values.

4. Restart the Profiler trace in preparation of running the second scenario.

5. Make sure you have closed the query window opened in step 2. Open the second SQL file scenario2.sql and run it. Stop the trace when this completes.

Question D: Looking at the trace, were the statistics updated asynchronously?
Why or Why not?

6. Query the sys.dm_exec_background_job_queue_stats DMV

Question E: How many stats jobs have run?

Question F: What are their average and max response times?

Exercise 2: Identifying and tuning expensive queries in the cache

Upon completing this exercise, you will be able to:

- Understand how to use sys.dm_exec_query_stats to identify expensive queries
- Understand the XML Showplan

Identify expensive query

1. Open a Query Window and run the Exercise2_setup.sql.
2. Open a Query Window and run execute_perf_proc1.sql
3. Open another Query Window and use the query below to identify the expensive queries based on IO.

```
SELECT TOP 20 last_execution_time, (total_physical_reads +
total_logical_writes + total_logical_reads) AS [Total IO],

text, qp.query_plan, statement_start_offset, statement_end_offset,

sql_handle, plan_handle

FROM sys.dm_exec_query_stats AS qs

CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) AS st

CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle) as qp

ORDER BY [total io] DESC;
```

Question A: How many rows are returned that list the procedure perf_proc1 in the text column? Write down the value for the [total io] column for each row which has perf_proc1 in the text column.

- A. For the rows which list create procedure perf_proc1, are the sql_handle columns different for each row? How about the plan_handle columns? Note the value for the sql_handle, and the plan_handle.

4. Use the query below to identify expensive queries based on IO.

```
SELECT TOP 20 last_execution_time, (total_physical_reads +
total_logical_writes + total_logical_reads) AS [Total IO],

qp.query_plan, sql_handle, plan_handle,

(SELECT SUBSTRING(text, statement_start_offset/2,

(CASE WHEN statement_end_offset = -1 THEN LEN(CONVERT(nvarchar(max),
text)) * 2

ELSE statement_end_offset END - statement_start_offset)/2)

FROM sys.dm_exec_sql_text(sql_handle)) AS query_text

FROM sys.dm_exec_query_stats qs

CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle) as qp

ORDER BY [total io] DESC;
```

- Question B:** Find the rows with the same sql_handle that you noted in the previous step. Are you able to determine based on the query the select statements are part of a stored procedure?

5. Use the query below to identify expensive queries based on IO.

```
SELECT TOP 20 last_execution_time, (total_physical_reads +
total_logical_writes + total_logical_reads) AS [Total IO],
```



```

sql_handle, plan_handle, qp.*,

(SELECT SUBSTRING(text, statement_start_offset/2,

(CASE WHEN statement_end_offset = -1 THEN LEN(CONVERT(nvarchar(max),
text)) * 2

ELSE statement_end_offset END - statement_start_offset)/2)

FROM sys.dm_exec_sql_text(sql_handle)) AS query_text

FROM sys.dm_exec_query_stats qs

CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle) as qp

ORDER BY [total io] DESC;

GO

```

Question C: How many rows have the same values for the dbid column and objectid?

Question D: How can you get the name for the object?

6. Click the showplan column for the row identified with the highest Total IO. Right click the bar at the top with the query text and choose “Show Execution Plan XML.” Find RelOp NodeId="1".

Question E: What type of join algorithm is being used?

7. Find <HashKeysBuild> under RelOp NodeId = 1.

Question F: Which table is the build table?

8. Find <HashKeysProbe> under RelOp NodeId = 1.

Question G: Which table is the probe table?

9. Find RelOpNodeId="2".

Question H: What table is this? Is this the build or probe table? How is this table accessed?

10. Find RelOpNodeId="3".

Question I: What table is this? Is this the build or probe table? How is this table accessed?

11. Collapse RelOpNodeId= "1". So find <RelOpNodeId="1"> and click on the minus sign on the left.

12. Find RelOpNodeId="0".

Question J: What type of join algorithm is being used?

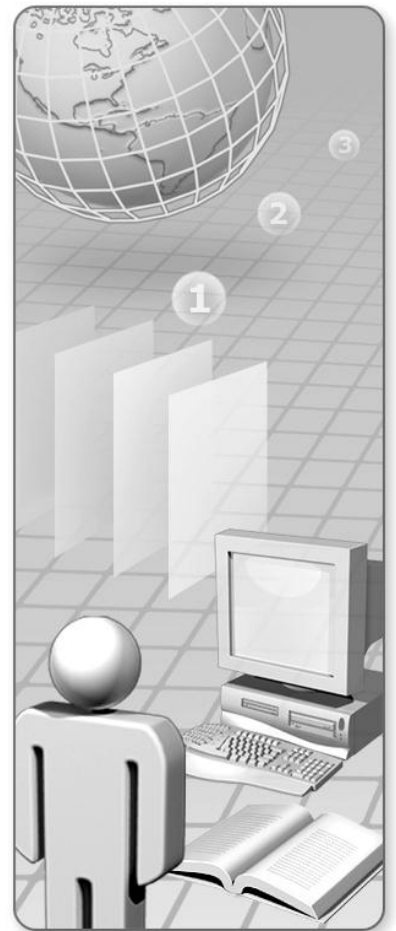
13. Find RelOpNodeId="5".

Question K: What table is this? Is this the build or probe table? How is this table accessed?

In summary, the query plan shows salesorderheader_pto and customer_pto are joined together to produce a results set using a hash join. Then the results set is joined to salesorderdetail_pto again using a hash join.

