CHAPTER 3

Manage and monitor SQL Server instances

LISTING 3-1 Querying current sessions

```
-- Query currently connected sessions with their stastics
SELECT c.session_id, c.net_transport, c.encrypt_option,
    c.auth_scheme, s.host_name, s.program_name,
    s.client_interface_name, s.login_name, s.nt_domain,
    s.nt_user_name, c.connect_time, s.login_time,
    s.reads, s.writes, s.logical_reads, s.status,
    s.cpu_time, s.total_scheduled_time, s.total_elapsed_time
FROM sys.dm_exec_connections AS c
JOIN sys.dm_exec_sessions AS s ON c.session_id = s.session_id;
GO
```

LISTING 3-2 Idle sessions with open transactions

```
-- Using the sys.dm_exec_requests
SELECT session_id, blocking_session_id, open_transaction_count, wait_time, wait_type,
    last_wait_type, wait_resource, transaction_isolation_level, lock_timeout
FROM sys.dm_exec_requests
WHERE blocking_session_id <> 0;
GO
-- Using the sys.dm_os_waiting_tasks
SELECT session_id, blocking_session_id, wait_duration_ms, wait_type,
resource_description
FROM sys.dm_os_waiting_tasks
WHERE blocking_session_id IS NOT NULL
```

LISTING 3-3 Idle sessions with open transactions

```
SELECT s.*
FROM sys.dm_exec_sessions AS s
WHERE EXISTS (
    SELECT *
    FROM sys.dm_tran_session_transactions AS t
    WHERE t.session_id = s.session_id
)
```



```
AND NOT EXISTS (
    SELECT *
    FROM sys.dm_exec_requests AS r
    WHERE r.session_id = s.session_id
);
```

LISTING 3-4 Queries allocated the most space in tempdb

```
SELECT r.session_id, r.request_id, t.text AS query,
    u.allocated AS task_internal_object_page_allocation_count,
    u.deallocated AS task_internal_object_page_deallocation_count
FROM (
    SELECT session_id, request_id,
        SUM(internal_objects_alloc_page_count) AS allocated,
        SUM (internal_objects_dealloc_page_count) AS deallocated
    FROM sys.dm_db_task_space_usage
    GROUP BY session_id, request_id) AS u
JOIN sys.dm_exec_requests AS r
ON u.session_id = r.session_id  AND u.request_id = r.request_id
CROSS APPLY sys.dm_exec_sql_text (r.sql_handle) as t
ORDER BY u.allocated DESC;
```

LISTING 3-5 Configuring query store

```
ALTER DATABASE [WideWorldImporters]
    SET QUERY_STORE (OPERATION_MODE = READ_WRITE,
    CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 366),
    DATA_FLUSH_INTERVAL_SECONDS = 3600,
    INTERVAL_LENGTH_MINUTES = 5,
    MAX_STORAGE_SIZE_MB = 5000);
GO
```

LISTING 3-6 Analyzing and managing the query store

```
-- Determine the top 100 queries with the longest average duration for the last week
SELECT TOP 100 r.avg_duration, r.last_execution_time, t.query_sql_text
FROM sys.query_store_query_text AS t
JOIN sys.query_store_query AS q ON t.query_text_id = q.query_text_id
JOIN sys.query_store_plan AS p ON q.query_id = p.query_id
JOIN sys.query_store_runtime_stats AS r ON p.plan_id = r.plan_id
WHERE r.last_execution_time > DATEADD(day, -7, GETUTCDATE())
ORDER BY r.avg_duration DESC;
GO
```


LISTING 3-7 Analyzing and managing the query store

```
-- Delete ad-hoc queries (single use plans) older than 24 hours
DECLARE @query_id INT;
DECLARE adhoc_queries_cursor CURSOR FOR
    SELECT q.query_id
    FROM sys.query_store_query_text AS t
    JOIN sys.query_store_query AS q ON q.query_text_id = t.query_text_id
    JOIN sys.query_store_plan AS p ON p.query_id = q.query_id
    JOIN sys.query_store_runtime_stats AS r ON r.plan_id = p.plan_id
    GROUP BY q.query_id
    HAVING SUM(r.count_executions) < 2
    AND MAX(r.last_execution_time) < DATEADD (hour, -24, GETUTCDATE())
    ORDER BY q.query_id;
OPEN adhoc_queries_cursor;
FETCH NEXT FROM adhoc_queries_cursor INTO @query_id;
WHILE (@@FETCH_STATUS = 0) BEGIN
    EXEC sp_query_store_remove_query @query_id
    FETCH NEXT FROM adhoc_queries_cursor INTO @query_id
END
CLOSE adhoc_queries_cursor;
DEALLOCATE adhoc_queries_cursor;
GO
-- Force a query to use a specific plan
EXEC sp_query_store_force_plan @query_id = 66, @plan_id = 69;
GO
-- Unforce a query to use a specific plan
EXEC sp_query_store_unforce_plan @query_id = 66, @plan_id = 69;
GO
```

