John Deardurff

Microsoft Customer Engineer (Global Technical Team)
Microsoft Certified Trainer (Regional Lead)
MVP: Data Platform (2016 – 2018)
Email: John.Deardurff@Microsoft.com
Twitter: @SQLMCT
Website: www.SQLMCT.com
GitHub: github.com\SQLMCT

MVP Microsoft® Most Valuable Professional Reconnect

Microsoft Certified Trainer Microsoft MCT Trainer 2021 - 2022

Microsoft Certified Professional Microsoft MCSE Data Management and Analytics Earned 2016

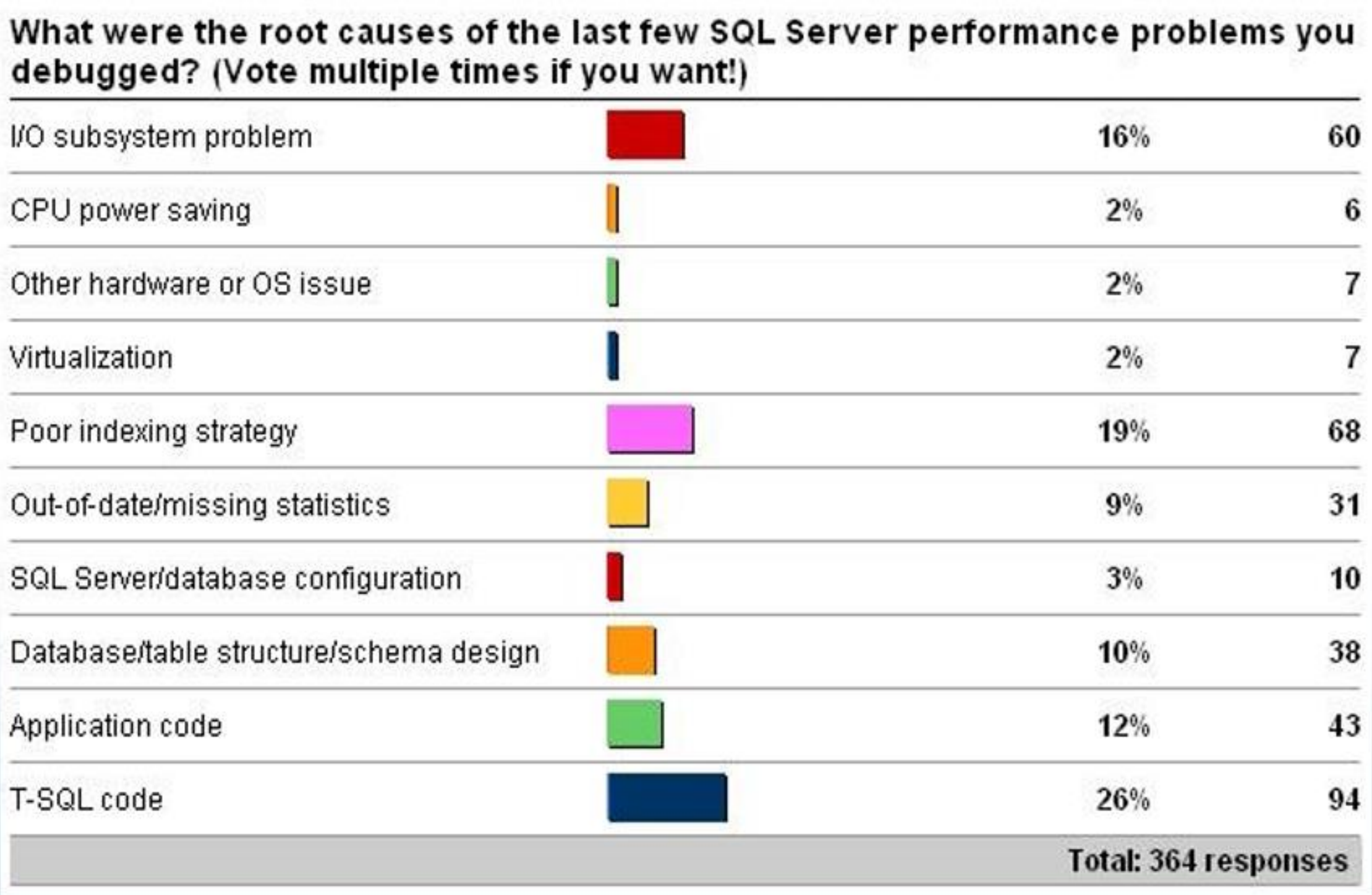
Microsoft Certified Azure Database Administrator Associate ★★

Microsoft Certified Azure Data Engineer Associate ★★

Microsoft Certified Azure Solutions Architect Expert ★★★

Common Cause of Performance Problems

http://www.sqlskills.com/blogs/paul/survey-results-common-causes-of-performance-problems/

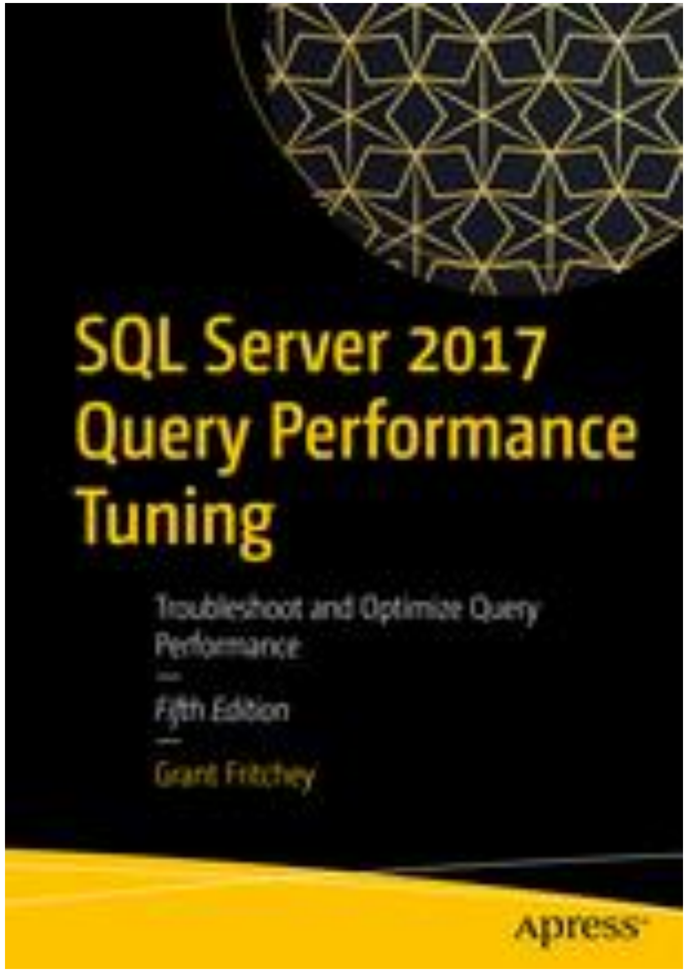


What were the root causes of the last few SQL Server performance problems you debugged? (Vote multiple times if you want!)

I/O subsystem problem	🟥	16%	60
CPU power saving	🟧	2%	6
Other hardware or OS issue	🟩	2%	7
Virtualization	🟦	2%	7
Poor indexing strategy	🟪	19%	68
Out-of-date/missing statistics	🟨	9%	31
SQL Server/database configuration	🟥	3%	10
Database/table structure/schema design	🟧	10%	38
Application code	🟩	12%	43
T-SQL code	🟦	26%	94
			Total: 364 responses

SQL Server Performance Killers

Poor Indexing
Inaccurate Statistics
Poor Query Design
Poor Execution Plans
Excessive blocking and deadlocks
Non set-based operations
Poor database design
Excessive fragmentation
Non-reusable execution plans
Frequent recompilation of queries
Improper use of cursors
Improper configuration of database log
Excessive use or improper configuration of tempdb



SQL Server 2017 Query Performance Tuning

Troubleshoot and Optimize Query Performance

Fifth Edition

Grant Fritchey

Apress

Two Main Functions of SQLOS

Management

- Memory Manager
- Process Scheduler
- Synchronization
- I/O
- Support for Non-Uniform Memory Access (NUMA) and Resource Governor

Monitoring

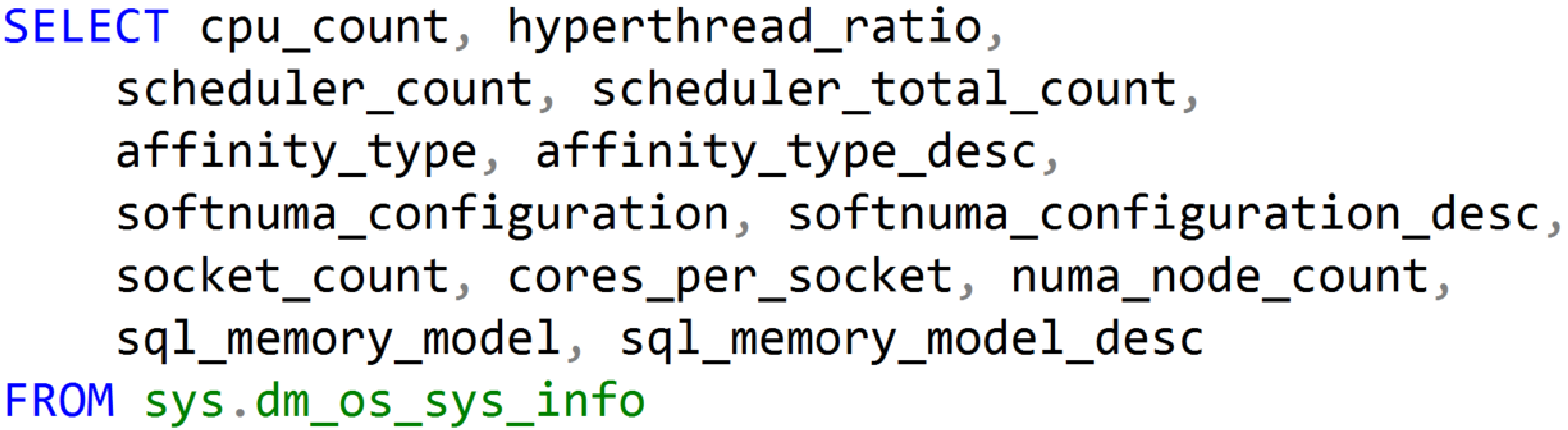
- Resource Monitor
- Deadlock Monitor
- Scheduler Monitor
- Lazy Writer (Buffer Pool management)
- Dynamic Management Views (DMVs)
- Extended Events
- Dedicated Administrator Connection (DAC)

Dynamic Management Views and Functions

Category	Description
sys.dm_exec_%	Execution and connection information
sys.dm_os_%	Operating system related information
sys.dm_tran_%	Transaction management information
sys.dm_io_%	I/O related information
sys.dm_db_%	Database information

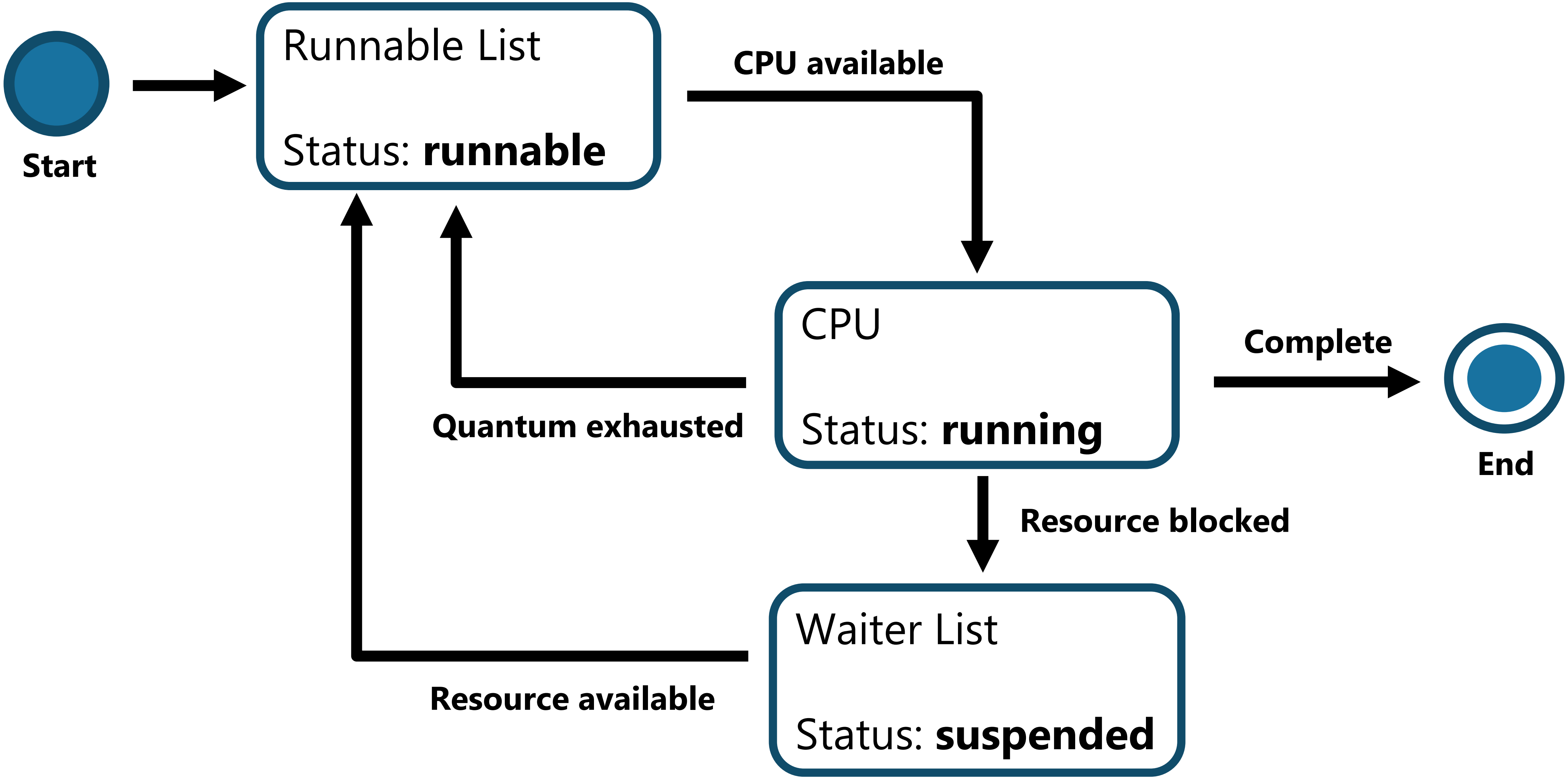
Using Dynamic Management Objects (DMOs)

- Must reference using the sys schema
- Two basic types:
  - Real-time state information
  - Historical information



```sql
SELECT cpu_count, hyperthread_ratio,
    scheduler_count, scheduler_total_count,
    affinity_type, affinity_type_desc,
    softnuma_configuration, softnuma_configuration_desc,
    socket_count, cores_per_socket, numa_node_count,
    sql_memory_model, sql_memory_model_desc
FROM sys.dm_os_sys_info
```

Yielding


Start
Runnable List
Status: **runnable**
CPU available
CPU
Status: **running**
Complete
End
Quantum exhausted
Resource blocked
Waiter List
Status: **suspended**
Resource available

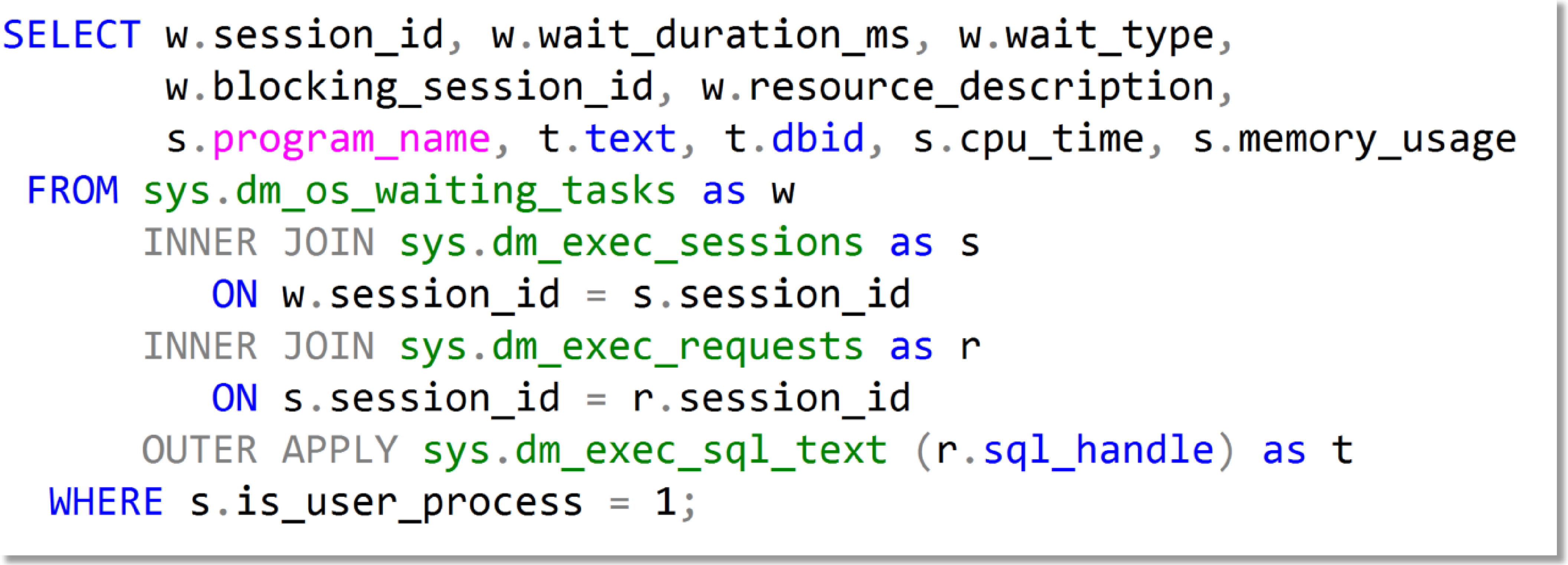
Thread States and Queues

Runnable: The thread is currently in the Runnable Queue waiting to execute. (First In, First Out).

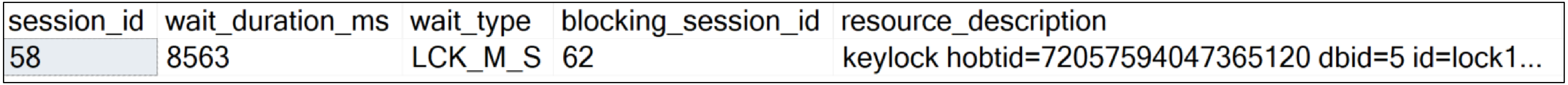
Running: One active thread executing on a processor.

Suspended: Placed on a Waiter List waiting for a resource other than a processor. (No specific order).