14. Find the <MissingIndexes> tag. Generate a create index statement for each <MissingIndexGroup> node.
15. On the graphical execution plan, right click the green text that says “Missing Index,” and choose “Missing Index Details” to generate a create index statement for each missing index. This can only be done using the SQL 2008 tools.

Module 7: Resource Governor



Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. These materials are intended for distribution to and use only by Microsoft Premier Customers. Use or distribution of these materials by any other persons is prohibited without the express written permission of Microsoft Corporation. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

©2010 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, Windows, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Lab 1

Exercise 1: Setup Resource Governor for Workloads

Upon completing this exercise, you will be able to:

- Setup the Resource Governor for multiple workloads
- Setup and configure workload groups and resource pools
- Define and configure a classifier function
- Monitor performance and match to the configuration of the Resource Governor

Setup Resource Governor

1. Open C:\Labs\Module_7_Resource_Governor_SQLPTO_08\SetupMemory.sql in management studio and execute the script. This configures SQL Server's max and min server to a fixed size that will be easy to monitor in the lab.

The next two steps will setup for three workload groups (Marketing, Admin, and VP) who will use two resource pools (PoolMarketingAdmin, and PoolVP). The pools will be built first.

2. Open C:\Labs\Module_7_Resource_Governor_SQLPTO_08\CreateResourcePoolsGroups.sql In a new query window, execute the following to set up the pools:

```
CREATE RESOURCE POOL PoolMarketingAdmin
WITH (MAX_CPU_PERCENT = 70, MIN_CPU_PERCENT = 60)
```

```
CREATE RESOURCE POOL PoolVP
WITH (MAX_CPU_PERCENT = 60, MIN_CPU_PERCENT = 40)
```

3. Execute the following to setup the three workload groups and assign them to the resource pools created in step 2:

```
CREATE WORKLOAD GROUP GroupMarketing
USING PoolMarketingAdmin
```

```
CREATE WORKLOAD GROUP GroupAdmin
USING PoolMarketingAdmin
```

```
CREATE WORKLOAD GROUP GroupVP
USING PoolVP
```

4. Check the configurations you have just created:

```
SELECT * from sys.resource_governor_resource_pools
```

```
SELECT * FROM sys.resource_governor_workload_groups
```

At this point, the resource pools and groups are created, but SQL Server does not know which queries to use them for. Classification is necessary for SQL Server to make use of them.

To perform the classification, we need to create a user defined function that will be used for every new connection. This function will place the new connections into the corresponding workload groups.

5. There must be some criteria used to classify the connections. For demo purposes, we will use three logins. The user names for these logins will be used to classify the connections into workload pools. Validate the following three logins exist. If not, use the CreateLogins.sql script to create them.

```
UserMarketing, pwd = MarketingGroup1  
UserAdmin, pwd = AdminGroup1  
UserVP, pwd = VPGroup1”
```

6. Open the script SetupClassifierFunction.sql. Examine the function defined there. The name of the resource group to which this connection will be assigned. Execute the script to create the function.
7. At this point, the function is created, but SQL Server has not associated the function with the resource governor. Associate the governor with the newly created function by executing the following statement:

```
ALTER RESOURCE GOVERNOR  
WITH (CLASSIFIER_FUNCTION = dbo.CLASSIFIER_V1)
```

8. Execute each of the following pairs of queries.

```
SELECT * FROM sys.resource_governor_workload_groups  
SELECT * FROM sys.dm_resource_governor_workload_groups
```

```
SELECT * FROM sys.resource_governor_resource_pools  
SELECT * FROM sys.dm_resource_governor_resource_pools
```

```
SELECT * FROM sys.resource_governor_configuration  
SELECT * FROM sys.dm_resource_governor_configuration
```

Question A: Does the information from the system catalogs match the information from the DMVs?

9. Execute the following to sync the in-memory information with the configuration in the system catalogs:

```
ALTER RESOURCE GOVERNOR RECONFIGURE
```

10. Execute the queries in step 8 again.

Question B: Have the newly created Resource Groups and Resource Pools now appeared in the DMVs?

11. Open perfmon, and add the following counters:

- a. Processor: % Processor Time (_total instance)
- b. SQL Server: Workload Group Stats\CPU Usage % (All instances)

12. Open a new command prompt window. Navigate to C:\Labs\Module_7_Resource_Governor_SQLPTO_08 and issue the following statement:

```
start sqlcmd -U UserMarketing -P MarketingGroup1 -i CreateLoad.sql
```

13. Look at the values for each of the instances of Workload Group Stats CPU Usage % and the Processor: % Processor time. After you have a feel for the counters, issue the following statement in the original command prompt window:

```
start sqlcmd -U UserVP -P VPGroup1 -i CreateLoad.sql
```

14. Look again at perfmon, and notice now that the two new processes are splitting the processor time about 60% and 40%. Note that this is how we set the minimum processor values in step 1.

15. In the original command prompt window, issue the following statement:

```
start sqlcmd -U UserAdmin -P AdminGroup1 -i CreateLoad.sql
```

16. In perfmon, look at the counters for the 3 instances of the WorkloadGroup Stats CPU Usage %.

Question C: How much CPU is being used by the GroupVP group?

Question D: How much CPU is being used by the GroupAdmin group?

Question E: How much CPU is being used by the GroupMarketing group?

Question F: In step 3 notice that GroupMarketing and GroupAdmin were both set up in the PoolMarketingAdmin resource pool. Does the sum of the CPU usage for these two groups match the minimum or maximum configured CPU percentage for PoolMarketingAdmin?

17. Close the SQLCMD windows one by one and watch the effect on the perfmon counters of closing each one.
18. Close all query windows in Management studio, restart SQL Server, then run the CleanupResourceGovernor.sql script.