LISTING 3-8 Extended event actions

```
    SELECT p.name AS package_name, o.name AS action_name, o.description
    FROM sys.dm_xe_packages AS p
    JOIN sys.dm_xe_objects AS o ON p.guid = o.package_guid
    WHERE (p.capabilities IS NULL OR p.capabilities & 1 = 0) -- Exclude private
    packages
    AND (o.capabilities IS NULL OR o.capabilities & 1 = 0) -- Exclude private objects
    AND o.object_type = 'action'
    ORDER BY package_name, action_name;
```

LISTING 3-9 Extended event creation


```
CREATE EVENT SESSION [Page Splits] ON SERVER
ADD EVENT sqlserver.page_split(
    ACTION(sqlserver.sql_text)
    WHERE ([database_id]=(5)))
ADD TARGET package0.event_counter,
ADD TARGET package0.ring_buffer
WITH (MAX_MEMORY=4096 KB, EVENT_RETENTION_MODE=ALLOW_SINGLE_EVENT_LOSS,
MAX_DISPATCH_LATENCY=30 SECONDS, MAX_EVENT_SIZE=0 KB,MEMORY_PARTITION_MODE=NONE,
TRACK_CAUSALITY=OFF,STARTUP_STATE=OFF)
GO
```

LISTING 3-10 Extended event creation

```
CREATE EVENT SESSION [Page Splits] ON SERVER
ADD EVENT sqlserver.page_split(
    ACTION(sqlserver.sql_text)
    WHERE ([database_id]=(5)))
ADD TARGET package0.event_counter,
ADD TARGET package0.ring_buffer
WITH (MAX_MEMORY=4096 KB, EVENT_RETENTION_MODE=ALLOW_SINGLE_EVENT_LOSS,
MAX_DISPATCH_LATENCY=30 SECONDS, MAX_EVENT_SIZE=0 KB,MEMORY_PARTITION_MODE=NONE,
TRACK_CAUSALITY=OFF,STARTUP_STATE=OFF)
GO
```

LISTING 3-11 Extended event creation

```
USE master;
exec sp_configure 'show advanced options',1;
GO
RECONFIGURE;
GO
EXEC sp_configure 'blocked process threshold',5;
GO
RECONFIGURE;
GO
CREATE EVENT SESSION [BlockedProcesses] ON SERVER
ADD EVENT sqlserver.blocked_process_report(
ACTION(sqlserver.client_app_name,sqlserver.client_hostname,sqlserver.database_name))
ADD TARGET package0.event_file(
    SET filename=N'C:\ExtendedEvents\BlockedProcesses.xel',
    max_rollover_files=(10)
)
WITH (STARTUP_STATE=ON)
GO
```


LISTING 3-12 Most expensive cached plans by average execution time

```
SELECT TOP(100) OBJECT_NAME(t.objectid, t.dbid) AS object_name,
    s.total_elapsed_time / s.execution_count AS average_duration,
    s.execution_count,
    s.last_execution_time,
    total_worker_time,
    SUBSTRING (t.[text],
        (s.statement_start_offset/2)+1,
        (( CASE statement_end_offset
        WHEN -1 THEN DATALENGTH(t.[text])
        ELSE s.statement_end_offset
        END - s.statement_start_offset)/2)+1
    ) AS statement,
    [text] as query,
    query_plan
FROM sys.dm_exec_query_stats AS s
CROSS APPLY sys.dm_exec_sql_text(s.sql_handle) AS t
CROSS APPLY sys.dm_exec_query_plan(s.plan_handle) AS p
ORDER BY average_duration DESC;
GO
```

LISTING 3-13 Cached plans with missing indexes

```
;WITH XMLNAMESPACES(DEFAULT N'http://schemas.microsoft.com/sqlserver/2004/07/showplan')
SELECT p.usecounts, p.refcounts, p.objtype, p.cacheobjtype,
    db_name(t.dbid) as database_name, t.text as query, q.query_plan
FROM sys.dm_exec_cached_plans AS p
    CROSS APPLY sys.dm_exec_sql_text(p.plan_handle) AS t
    CROSS APPLY sys.dm_exec_query_plan(p.plan_handle) AS q
WHERE
q.query_plan.exist(N'/ShowPlanXML/BatchSequence/Batch/Statements/StmtSimple/QueryPlan/Mi
ssingIndexes/MissingIndexGroup') <> 0
ORDER BY p.usecounts DESC
```