Waiting Tasks DMV



```sql
SELECT w.session_id, w.wait_duration_ms, w.wait_type,
       w.blocking_session_id, w.resource_description,
       s.program_name, t.text, t.dbid, s.cpu_time, s.memory_usage
 FROM sys.dm_os_waiting_tasks as w
      INNER JOIN sys.dm_exec_sessions as s
         ON w.session_id = s.session_id
      INNER JOIN sys.dm_exec_requests as r
         ON s.session_id = r.session_id
      OUTER APPLY sys.dm_exec_sql_text (r.sql_handle) as t
  WHERE s.is_user_process = 1;
```

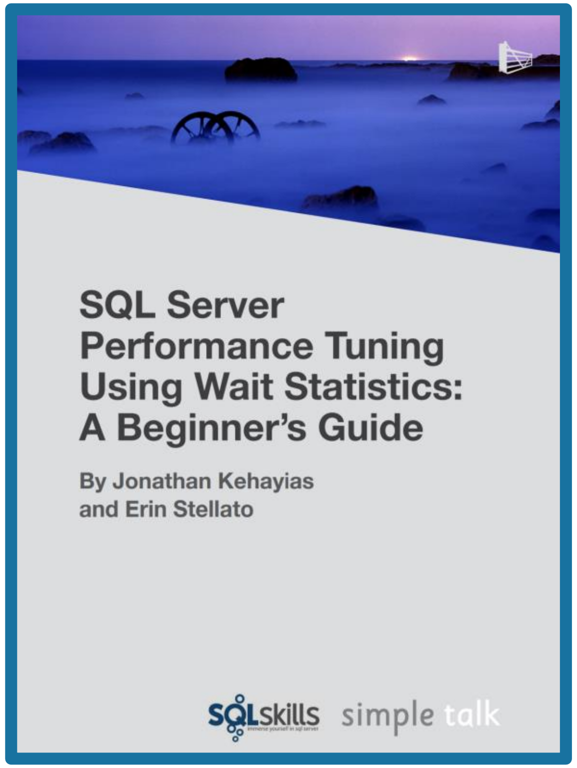


session_id	wait_duration_ms	wait_type	blocking_session_id	resource_description
58	8563	LCK_M_S	62	keylock hobtid=72057594047365120 dbid=5 id=lock1...

Troubleshooting Wait Types

Aaron Bertrand – Top Wait Types
https://sqlperformance.com/2018/10/sql-performance/top-wait-stats

Paul Randal – SQL Skills Wait Types Library
https://www.sqlskills.com/help/waits/



SQL Server Performance Tuning Using Wait Statistics: A Beginner's Guide

By Jonathan Kehayias and Erin Stellato

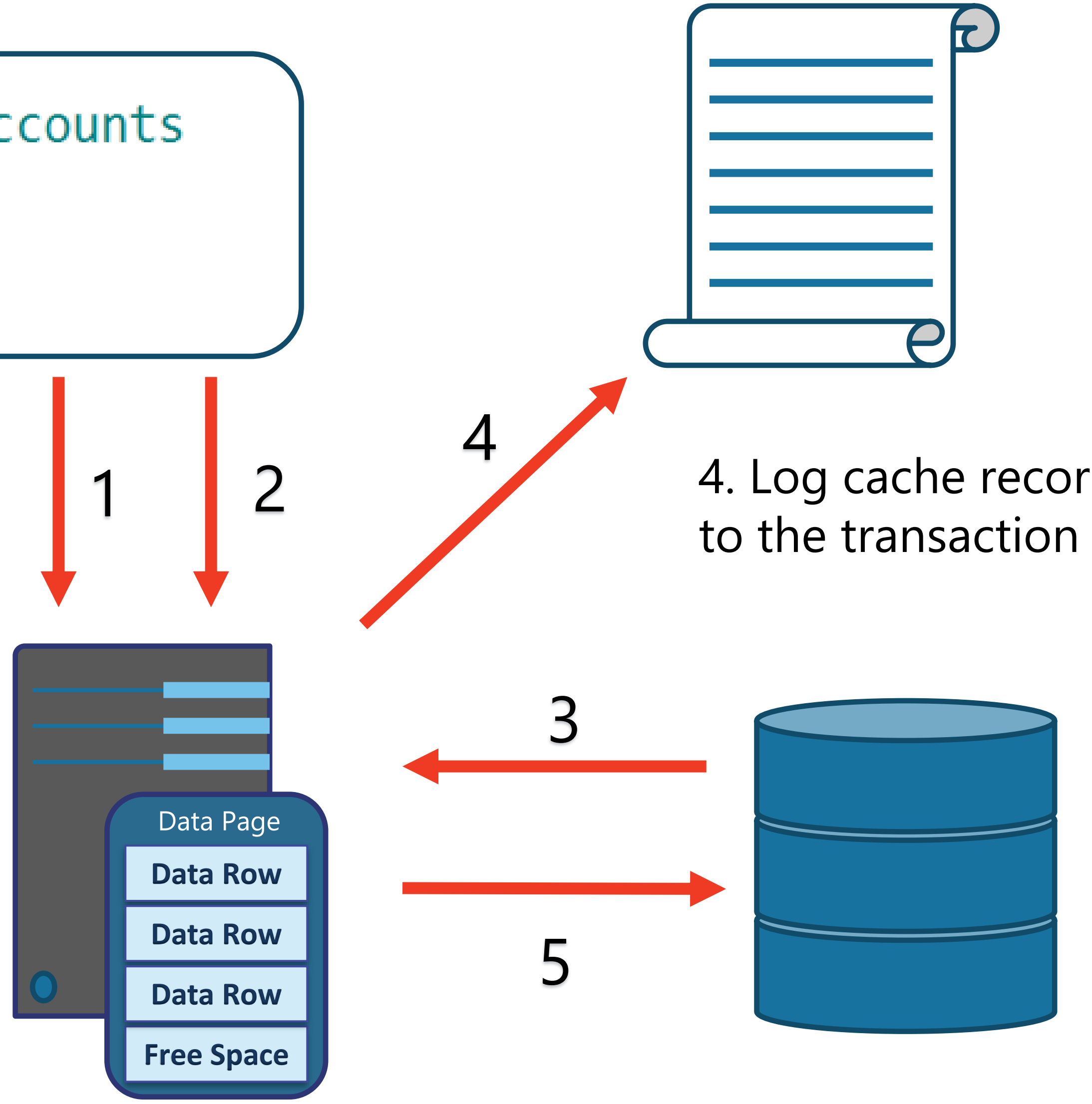
SQL Server Disk I/O (Write-Ahead Logging)

```
UPDATE Accounting.BankAccounts
SET Balance -= 200
WHERE AcctID = 1
```

1. Data modification is sent to buffer cache in memory.

2. Modification is recorded in the log cache.

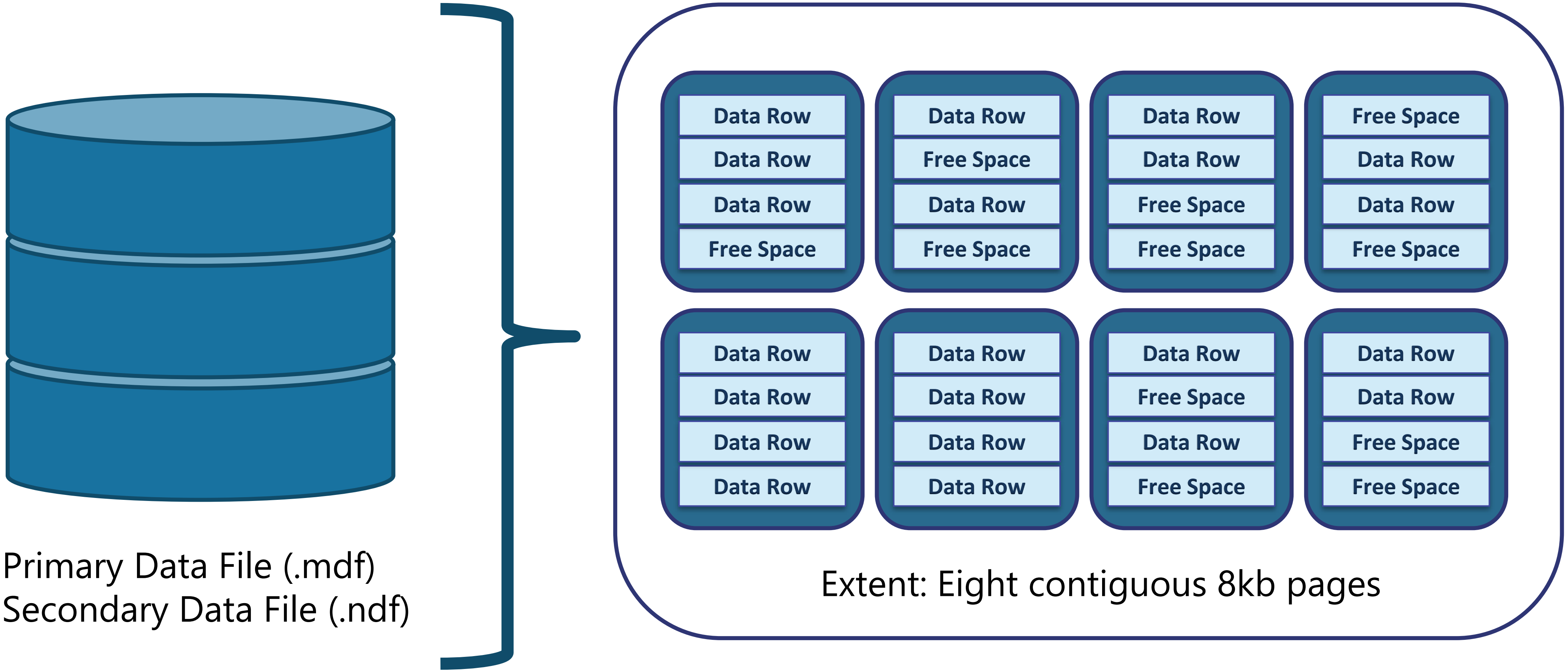
3. Data pages are located or read into the buffer cache and then modified.



4. Log cache record is flushed to the transaction log

5. At checkpoint, dirty data pages are written to the database file.

SQL Server Object Allocation



Primary Data File (.mdf)
Secondary Data File (.ndf)

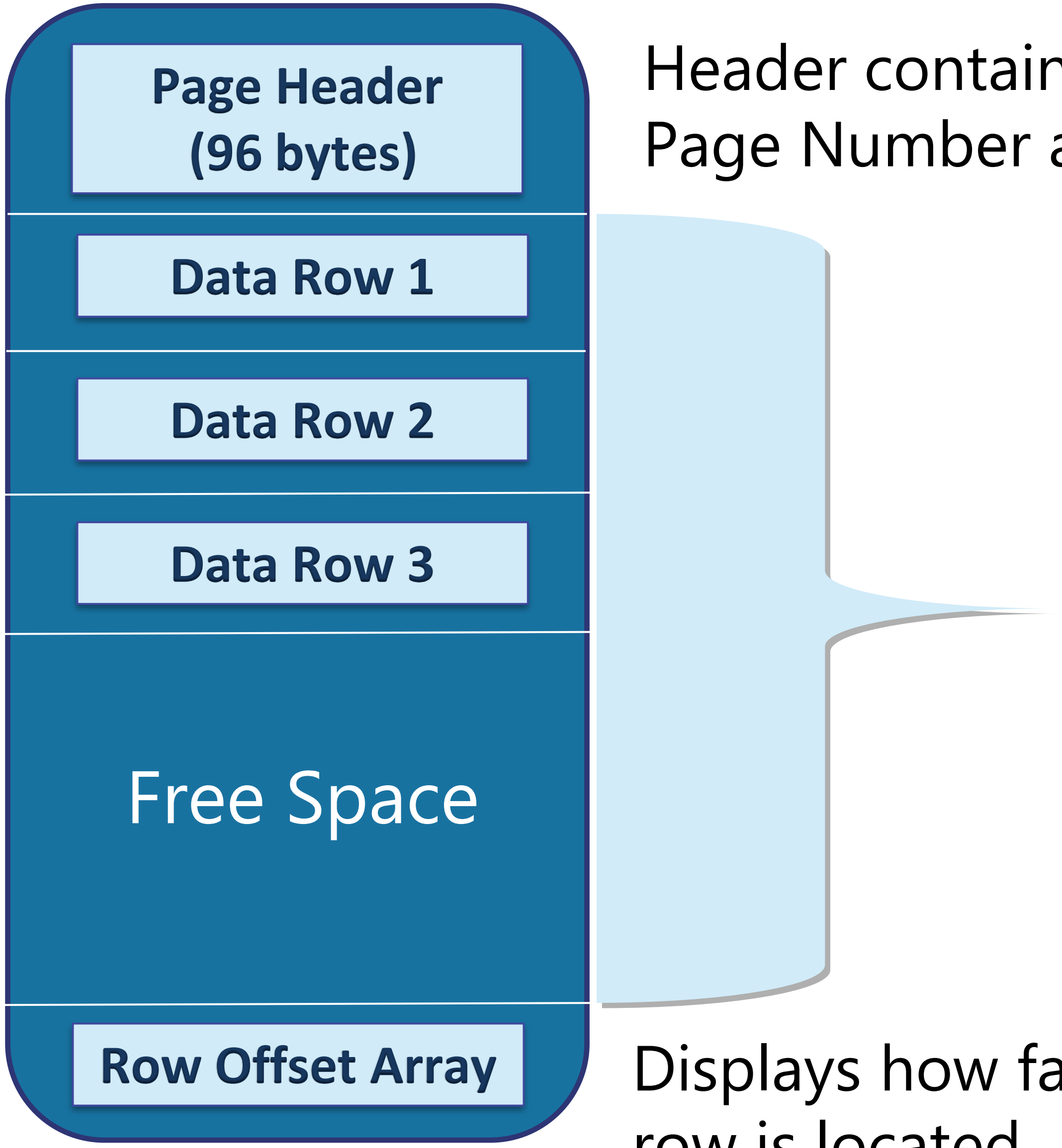
Data Row	Data Row	Data Row	Free Space
Data Row	Free Space	Data Row	Data Row
Data Row	Data Row	Free Space	Data Row
Free Space	Free Space	Free Space	Free Space
Data Row	Data Row	Data Row	Data Row
Data Row	Data Row	Free Space	Data Row
Data Row	Data Row	Data Row	Free Space
Data Row	Data Row	Free Space	Free Space

Extent: Eight contiguous 8kb pages

Uniform extents: Pages used by a single object.

Mixed extents: Pages used by different objects.

Basic Page Structure



Header contains Page Information such a Page Number and Page Type

8060 bytes for Data Rows or Free Space

Displays how far from beginning of page each row is located.

Allocation Units

IN_ROW_DATA

- Fixed length data must be store here.
- Rows cannot extend beyond pages
- Data Page is 8060 bytes

LOB_DATA (For out of row storage)

- varchar(max) / nvarchar(max) / varbinary(max)
- 16-byte point to out of row tree
- Uses text page to store a stream of data

ROW_OVERFLOW_DATA (SLOB)

- varchar(8000) / nvarchar(4000) / varbinary(8000)
- When a column can’t fit onto a page
- No control over which column overflows

Allocation Structures

Page Types
Data (1)
Index (2)
Text (3 and 4)
Boot (13)
File Header (15)
PFS (11)
GAM (8)
SGAM (9)
IAM (10)
DIFF_MAP(16)
ML_MAP (17)

Page Free Space (PFS)

- Tracks free space on a page (1 Byte/Page)
- Covers 64 megabytes (MB) worth of pages

Global Allocation Map (GAM)

- Tracks which extents have been allocated (1 Bit)
- Covers 64,000 extents (4 gigabytes (GB) worth of data)

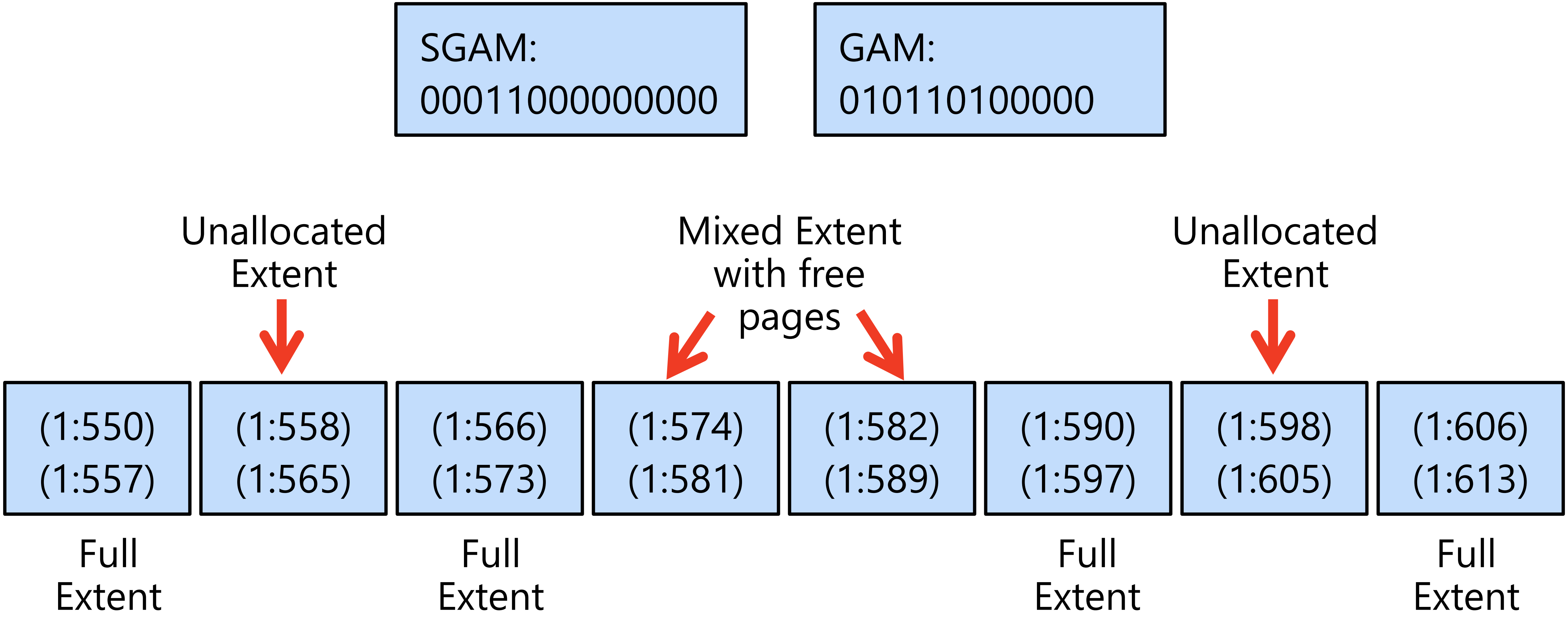
Shared Global Allocation Map (SGAM)

- Tracks which extents are used for mixed extent allocations (1 Bit)
- Covers 64,000 extents (4 GB worth of data)

Index Allocation Map (IAM)

- Tracks which extents are allocated to an allocation unit
- Covers 4 GB worth of data
- One IAM chain per table, per index, per partition, per allocation unit type

Index Allocation Map Pages



SGAM:
000110000000

GAM:
010110100000

Unallocated Extent

Mixed Extent with free pages

Unallocated Extent

(1:550) (1:557)	(1:558) (1:565)	(1:566) (1:573)	(1:574) (1:581)	(1:582) (1:589)	(1:590) (1:597)	(1:598) (1:605)	(1:606) (1:613)