LISTING 3-14 Cached plans with implicit warnings

```
;WITH XMLNAMESPACES(DEFAULT N'http://schemas.microsoft.com/sqlserver/2004/07/showplan')
SELECT
    operators.value('@ConvertIssue', 'nvarchar(250)') as convert_issue,
    operators.value('@Expression', 'nvarchar(250)') AS convert_expression,
    t.text AS query, p.query_plan
FROM sys.dm_exec_query_stats s
```



```
CROSS APPLY sys.dm_exec_query_plan(s.plan_handle) p
CROSS APPLY query_plan.nodes('//Warnings/PlanAffectingConvert') rel(operators)
CROSS APPLY sys.dm_exec_sql_text(s.plan_handle) AS t
ORDER BY p.usecounts DESC
```

LISTING 3-15 Detecting deadlock in the server_health session

```
SELECT x.value('@timestamp', 'datetime') as deadlock_datetime,
    x.query('.') AS deadlock_payload
FROM (
    SELECT CAST(target_data AS XML) AS Target_Data
    FROM sys.dm_xe_session_targets AS t
    JOIN sys.dm_xe_sessions AS s ON s.address = t.event_session_address
    WHERE s.name = N'system_health' AND t.target_name = N'ring_buffer'
) AS XML_Data
CROSS APPLY
Target_Data.nodes('RingBufferTarget/event[@name="xml_deadlock_report"]') AS
XEventData(x);
```

LISTING 3-16 Identifying missing indexes

```
SELECT g.*, statement AS table_name, column_id, column_name, column_usage
FROM sys.dm_db_missing_index_details AS d
CROSS APPLY sys.dm_db_missing_index_columns (d.index_handle)
INNER JOIN sys.dm_db_missing_index_groups AS g ON g.index_handle = d.index_handle
ORDER BY g.index_group_handle, g.index_handle, column_id;
GO
```

LISTING 3-17 Identifying underutilized indexes

```
SELECT DB_NAME(s.database_id) AS 'datase_name', OBJECT_NAME(s.object_id) AS
'table_name',
    i.name AS 'index_name',
    s.user_seeks, s.user_scans, s.user_lookups, s.user_updates,
    s.last_user_seek, s.last_user_scan, s.last_user_lookup, s.last_user_update
FROM sys.dm_db_index_usage_stats AS s
JOIN sys.indexes AS i ON s.index_id = i.index_id
AND s.object_id = i.object_id
WHERE s.database_id = DB_ID()
AND s.user_seeks = 0 AND s.user_scans = 0 AND s.user_lookups = 0;
GO
```


LISTING 3-18 Identifying underutilized indexes

```
SELECT DB_NAME(s.database_id) AS 'datase_name', OBJECT_NAME(s.object_id) AS
'table_name',
    i.name AS 'index_name',
    s.user_seeks, s.user_scans, s.user_lookups, s.user_updates,
    s.last_user_seek, s.last_user_scan, s.last_user_lookup, s.last_user_update
FROM sys.dm_db_index_usage_stats AS s
JOIN sys.indexes AS i ON s.index_id = i.index_id
AND s.object_id = i.object_id
WHERE s.database_id = DB_ID()
AND s.user_updates > (s.user_seeks + s.user_scans + s.user_lookups)
AND s.index_id > 1;
```

LISTING 3-19 Identifying fragmentation in columnstore indexes

```
SELECT i.object_id,
    OBJECT_NAME(i.object_id) AS table_name,
    i.name AS index_name,
    i.index_id,
    i.type_desc,
    100*(ISNULL(deleted_rows,0))/total_rows AS 'Fragmentation',
    s.*
FROM sys.indexes AS i
JOIN sys.dm_db_column_store_row_group_physical_stats AS s
ON i.object_id = s.object_id AND i.index_id = s.index_id
ORDER BY fragmentation DESC;
```

LISTING 3-20 Reorganizing columnstore index

```
USE AdventureworksDW
GO
ALTER INDEX IndFactResellerSalesXL_CCI ON dbo.FactResellerSalesXL_CCI
REORGANIZE WITH (LOB_COMPACTION = ON, COMPRESS_ALL_ROW_GROUPS = ON);
```

LISTING 3-21 Rebuilding columnstore index

```
USE AdventureworksDW
GO
ALTER INDEX IndFactResellerSalesXL_CCI ON dbo.FactResellerSalesXL_CCI
REBUILD WITH (DATA_COMPRESSION = COLUMNSTORE_ARCHIVE);
/*
-- Let's assume the [IndFactResellerSalesXL_CCI] table was partitioned
-- Rebuild only 1 partition
```



```
ALTER TABLE IndFactResellerSalesXL_CCI
REBUILD PARTITION = 1 WITH (DATA_COMPRESSION =  COLUMNSTORE_ARCHIVE);
GO
-- Rebuild all partitions but with different data compression
ALTER TABLE [ColumnstoreTable]
REBUILD PARTITION = ALL WITH (
    DATA_COMPRESSION = COLUMNSTORE ON PARTITIONS (5,6,7,8,9,10,11,12),
    DATA COMPRESSION = COLUMNSTORE_ARCHIVE ON PARTITIONS (1,2,3,4)
);
*/
```