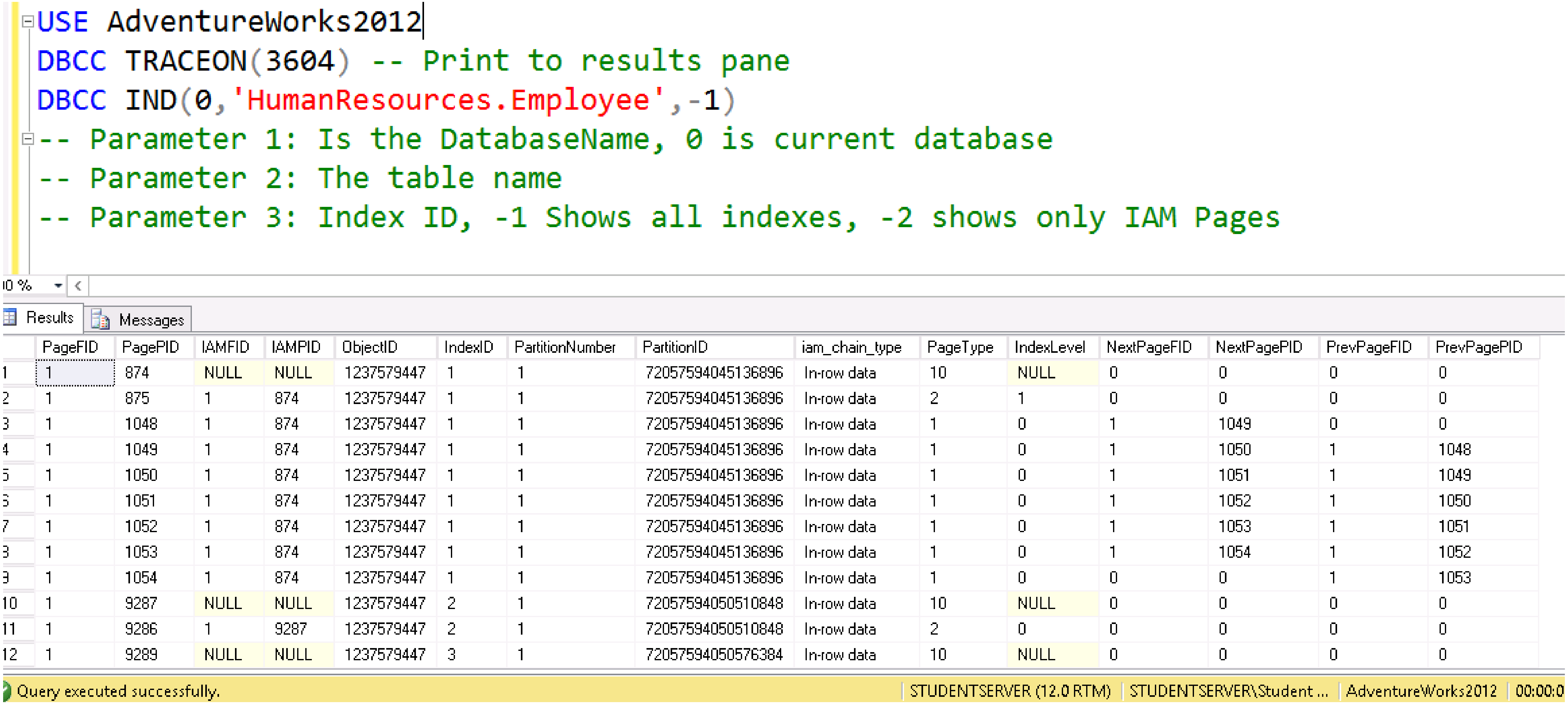
Full Extent

Full Extent

Full Extent

Full Extent

DBCC IND

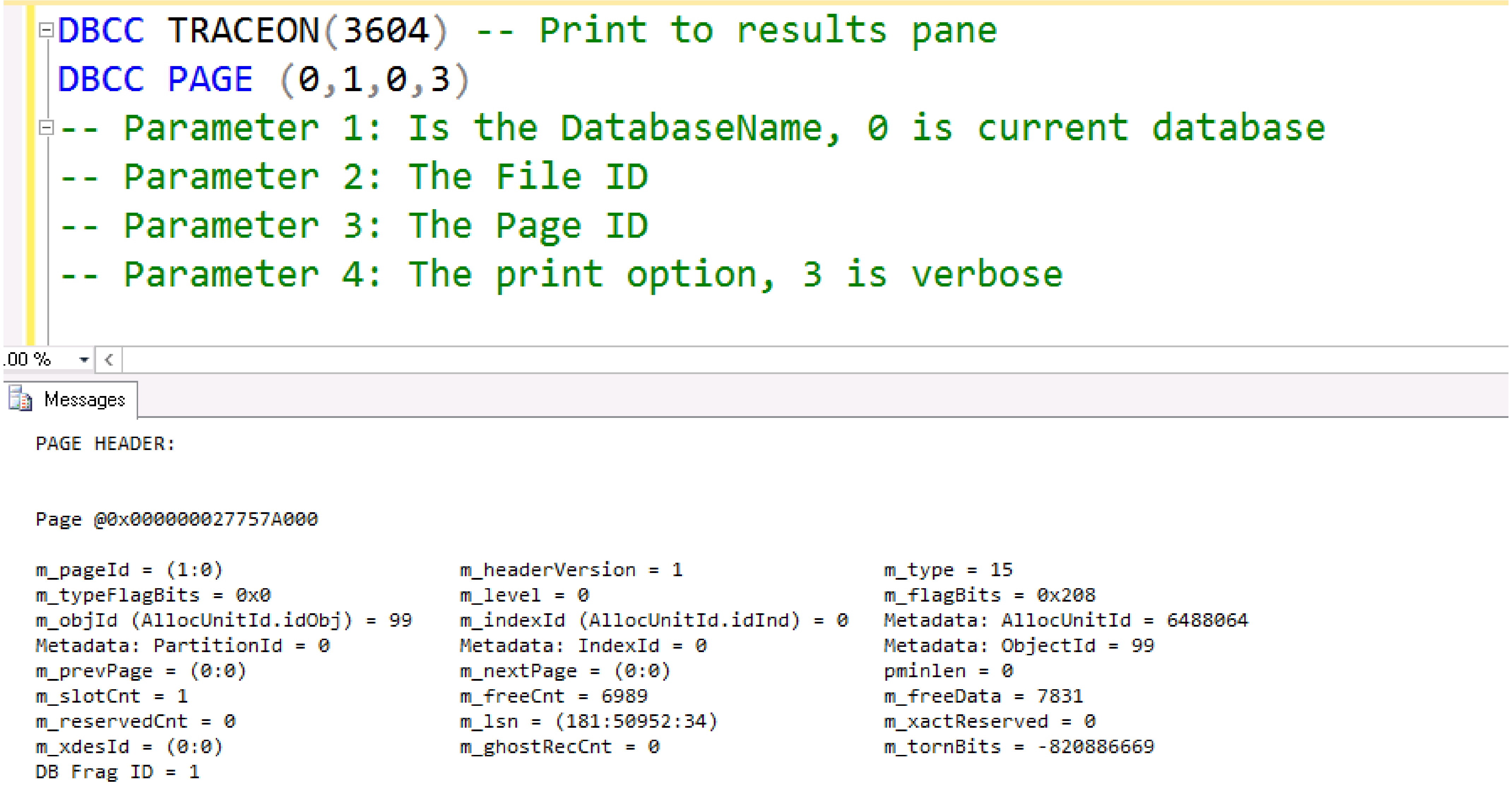


```
USE AdventureWorks2012
DBCC TRACEON(3604) -- Print to results pane
DBCC IND(0,'HumanResources.Employee',-1)
-- Parameter 1: Is the DatabaseName, 0 is current database
-- Parameter 2: The table name
-- Parameter 3: Index ID, -1 Shows all indexes, -2 shows only IAM Pages
```

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID	NextPagePID	PrevPageFID	PrevPagePID
1	1	874	NULL	NULL	1237579447	1	1	72057594045136896	In-row data	10	NULL	0	0	0	0
2	1	875	1	874	1237579447	1	1	72057594045136896	In-row data	2	1	0	0	0	0
3	1	1048	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	1	1049	0	0
4	1	1049	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	1	1050	1	1048
5	1	1050	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	1	1051	1	1049
6	1	1051	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	1	1052	1	1050
7	1	1052	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	1	1053	1	1051
8	1	1053	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	1	1054	1	1052
9	1	1054	1	874	1237579447	1	1	72057594045136896	In-row data	1	0	0	0	1	1053
10	1	9287	NULL	NULL	1237579447	2	1	72057594050510848	In-row data	10	NULL	0	0	0	0
11	1	9286	1	9287	1237579447	2	1	72057594050510848	In-row data	2	0	0	0	0	0
12	1	9289	NULL	NULL	1237579447	3	1	72057594050576384	In-row data	10	NULL	0	0	0	0

Query executed successfully. | STUDENTSERVER (12.0 RTM) | STUDENTSERVER\Student ... | AdventureWorks2012 | 00:00:0

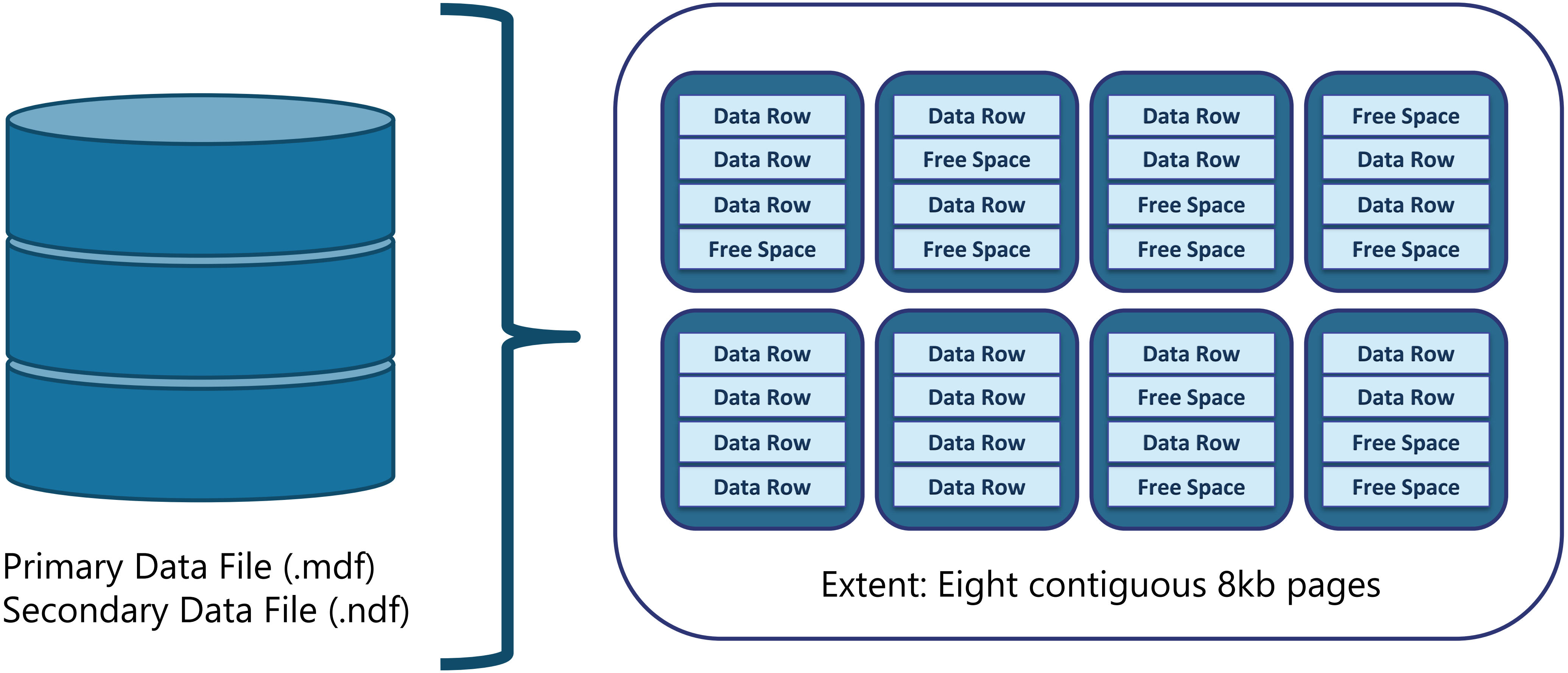
DBCC PAGE



```
DBCC TRACEON(3604) -- Print to results pane
DBCC PAGE (0,1,0,3)
-- Parameter 1: Is the DatabaseName, 0 is current database
-- Parameter 2: The File ID
-- Parameter 3: The Page ID
-- Parameter 4: The print option, 3 is verbose
```

```
PAGE HEADER:


Page @0x000000027757A000

m_pageId = (1:0)                    m_headerVersion = 1                 m_type = 15
m_typeFlagBits = 0x0                m_level = 0                         m_flagBits = 0x208
m_objId (AllocUnitId.idObj) = 99    m_indexId (AllocUnitId.idInd) = 0   Metadata: AllocUnitId = 6488064
Metadata: PartitionId = 0           Metadata: IndexId = 0               Metadata: ObjectId = 99
m_prevPage = (0:0)                  m_nextPage = (0:0)                  pminlen = 0
m_slotCnt = 1                       m_freeCnt = 6989                    m_freeData = 7831
m_reservedCnt = 0                   m_lsn = (181:50952:34)              m_xactReserved = 0
m_xdesId = (0:0)                    m_ghostRecCnt = 0                   m_tornBits = -820886669
DB Frag ID = 1
```

How Data is stored in a Database



Primary Data File (.mdf)
Secondary Data File (.ndf)

Data Row	Data Row	Data Row	Free Space
Data Row	Free Space	Data Row	Data Row
Data Row	Data Row	Free Space	Data Row
Free Space	Free Space	Free Space	Free Space

Data Row	Data Row	Data Row	Data Row
Data Row	Data Row	Free Space	Data Row
Data Row	Data Row	Data Row	Free Space
Data Row	Data Row	Free Space	Free Space

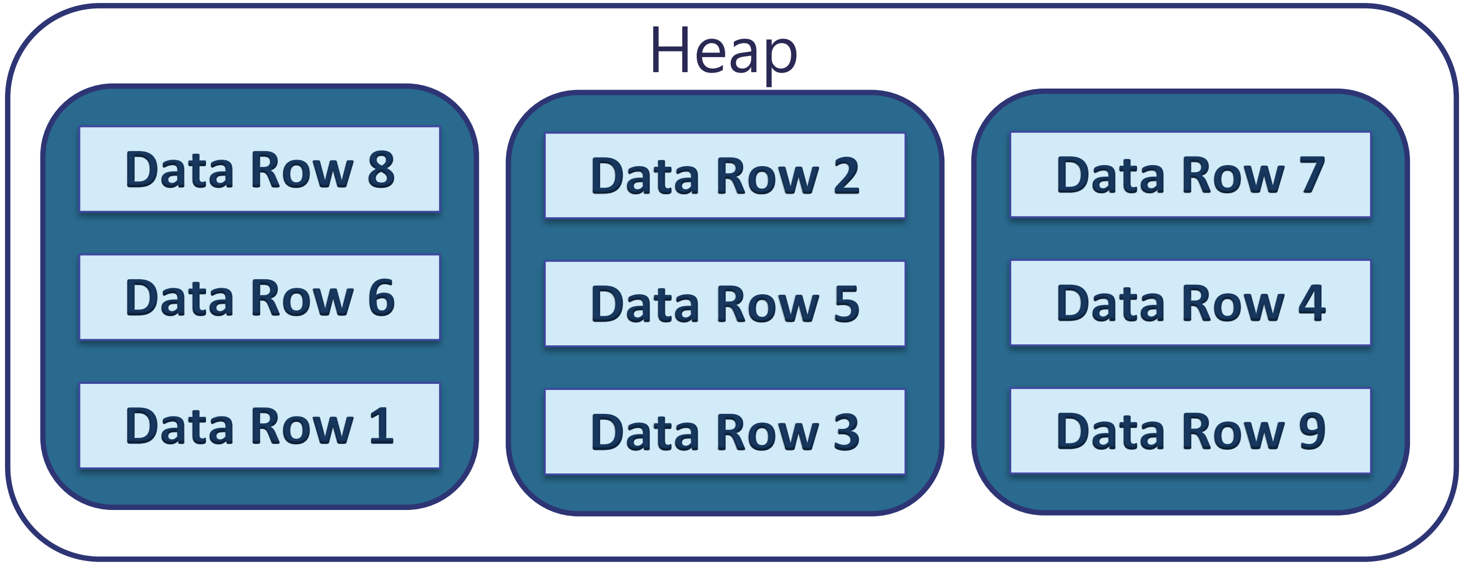
Extent: Eight contiguous 8kb pages

Uniform extents: Pages used by a single object.

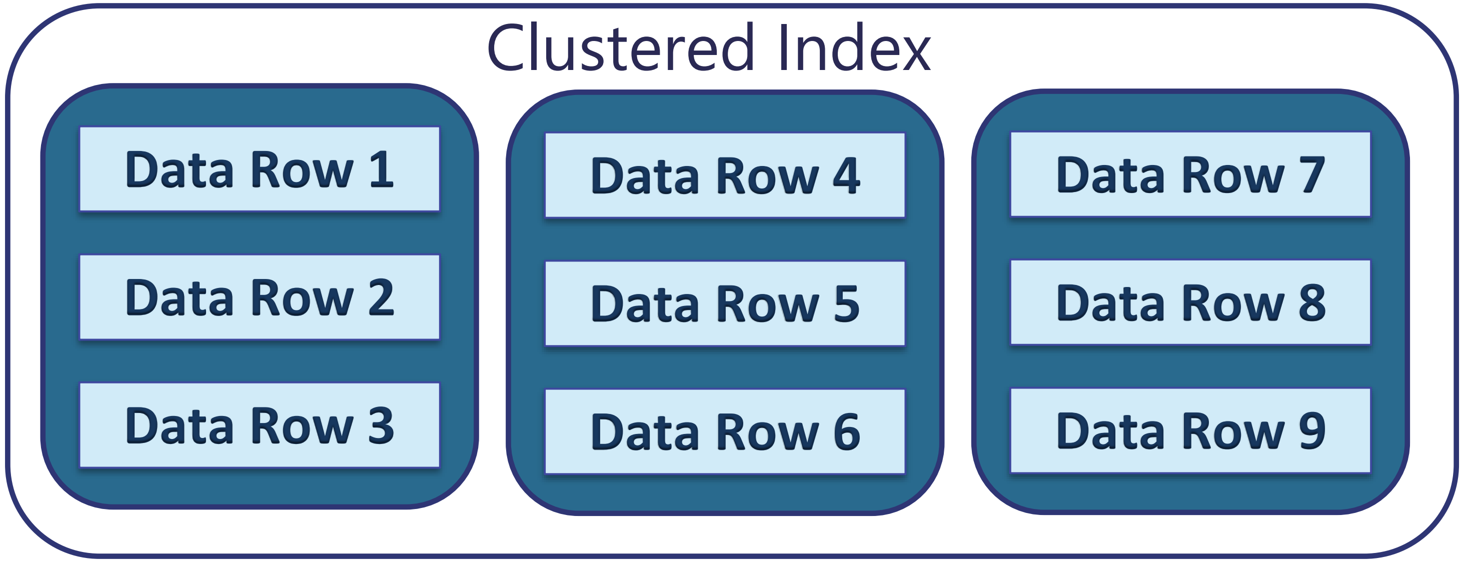
Mixed extents: Pages used by different objects.

How Data is Stored in Data Pages

Data stored in a Heap is not stored in any order and normally does not have a Primary Key.



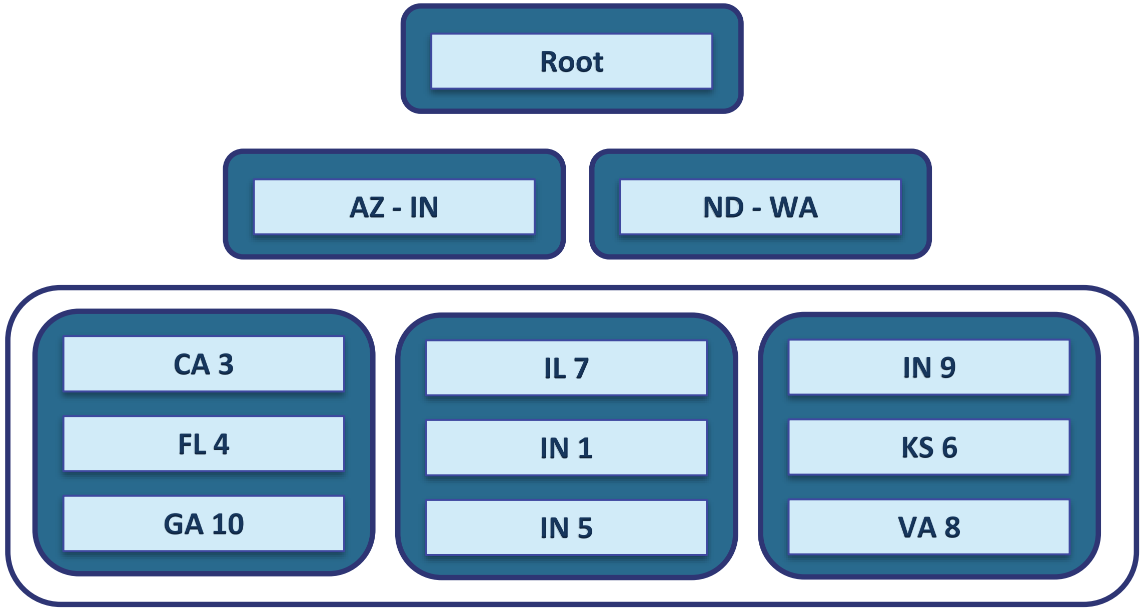
Clustered Index data is stored in sorted order by the Clustering key. In many cases, this is the same value as the Primary Key.



Key or Rid Lookup

A Non–Clustered Index is built separate from the table.

The leaf level will have RID or Key values to lookup additional columns.



Root

AZ - IN

ND - WA

CA 3

FL 4

GA 10

IL 7

IN 1

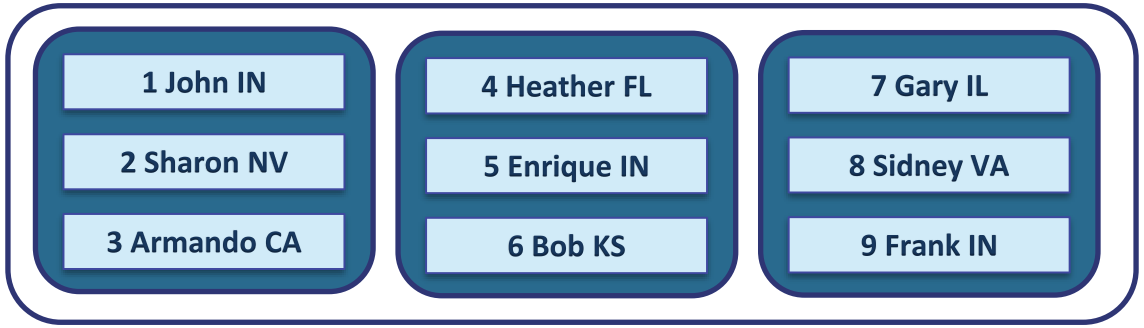
IN 5

IN 9

KS 6

VA 8

The data is stored in either a Clustered Index (sorted) or Heap (unsorted) table structure



1 John IN

2 Sharon NV

3 Armando CA

4 Heather FL

5 Enrique IN

6 Bob KS

7 Gary IL

8 Sidney VA

9 Frank IN

Characteristics of a Good Clustering Key

Narrow

- Use a data type with a small number of bytes to conserver space in tables and indexes

Unique

- To avoid SQL adding a 4-byte uniquifier

Static

- Allows data to stay constant without constant changes which could lead to page splits

Increasing

- Allows better write performance and reduces fragmentation issues

How to determine Thread Stack Memory

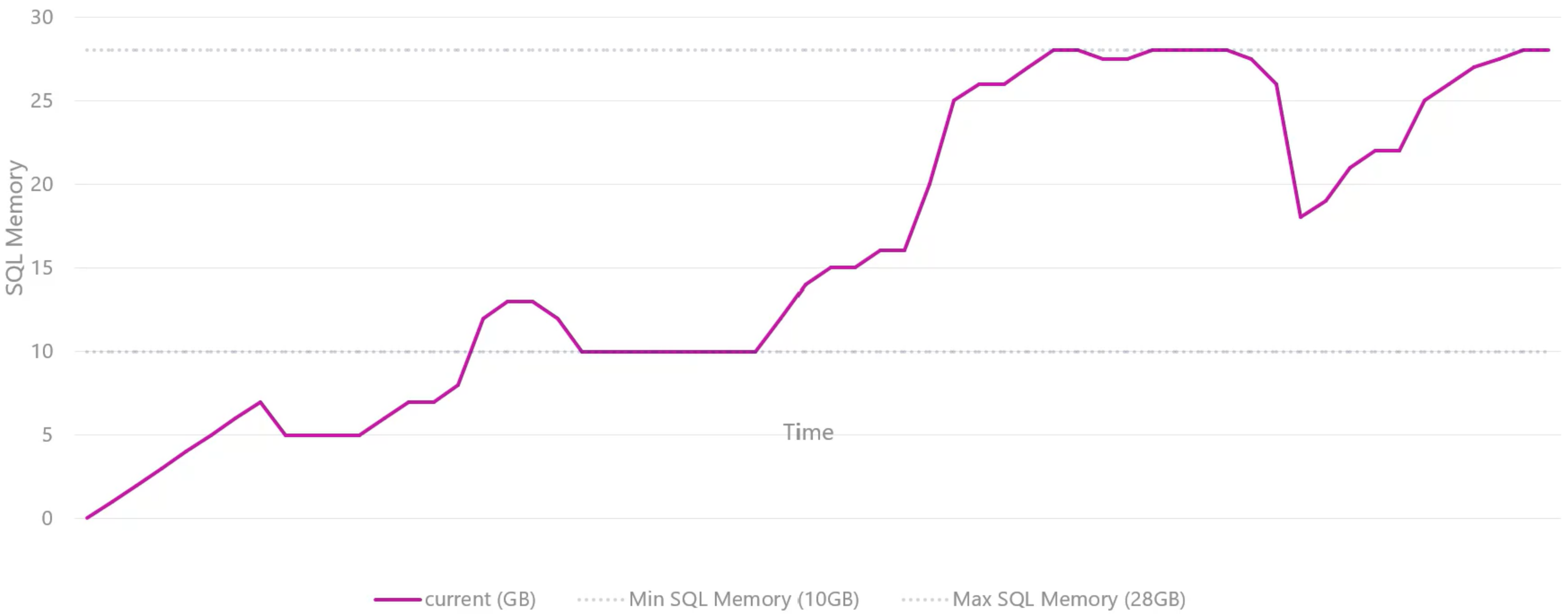
Maximum Worker Threads
512 + (Processors -4) *16

*

2mb per thread

Cores	Threads	Memory (MB)
4	512	1,024
8	576	1,152
16	704	1,408
32	960	1,920
64	1,472	2,944
80	1,728	3,456

Dynamic Memory Management

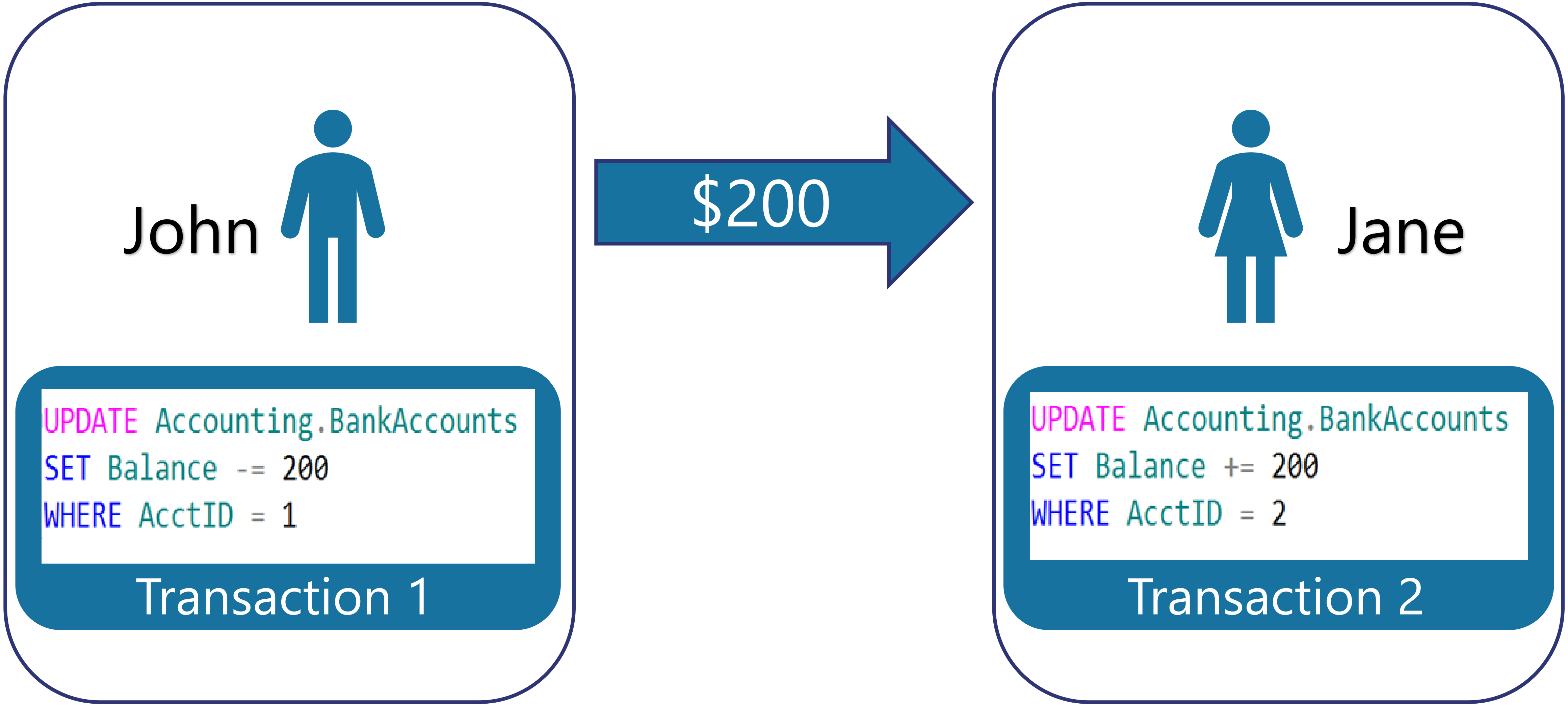


What is a Transaction?

A transaction is a series of one or more statements that need to operate as a single logical unit of work.

To qualify as a transaction, the logical unit of work must possess all four of the ACID properties.

Logical Units of Work – Auto Commit Transactions



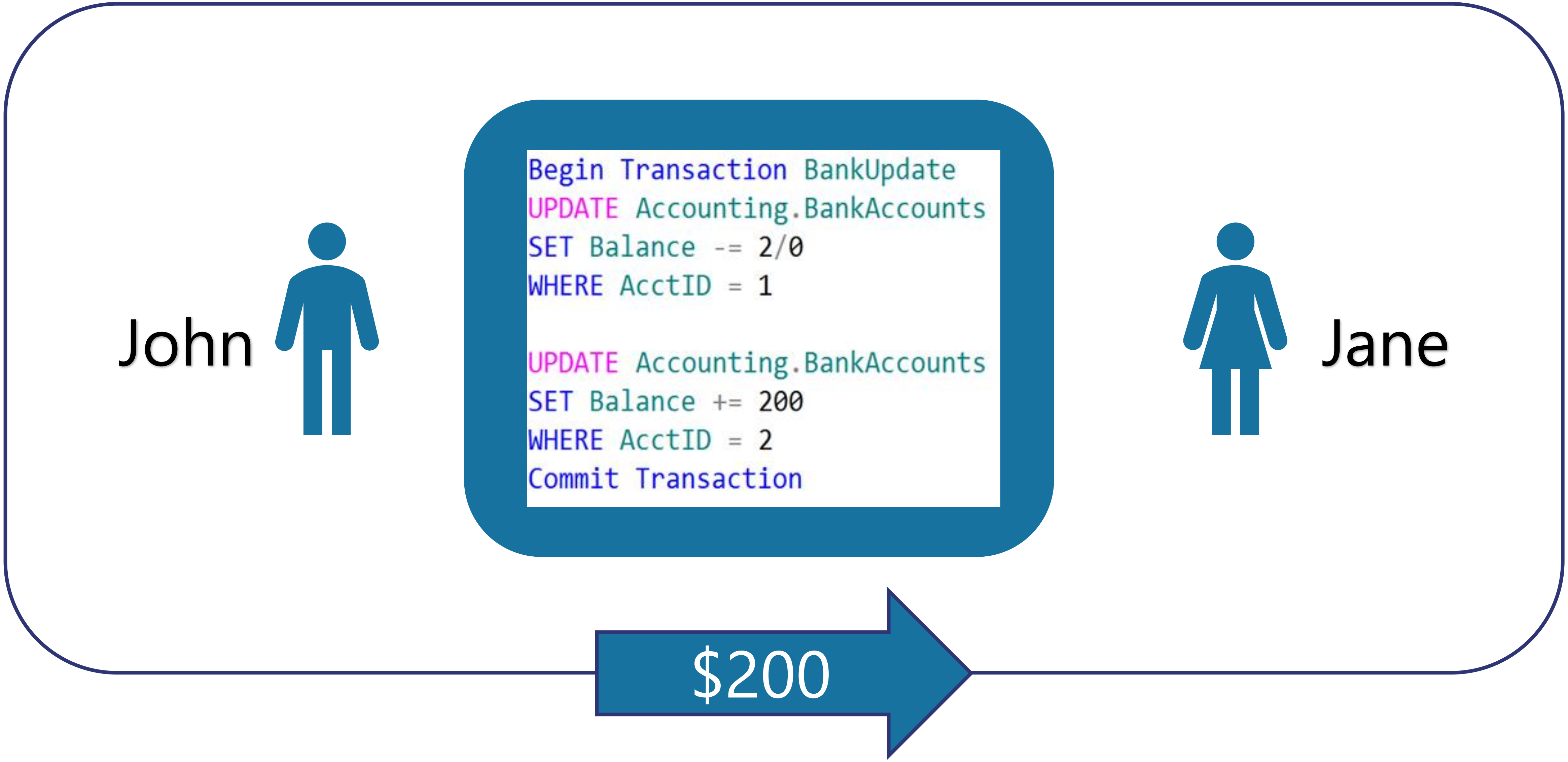
```sql
UPDATE Accounting.BankAccounts
SET Balance -= 200
WHERE AcctID = 1
```

Transaction 1

```sql
UPDATE Accounting.BankAccounts
SET Balance += 200
WHERE AcctID = 2
```

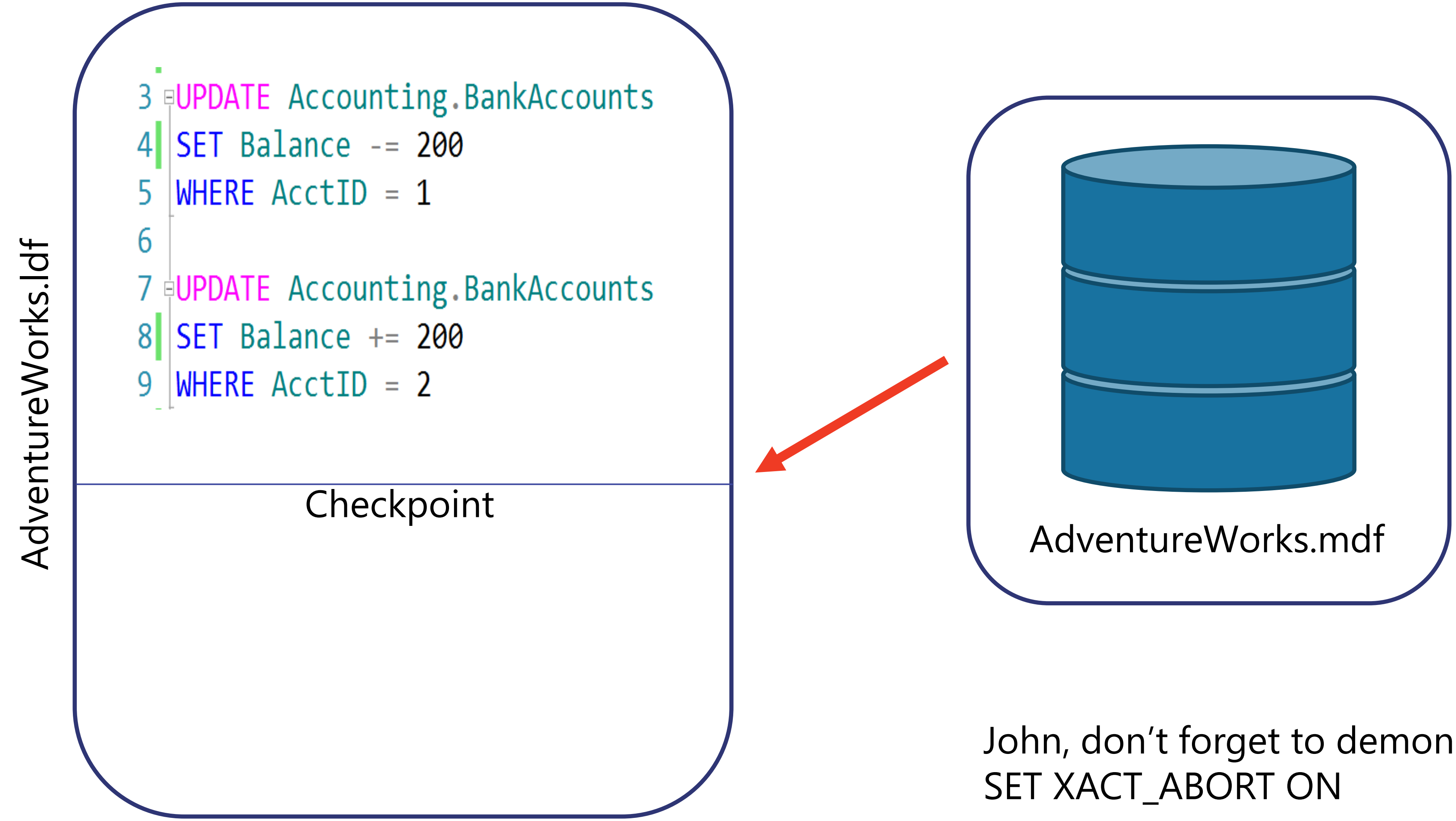
Transaction 2

Single Logical Unit of Work – Explicit Transactions



```
Begin Transaction BankUpdate
UPDATE Accounting.BankAccounts
SET Balance -= 2/0
WHERE AcctID = 1

UPDATE Accounting.BankAccounts
SET Balance += 200
WHERE AcctID = 2
Commit Transaction
```