LISTING 3-22 Querying statistics metadata for all user tables

```
SELECT s.name AS statistic_name, s.auto_created, s.user_created,
    s.no_recompute, s.is_incremental, s.is_temporary, s.has_filter,
    p.last_updated, DATEDIFF(day,p.last_updated, SYSDATETIME()) AS days_past,
    h.name AS schema_name, o.name AS table_name, c.name AS column_name,
    p.rows, p.rows_sampled, p.steps, p.modification_counter
FROM sys.stats AS s
JOIN sys.stats_columns i
ON s.stats_id = i.stats_id AND s.object_id = i.object_id
JOIN sys.columns c
ON c.object_id = i.object_id AND c.column_id = i.column_id
JOIN sys.objects o
ON s.object_id = o.object_id
JOIN sys.schemas h
ON o.schema_id = h.schema_id
OUTER APPLY sys.dm_db_stats_properties (s.object_id,s.stats_id) AS p
WHERE OBJECTPROPERTY(o.object_id, N'IsMSShipped') = 0
ORDER BY days_past DESC;
```

LISTING 3-23 sp_send_dbmail

```
DECLARE @tableHTML  NVARCHAR(MAX);
SET @tableHTML =
    N'<H1>Work Order Report</H1>' +
    N'<table border="1">' +
    N'<tr><th>Work Order ID</th><th>Product ID</th>' +
    N'<th>Name</th><th>Order Qty</th><th>Due Date</th>' +
    N'<th>Expected Revenue</th></tr>' +
    CAST ( ( SELECT td = wo.WorkOrderID,       '',
                    td = p.ProductID, '',
                    td = p.Name, '',
                    td = wo.OrderQty, '',
```



```
                    td = wo.DueDate, '',
                    td = (p.ListPrice - p.StandardCost) * wo.OrderQty
                FROM AdventureWorks.Production.WorkOrder as wo
                JOIN AdventureWorks.Production.Product AS p
                ON wo.ProductID = p.ProductID
                WHERE DueDate > '2004-04-30'
                  AND DATEDIFF(dd, '2004-04-30', DueDate) < 2
                ORDER BY DueDate ASC,
                         (p.ListPrice - p.StandardCost) * wo.OrderQty DESC
                FOR XML PATH('tr'), TYPE
    ) AS NVARCHAR(MAX) ) +
    N'</table>' ;

EXEC msdb.dbo.sp_send_dbmail @recipients='victor.isakov@sql.local',
    @subject = 'Work Order List',
    @body = @tableHTML,
    @body_format = 'HTML' ;
ORDER BY days_past DESC;
```

LISTING 3-24 Create new operator

```
EXEC msdb.dbo.sp_add_operator @name=N'Victor Isakov',
    @enabled=1,
    @weekday_pager_start_time=90000,
    @weekday_pager_end_time=180000,
    @saturday_pager_start_time=90000,
    @saturday_pager_end_time=180000,
    @sunday_pager_start_time=90000,
    @sunday_pager_end_time=180000,
    @pager_days=0,
    @email_address=N'victor.isakov@SQL.LOCAL',
    @pager_address=N'marcus.isakov@SQL.LOCAL',
    @category_name=N'[Microsoft Certified Architect]';
```

LISTING 3-25 Create SQL Agent alerts

```
EXEC msdb.dbo.sp_add_alert @name=N'WorldWideImporters Transaction Log Full',
    @message_id=9002,
    @severity=0,
    @enabled=1,
    @delay_between_responses=900,
    @include_event_description_in=1,
    @database_name=N'WideWorldImporters',
```



```
    @category_name=N'[DBA]',
    @job_id=N'00000000-0000-0000-0000-000000000000';
GO

EXEC msdb.dbo.sp_add_alert @name=N'WorldWideImporters Transaction Log 75% Full',
    @message_id=0,
    @severity=0,
    @enabled=1,
    @delay_between_responses=0,
    @include_event_description_in=0,
    @category_name=N'[DBA]',
    @performance_condition=N'Databases|Percent Log Used|WideWorldImporters|>|75',
    @job_id=N'f857a2e0-f118-4cf4-82d5-277a52941d80';
```

LISTING 3-26 Identifying available space on CSVs

```
SELECT DISTINCT DB_NAME(s.database_id) AS database_name,
    s.database_id, s.volume_mount_point, s.volume_id,
    s.logical_volume_name, s.file_system_type, s.total_bytes, s.available_bytes,
    ((s.available_bytes*1.0)/s.total_bytes) as percent_free
FROM sys.master_files AS f
CROSS APPLY sys.dm_os_volume_stats(f.database_id, f.file_id) AS s;
```