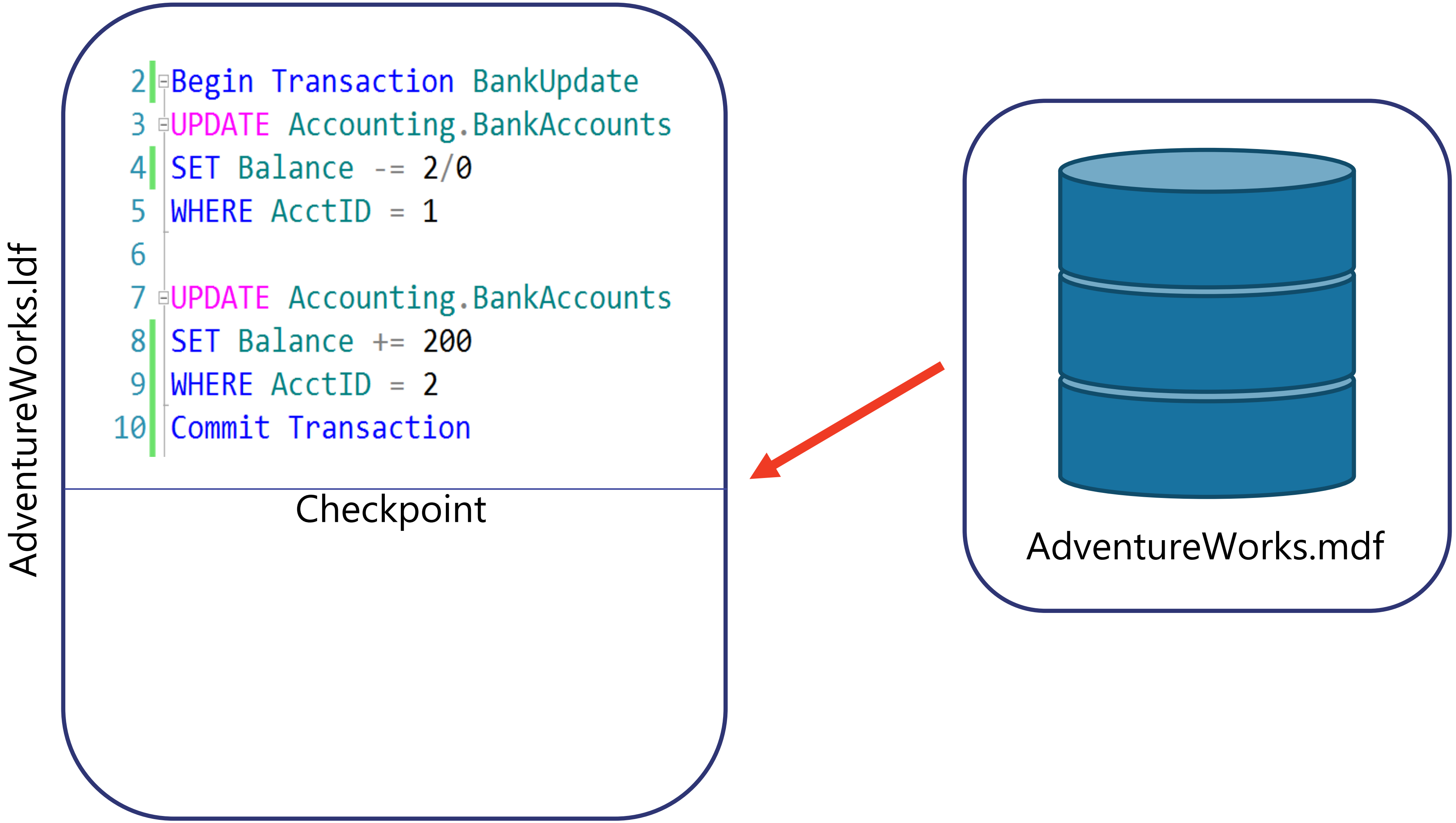
Auto-Commit Transactions without Error Handling



```
UPDATE Accounting.BankAccounts
SET Balance -= 200
WHERE AcctID = 1

UPDATE Accounting.BankAccounts
SET Balance += 200
WHERE AcctID = 2
```

John, don’t forget to demonstrate
SET XACT_ABORT ON

Explicit Transactions without Error Handling



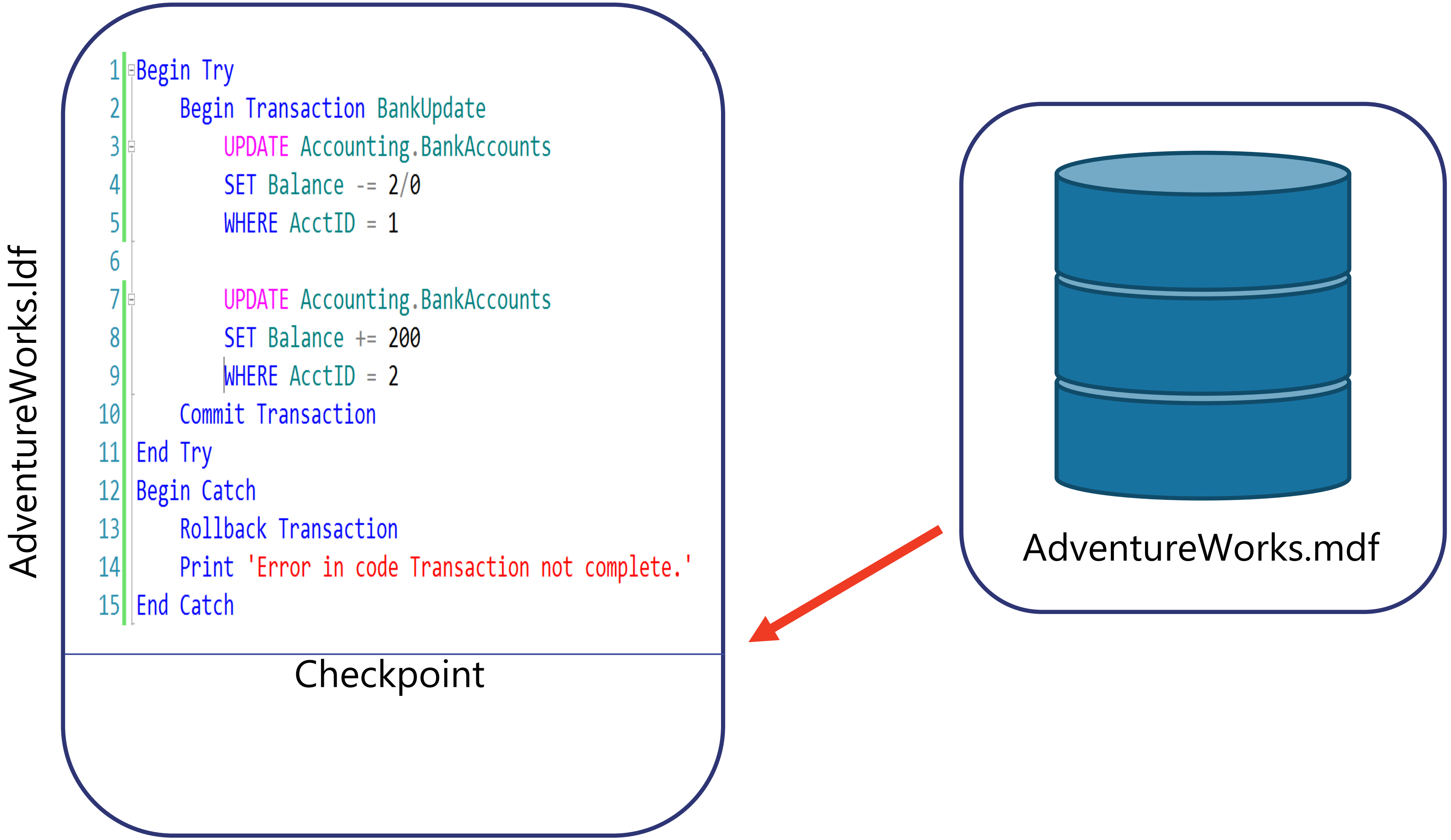
AdventureWorks.ldf

```
Begin Transaction BankUpdate
UPDATE Accounting.BankAccounts
SET Balance -= 2/0
WHERE AcctID = 1

UPDATE Accounting.BankAccounts
SET Balance += 200
WHERE AcctID = 2
Commit Transaction
```

Checkpoint

AdventureWorks.mdf

Explicit Transactions with Error Handling



AdventureWorks.ldf

```
Begin Try
    Begin Transaction BankUpdate
        UPDATE Accounting.BankAccounts
        SET Balance -= 2/0
        WHERE AcctID = 1

        UPDATE Accounting.BankAccounts
        SET Balance += 200
        WHERE AcctID = 2
    Commit Transaction
End Try
Begin Catch
    Rollback Transaction
    Print 'Error in code Transaction not complete.'
End Catch
```

Checkpoint

AdventureWorks.mdf

Transactions must pass the ACID test

Atomicity – All or Nothing

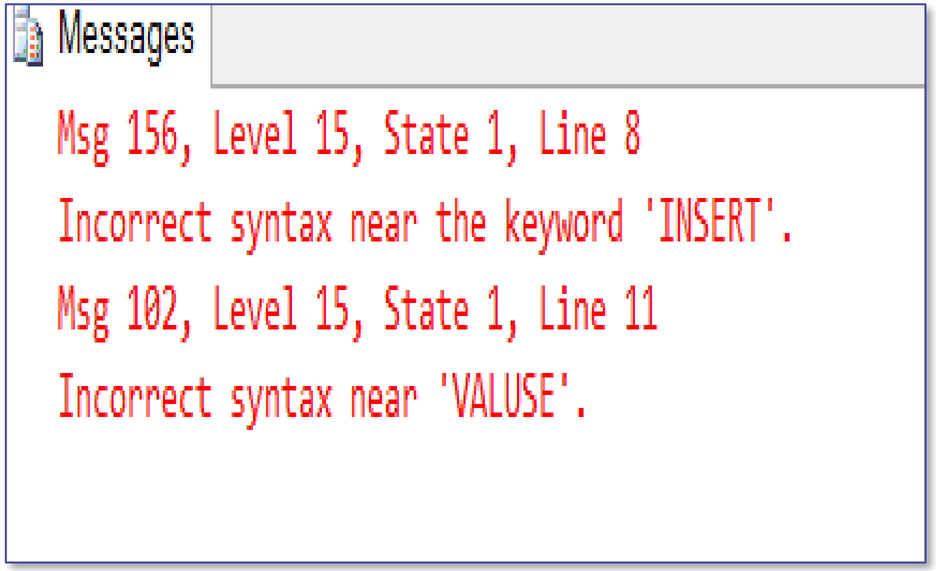
Consistent – Only valid data

Isolated – No interference

Durable – Data is recoverable

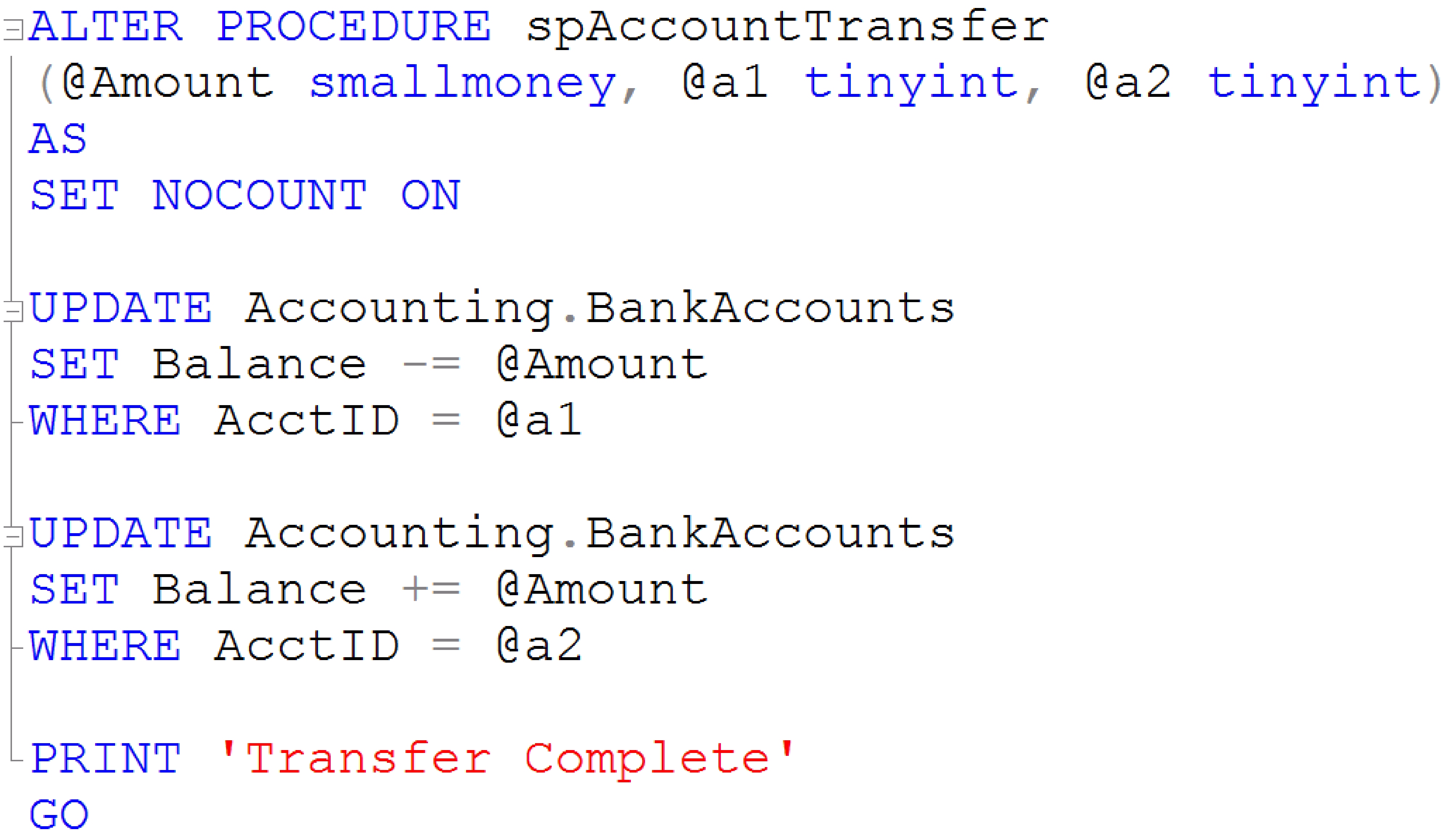
Working with Transactions

```
CREATE SCHEMA Accounting Authorization dbo
CREATE TABLE BankAccounts
 (AcctID int IDENTITY,
  AcctName char(15),
  Balance money,
  ModifiedDate date)

INSERT INTO Accounting.BankAccounts
VALUES('John',500, GETDATE())
INSERT INTO Accounting.BankAccounts
VALUSE('Jane', 750, GETDATE())
```



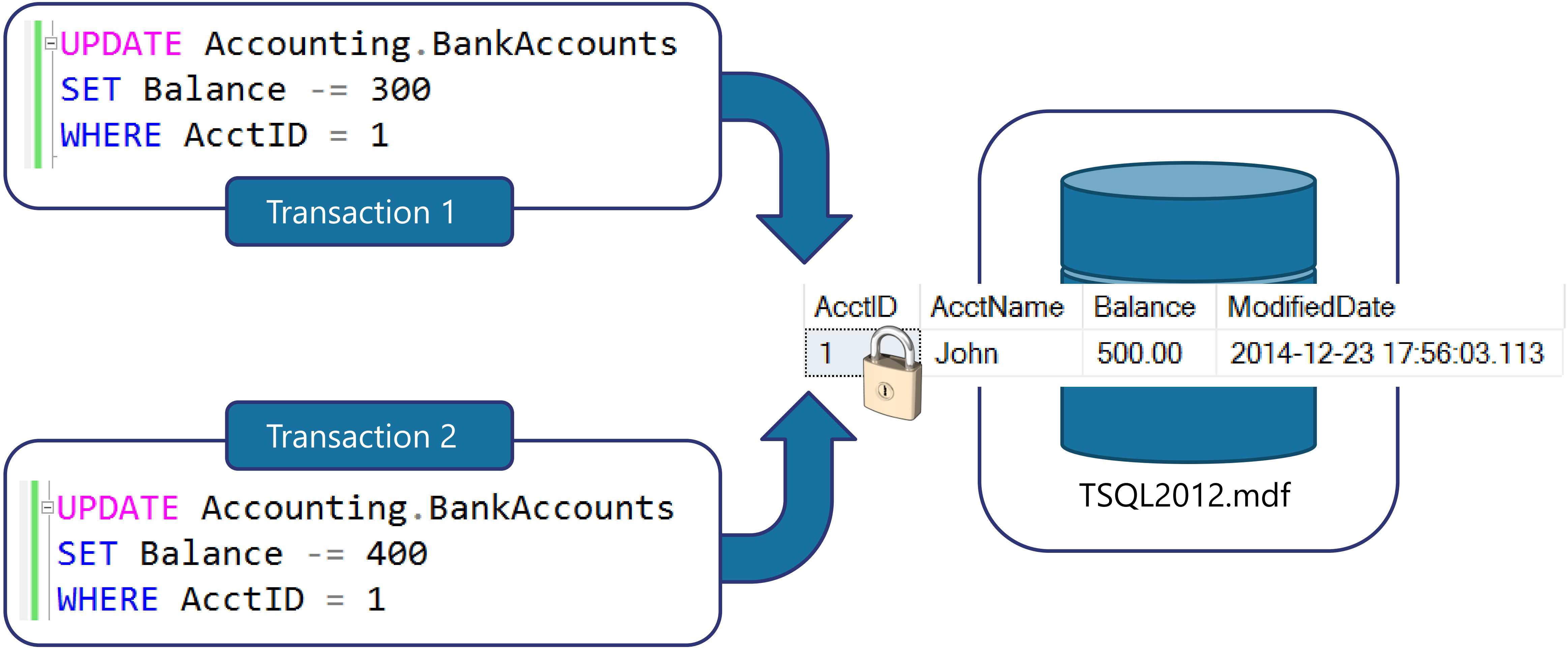
```
Msg 156, Level 15, State 1, Line 8
Incorrect syntax near the keyword 'INSERT'.
Msg 102, Level 15, State 1, Line 11
Incorrect syntax near 'VALUSE'.
```

Creating Stored Procedures



```
ALTER PROCEDURE spAccountTransfer
(@Amount smallmoney, @a1 tinyint, @a2 tinyint)
AS
SET NOCOUNT ON

UPDATE Accounting.BankAccounts
SET Balance -= @Amount
WHERE AcctID = @a1

UPDATE Accounting.BankAccounts
SET Balance += @Amount
WHERE AcctID = @a2

PRINT 'Transfer Complete'
GO
```

What is a Lock?



```sql
UPDATE Accounting.BankAccounts
SET Balance -= 300
WHERE AcctID = 1
```

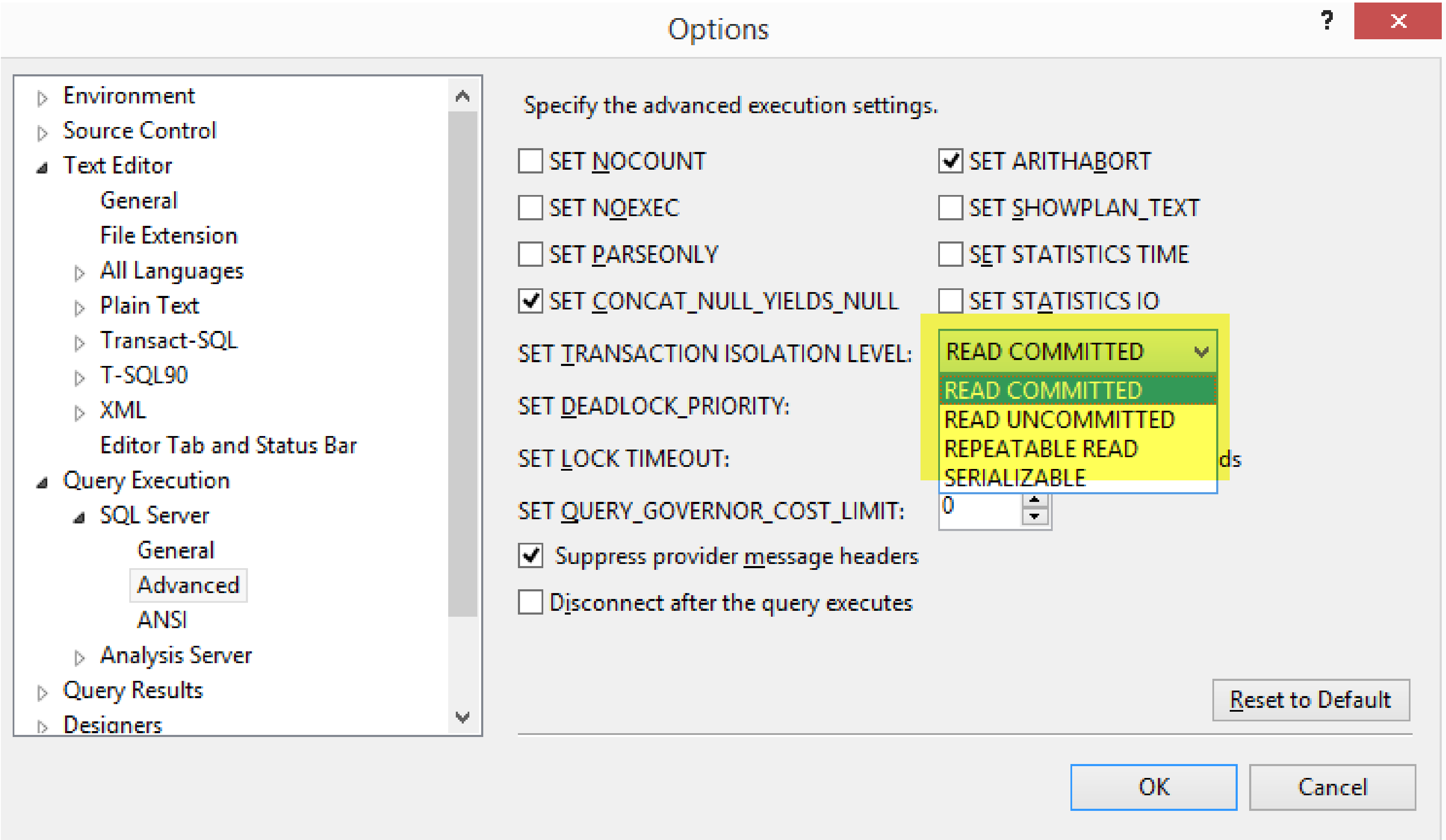
```sql
UPDATE Accounting.BankAccounts
SET Balance -= 400
WHERE AcctID = 1
```

AcctID	AcctName	Balance	ModifiedDate
1	John	500.00	2014-12-23 17:56:03.113

Transaction Isolation Levels

Isolation Level	Dirty Read	Lost Update	Nonrepeatable Read	Phantoms
Read uncommitted	Yes	Yes	Yes	Yes
Read committed (default)	No	Yes	Yes	Yes
Repeatable read	No	No	No	Yes
Serializable	No	No	No	No
Snapshot	No	No	No	No

Isolation Levels



Specify the advanced execution settings.

□ SET NOCOUNT
□ SET NOEXEC
□ SET PARSEONLY
☑ SET CONCAT_NULL_YIELDS_NULL
☑ SET ARITHABORT
□ SET SHOWPLAN_TEXT
□ SET STATISTICS TIME
□ SET STATISTICS IO

SET TRANSACTION ISOLATION LEVEL: READ COMMITTED

- READ COMMITTED
- READ UNCOMMITTED
- REPEATABLE READ
- SERIALIZABLE

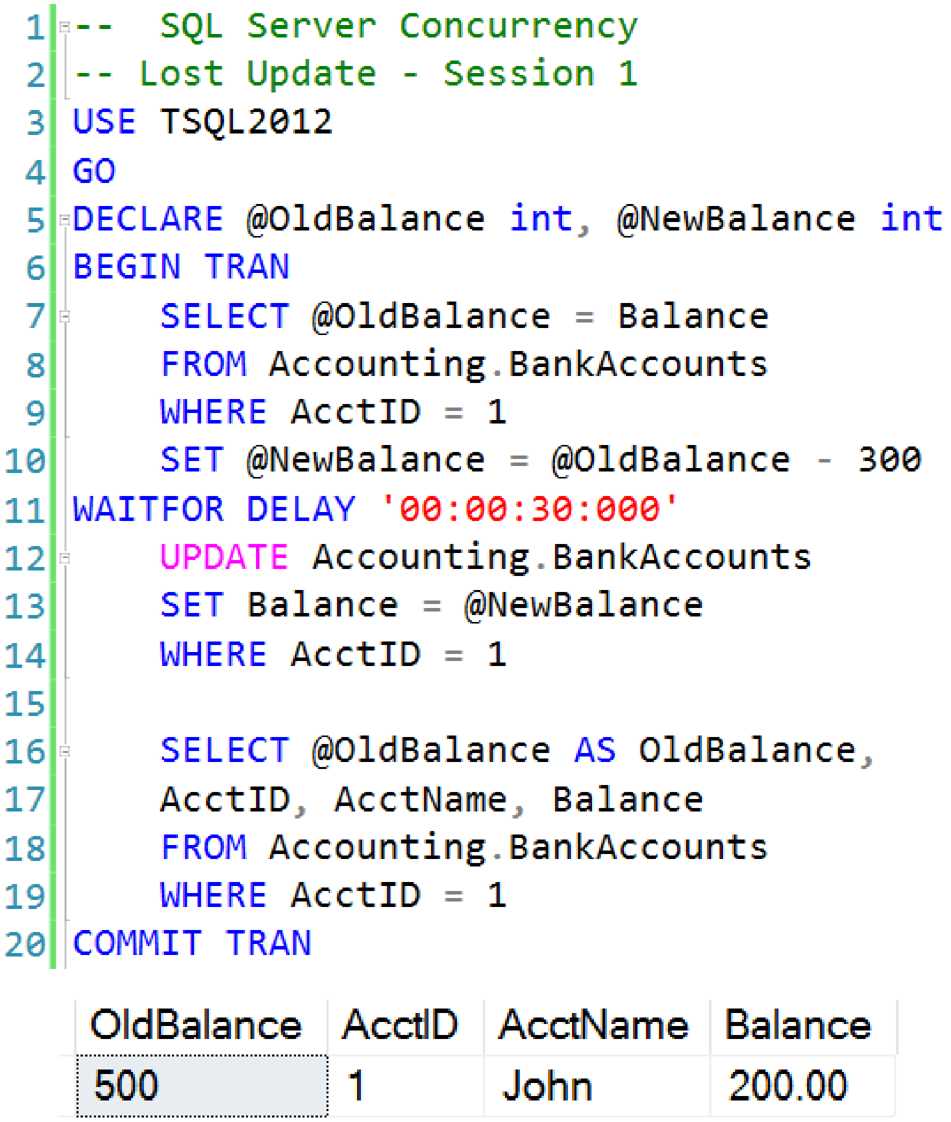
SET DEADLOCK_PRIORITY:

SET LOCK TIMEOUT: ds

SET QUERY_GOVERNOR_COST_LIMIT: 0

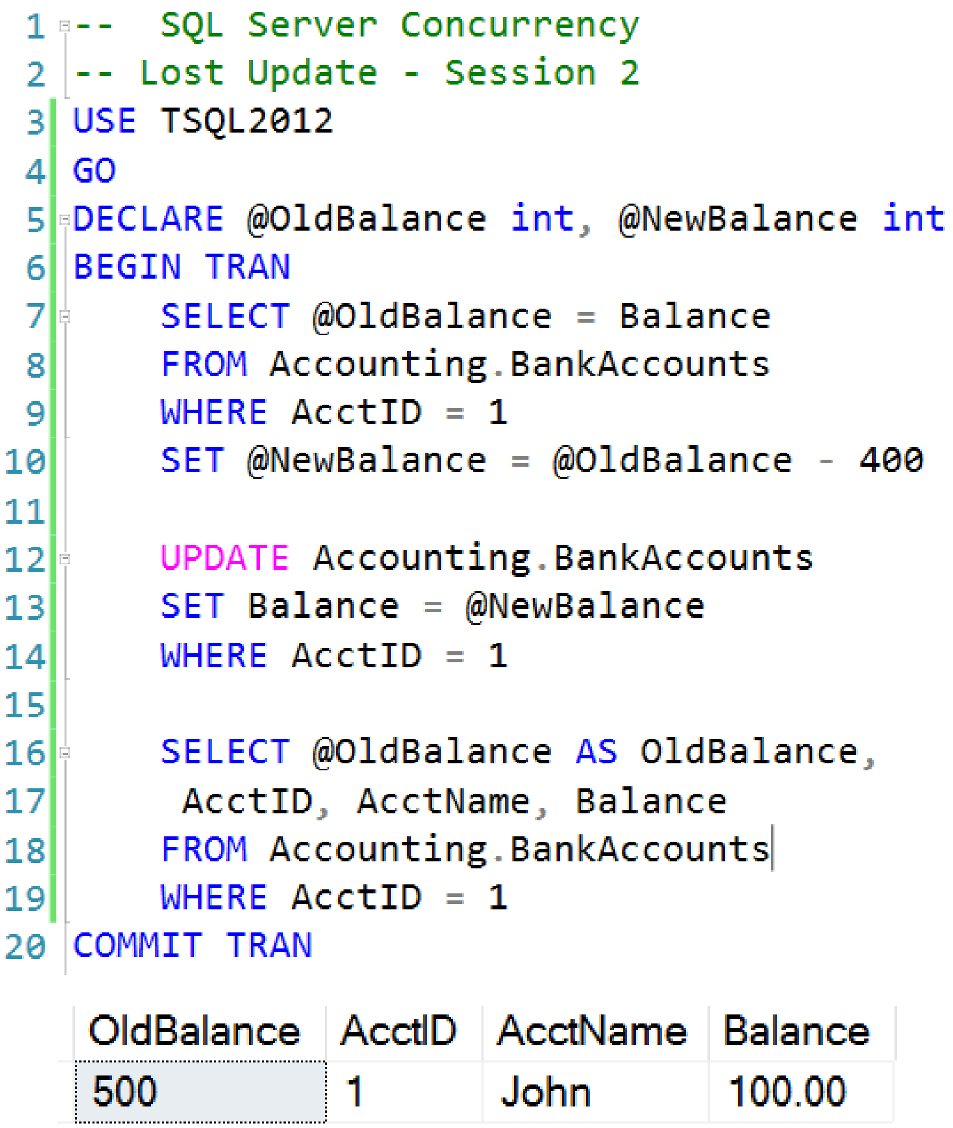
☑ Suppress provider message headers
□ Disconnect after the query executes

Lost Updates



```
--  SQL Server Concurrency
-- Lost Update - Session 1
USE TSQL2012
GO
DECLARE @OldBalance int, @NewBalance int
BEGIN TRAN
    SELECT @OldBalance = Balance
    FROM Accounting.BankAccounts
    WHERE AcctID = 1
    SET @NewBalance = @OldBalance - 300
WAITFOR DELAY '00:00:30:000'
    UPDATE Accounting.BankAccounts
    SET Balance = @NewBalance
    WHERE AcctID = 1

    SELECT @OldBalance AS OldBalance,
    AcctID, AcctName, Balance
    FROM Accounting.BankAccounts
    WHERE AcctID = 1
COMMIT TRAN
```

OldBalance	AcctID	AcctName	Balance
500	1	John	200.00



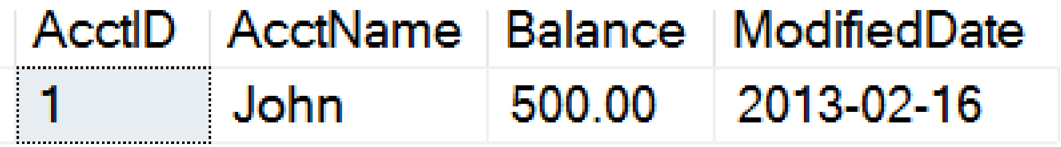
```
--  SQL Server Concurrency
-- Lost Update - Session 2
USE TSQL2012
GO
DECLARE @OldBalance int, @NewBalance int
BEGIN TRAN
    SELECT @OldBalance = Balance
    FROM Accounting.BankAccounts
    WHERE AcctID = 1
    SET @NewBalance = @OldBalance - 400

    UPDATE Accounting.BankAccounts
    SET Balance = @NewBalance
    WHERE AcctID = 1

    SELECT @OldBalance AS OldBalance,
     AcctID, AcctName, Balance
    FROM Accounting.BankAccounts
    WHERE AcctID = 1
COMMIT TRAN
```

OldBalance	AcctID	AcctName	Balance
500	1	John	100.00

Uncommitted dependency (dirty read)

```
--  SQL Server Concurrency
-- Dirty Read - Session 1
USE TSQL2012
GO
SET TRANSACTION ISOLATION LEVEL
READ UNCOMMITTED
BEGIN TRAN
    UPDATE Accounting.BankAccounts
    SET Balance -= 300
    WHERE AcctID = 1
        WAITFOR DELAY '00:00:10:000'
    ROLLBACK TRAN
    SELECT AcctID, AcctName, Balance
    FROM Accounting.BankAccounts
    WHERE AcctID = 1
```

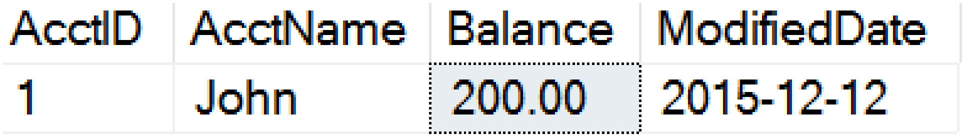
Clean Read



AcctID	AcctName	Balance	ModifiedDate
1	John	500.00	2013-02-16

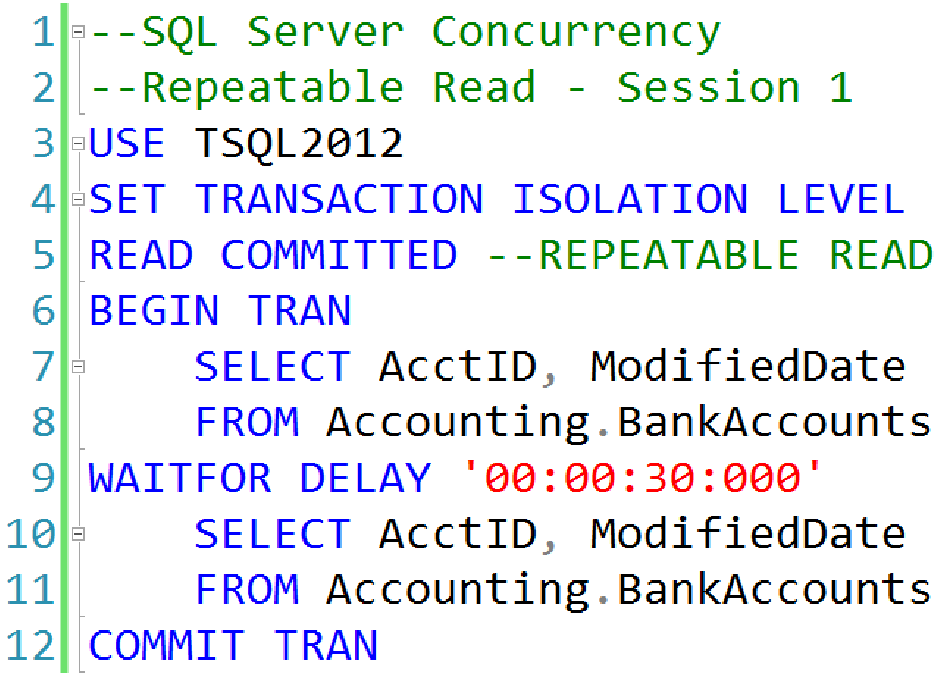
```
--SQL Server Concurrency
--Dirty Read - Session 2
USE TSQL2012
SET TRANSACTION ISOLATION LEVEL
READ UNCOMMITTED
    SELECT * FROM Accounting.BankAccounts
    WHERE AcctID = 1
```

Dirty Read

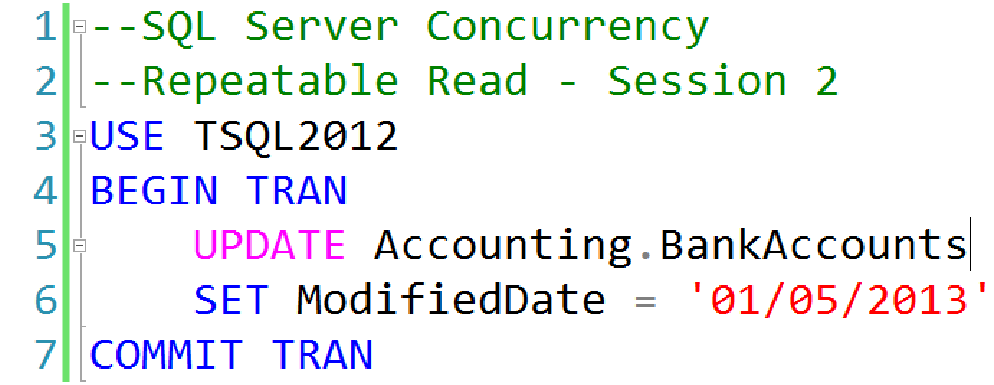


AcctID	AcctName	Balance	ModifiedDate
1	John	200.00	2015-12-12

Inconsistent analysis (non-repeatable read)

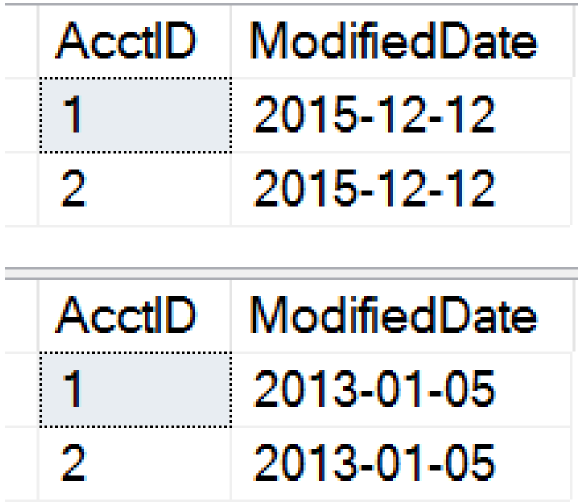


```
--SQL Server Concurrency
--Repeatable Read - Session 1
USE TSQL2012
SET TRANSACTION ISOLATION LEVEL
READ COMMITTED --REPEATABLE READ
BEGIN TRAN
    SELECT AcctID, ModifiedDate
    FROM Accounting.BankAccounts
WAITFOR DELAY '00:00:30:000'
    SELECT AcctID, ModifiedDate
    FROM Accounting.BankAccounts
COMMIT TRAN
```



```
--SQL Server Concurrency
--Repeatable Read - Session 2
USE TSQL2012
BEGIN TRAN
    UPDATE Accounting.BankAccounts
    SET ModifiedDate = '01/05/2013'
COMMIT TRAN
```

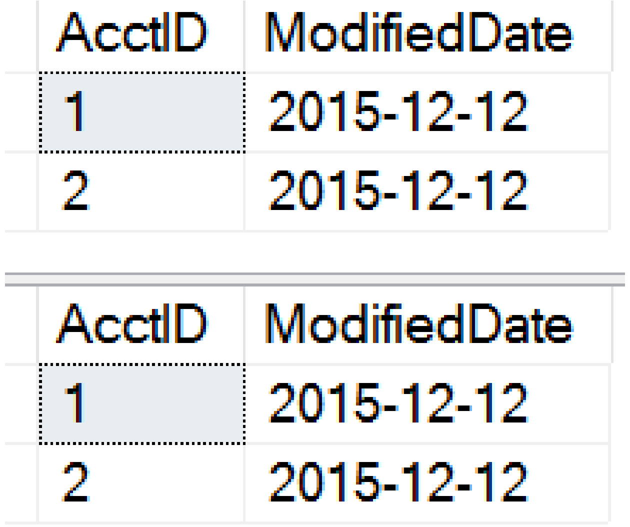
READ COMMITTED



AcctID	ModifiedDate
1	2015-12-12
2	2015-12-12

AcctID	ModifiedDate
1	2013-01-05
2	2013-01-05

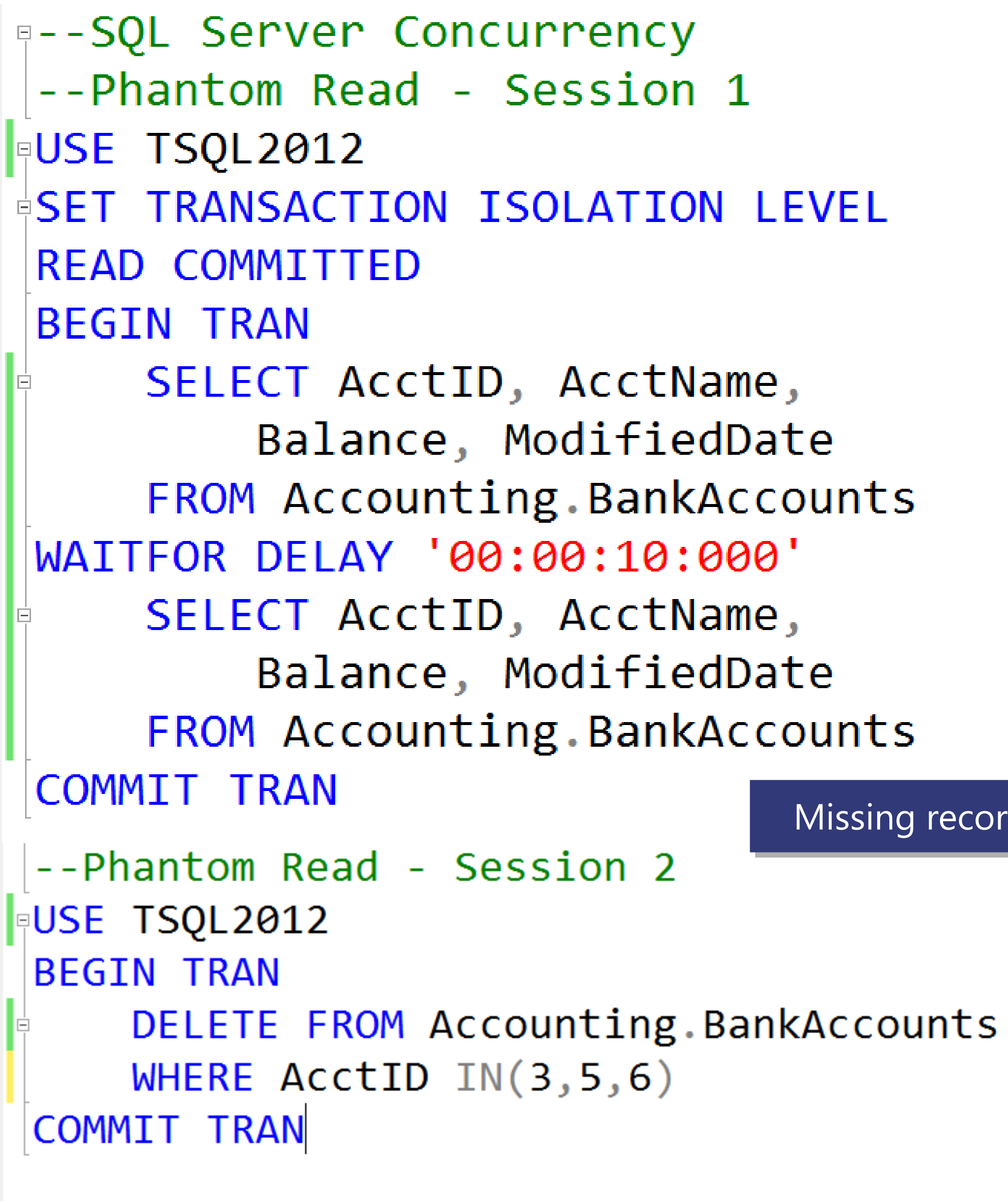
REPEATABLE READ



AcctID	ModifiedDate
1	2015-12-12
2	2015-12-12

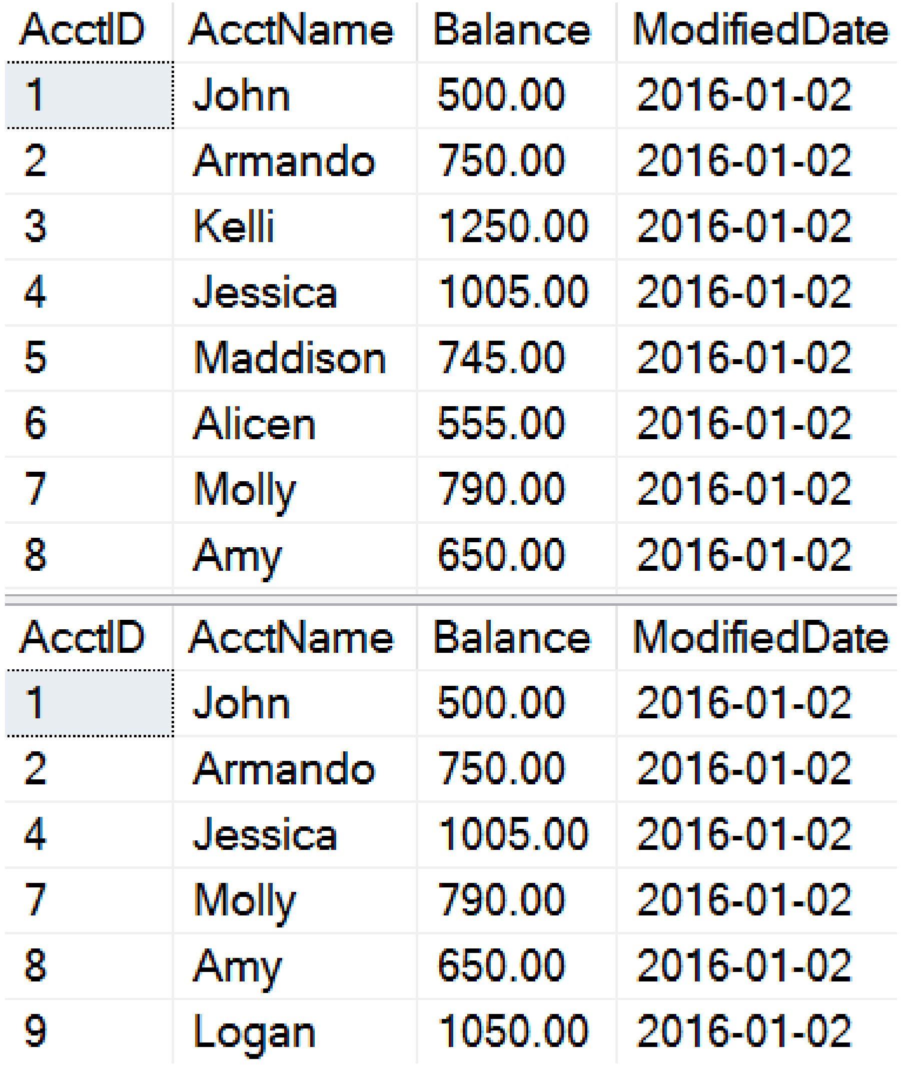
AcctID	ModifiedDate
1	2015-12-12
2	2015-12-12

Phantom Reads



```
--SQL Server Concurrency
--Phantom Read - Session 1
USE TSQL2012
SET TRANSACTION ISOLATION LEVEL
READ COMMITTED
BEGIN TRAN
    SELECT AcctID, AcctName,
        Balance, ModifiedDate
    FROM Accounting.BankAccounts
WAITFOR DELAY '00:00:10:000'
    SELECT AcctID, AcctName,
        Balance, ModifiedDate
    FROM Accounting.BankAccounts
COMMIT TRAN

--Phantom Read - Session 2
USE TSQL2012
BEGIN TRAN
    DELETE FROM Accounting.BankAccounts
    WHERE AcctID IN(3,5,6)
COMMIT TRAN
```

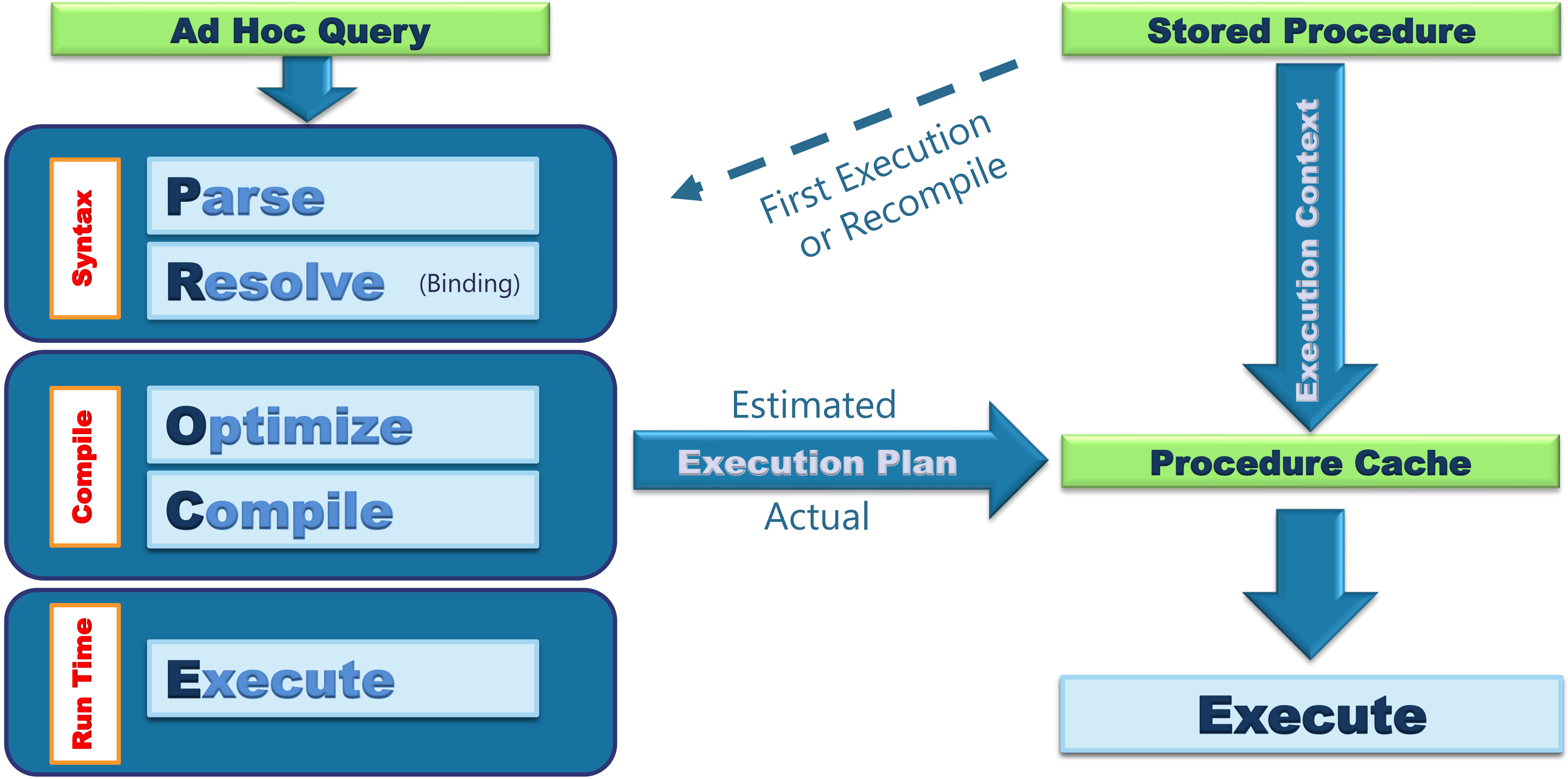
Missing records



AcctID	AcctName	Balance	ModifiedDate
1	John	500.00	2016-01-02
2	Armando	750.00	2016-01-02
3	Kelli	1250.00	2016-01-02
4	Jessica	1005.00	2016-01-02
5	Maddison	745.00	2016-01-02
6	Alicen	555.00	2016-01-02
7	Molly	790.00	2016-01-02
8	Amy	650.00	2016-01-02

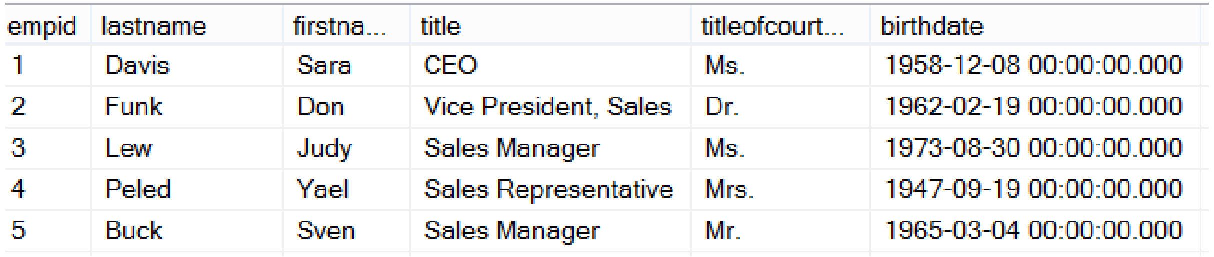
AcctID	AcctName	Balance	ModifiedDate
1	John	500.00	2016-01-02
2	Armando	750.00	2016-01-02
4	Jessica	1005.00	2016-01-02
7	Molly	790.00	2016-01-02
8	Amy	650.00	2016-01-02
9	Logan	1050.00	2016-01-02

How Queries are Processed



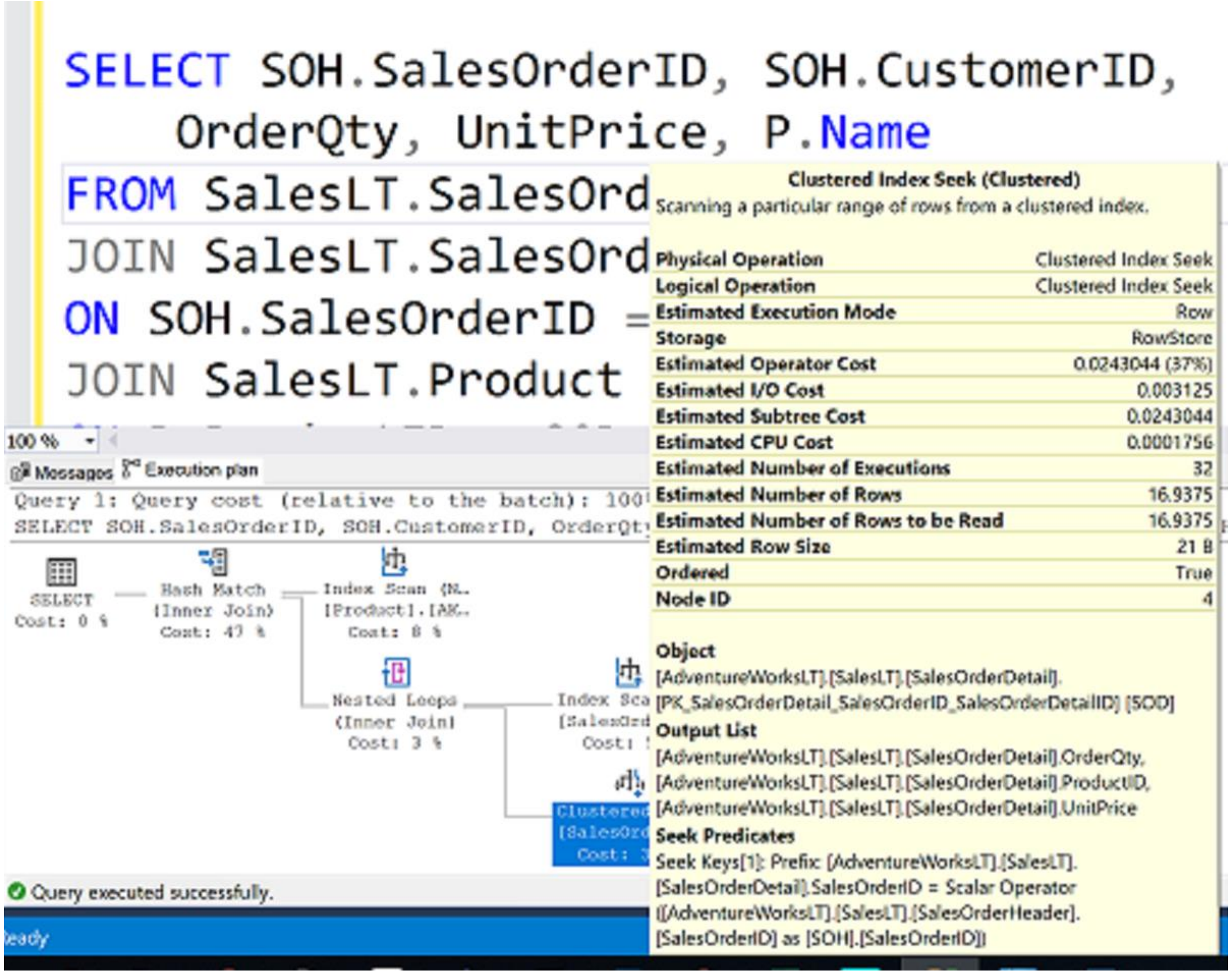
SQL

Sets



empid	lastname	firstna...	title	titleofcourt...	birthdate
1	Davis	Sara	CEO	Ms.	1958-12-08 00:00:00.000
2	Funk	Don	Vice President, Sales	Dr.	1962-02-19 00:00:00.000
3	Lew	Judy	Sales Manager	Ms.	1973-08-30 00:00:00.000
4	Peled	Yael	Sales Representative	Mrs.	1947-09-19 00:00:00.000
5	Buck	Sven	Sales Manager	Mr.	1965-03-04 00:00:00.000

What is an Execution Plan?



```
SELECT SOH.SalesOrderID, SOH.CustomerID,
    OrderQty, UnitPrice, P.Name
FROM SalesLT.SalesOrd
JOIN SalesLT.SalesOrd
ON SOH.SalesOrderID =
JOIN SalesLT.Product
```

```
Query 1: Query cost (relative to the batch): 100
SELECT SOH.SalesOrderID, SOH.CustomerID, OrderQt
```

Query executed successfully.

Clustered Index Seek (Clustered)

Scanning a particular range of rows from a clustered index.

Physical Operation	Clustered Index Seek
Logical Operation	Clustered Index Seek
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.0243044 (37%)
Estimated I/O Cost	0.003125
Estimated Subtree Cost	0.0243044
Estimated CPU Cost	0.0001756
Estimated Number of Executions	32
Estimated Number of Rows	16.9375
Estimated Number of Rows to be Read	16.9375
Estimated Row Size	21 B
Ordered	True
Node ID	4

Object

[AdventureWorksLT].[SalesLT].[SalesOrderDetail].
[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] [SOD]

Output List

[AdventureWorksLT].[SalesLT].[SalesOrderDetail].OrderQty,
[AdventureWorksLT].[SalesLT].[SalesOrderDetail].ProductID,
[AdventureWorksLT].[SalesLT].[SalesOrderDetail].UnitPrice

Seek Predicates

Seek Keys[1]: Prefix: [AdventureWorksLT].[SalesLT].
[SalesOrderDetail].SalesOrderID = Scalar Operator
([AdventureWorksLT].[SalesLT].[SalesOrderHeader].
[SalesOrderID] as [SOH].[SalesOrderID])

Execution Plan Table Operators

Data stored in a Heap is not stored in any order and normally does not have a Primary Key.

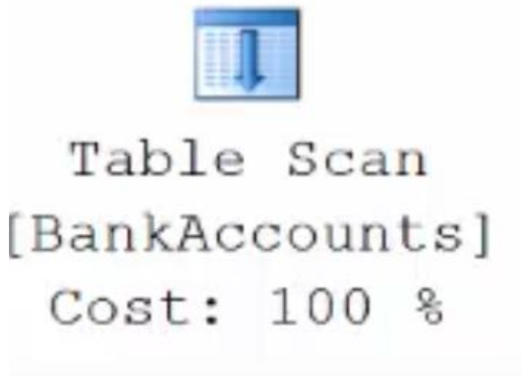
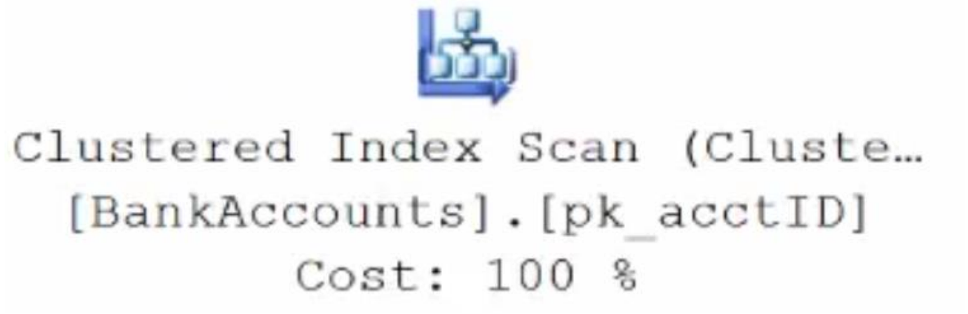


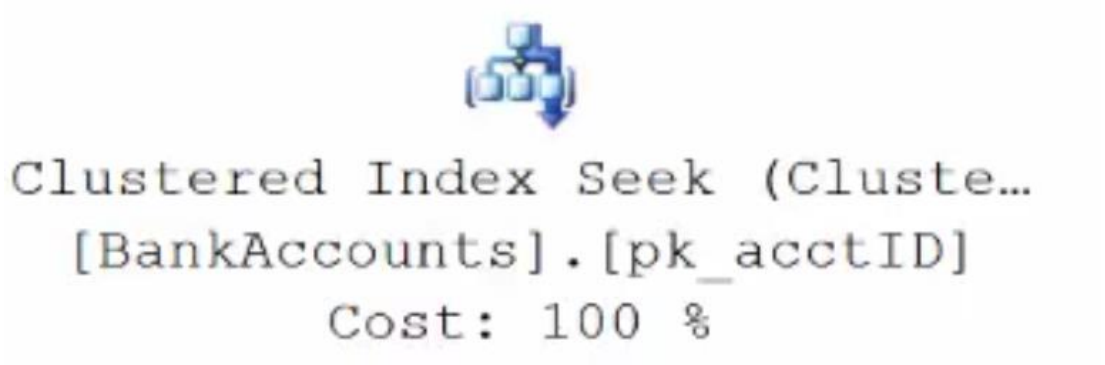
Table Scan
[BankAccounts]
Cost: 100 %

Clustered Index data is stored in sorted order by the Clustering key. In many cases, this is the same value as the Primary Key.



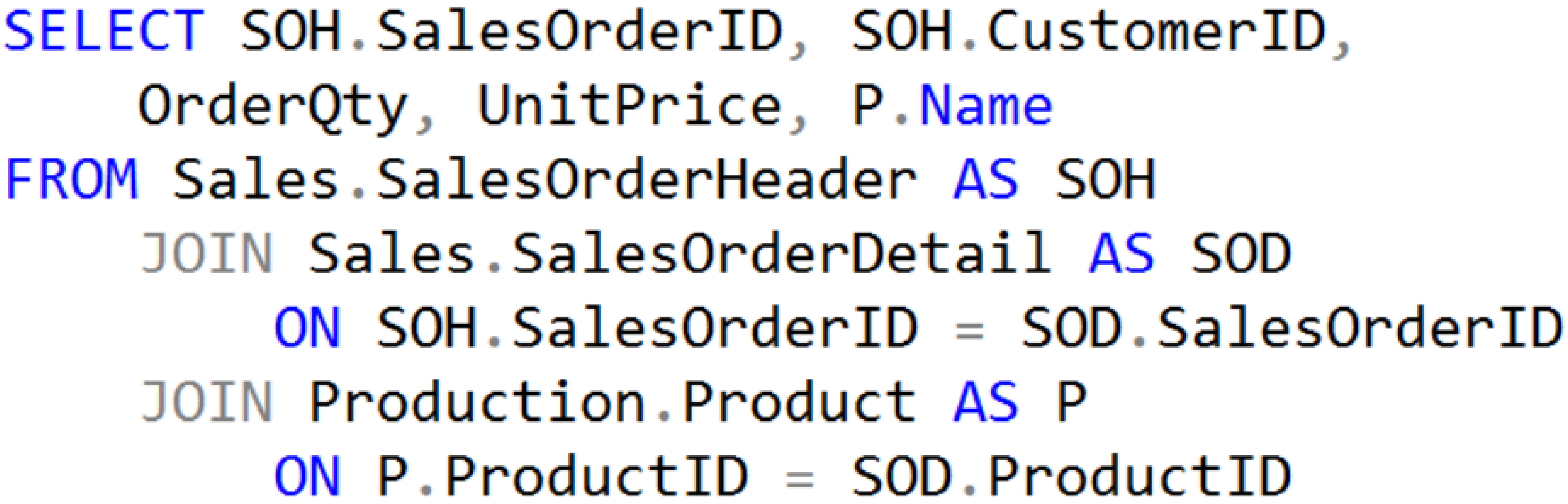
Clustered Index Scan (Cluste…
[BankAccounts].[pk_acctID]
Cost: 100 %

Using a WHERE statement on an Index could possibly have the Execution Plan seek the Index instead of scan.



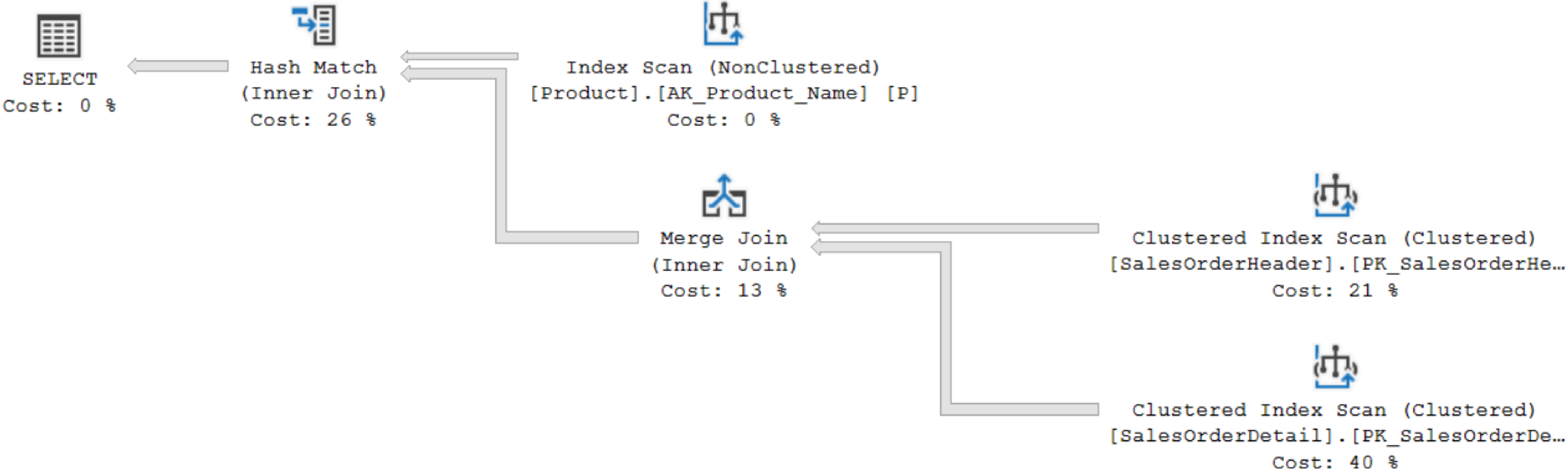
Clustered Index Seek (Cluste…
[BankAccounts].[pk_acctID]
Cost: 100 %

Execution Plan Join Operators (Code)



```
SELECT SOH.SalesOrderID, SOH.CustomerID,
    OrderQty, UnitPrice, P.Name
FROM Sales.SalesOrderHeader AS SOH
    JOIN Sales.SalesOrderDetail AS SOD
        ON SOH.SalesOrderID = SOD.SalesOrderID
    JOIN Production.Product AS P
        ON P.ProductID = SOD.ProductID
```

Execution Plan Join Operators (Plan)



SELECT
Cost: 0 %

Hash Match
(Inner Join)
Cost: 26 %

Index Scan (NonClustered)
[Product].[AK_Product_Name] [P]
Cost: 0 %

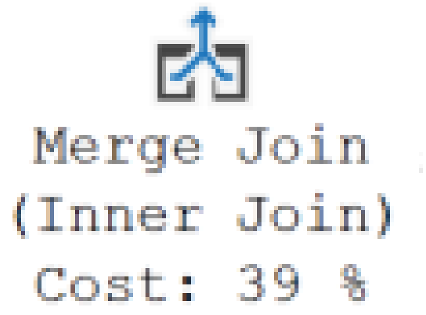
Merge Join
(Inner Join)
Cost: 13 %

Clustered Index Scan (Clustered)
[SalesOrderHeader].[PK_SalesOrderHe…
Cost: 21 %

Clustered Index Scan (Clustered)
[SalesOrderDetail].[PK_SalesOrderDe…
Cost: 40 %

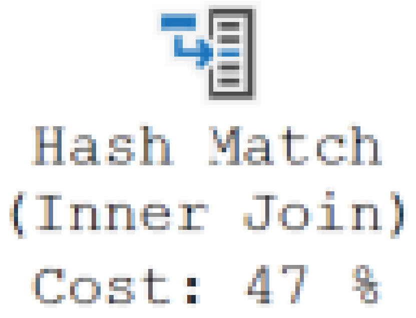
Execution Plan Join Operators

A Merge Join is useful if both table inputs are in the same sorted order on the same value.



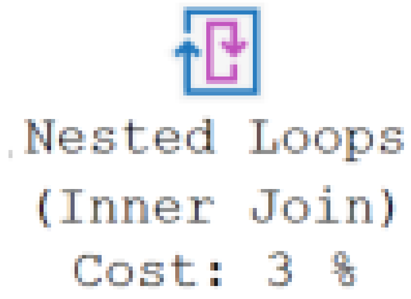
Merge Join
(Inner Join)
Cost: 39 %

A Hash Match is used when the tables being joined are not in the same sorted order.



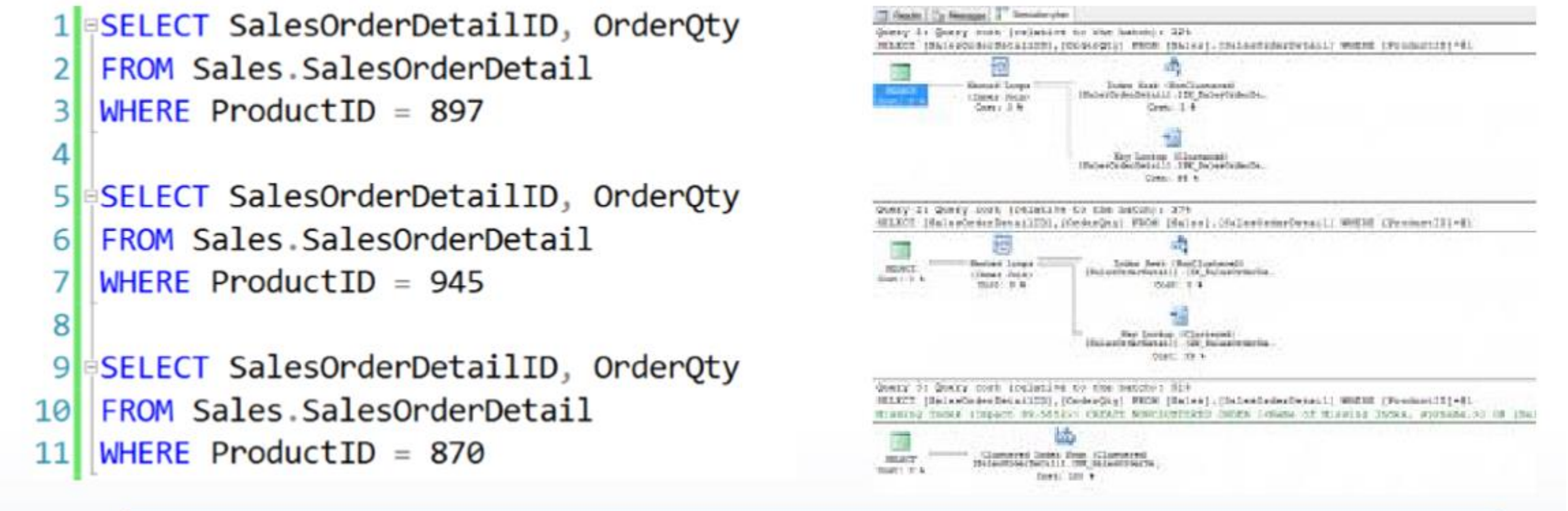
Hash Match
(Inner Join)
Cost: 47 %

A Nested Loop is use when a small (outer) table is used to lookup a value in a larger (inner) table.



Nested Loops
(Inner Join)
Cost: 3 %

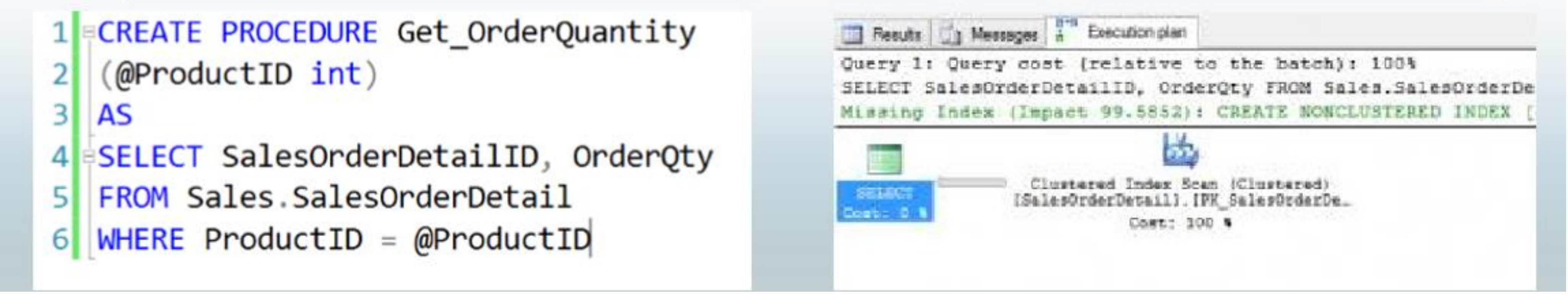
Parameter Sniffing



```
SELECT SalesOrderDetailID, OrderQty
FROM Sales.SalesOrderDetail
WHERE ProductID = 897

SELECT SalesOrderDetailID, OrderQty
FROM Sales.SalesOrderDetail
WHERE ProductID = 945

SELECT SalesOrderDetailID, OrderQty
FROM Sales.SalesOrderDetail
WHERE ProductID = 870
```

```
Query 1: Query cost (relative to the batch): [illegible]
SELECT [SalesOrderDetailID],[OrderQty] FROM [Sales].[SalesOrderDetail] WHERE [ProductID]=@1

Query 2: Query cost (relative to the batch): [illegible]
SELECT [SalesOrderDetailID],[OrderQty] FROM [Sales].[SalesOrderDetail] WHERE [ProductID]=@1

Query 3: Query cost (relative to the batch): [illegible]
SELECT [SalesOrderDetailID],[OrderQty] FROM [Sales].[SalesOrderDetail] WHERE [ProductID]=@1
Missing Index (Impact [illegible]): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [Sal
```

Results
Messages
Execution plan
SELECT
Nested Loops
(Inner Join)
Index Seek (NonClustered)
[SalesOrderDetail].[IX_SalesOrderDe...
Key Lookup (Clustered)
[SalesOrderDetail].[PK_SalesOrderDe...
Clustered Index Scan (Clustered)
[SalesOrderDetail].[PK_SalesOrderDe...
Cost: 100 %



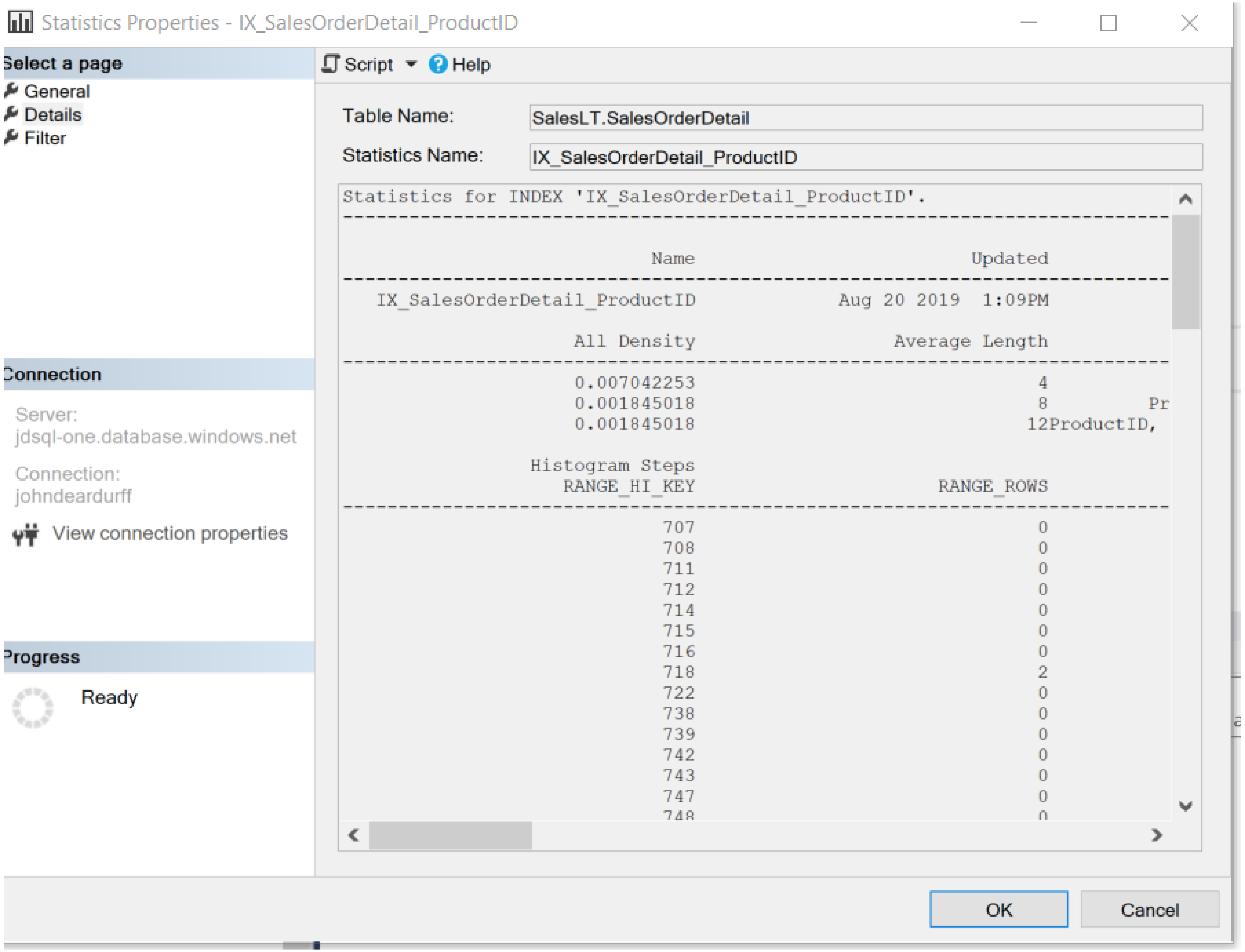
```
CREATE PROCEDURE Get_OrderQuantity
(@ProductID int)
AS
SELECT SalesOrderDetailID, OrderQty
FROM Sales.SalesOrderDetail
WHERE ProductID = @ProductID
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT SalesOrderDetailID, OrderQty FROM Sales.SalesOrderDe
Missing Index (Impact 99.5852): CREATE NONCLUSTERED INDEX [
```

Results
Messages
Execution plan
SELECT
Cost: 0 %
Clustered Index Scan (Clustered)
[SalesOrderDetail].[PK_SalesOrderDe...
Cost: 100 %

https://www.brentozar.com/archive/2013/06/the-elephant-and-the-mouse-or-parameter-sniffing-in-sql-server/

Cardinality Estimator and Statistics



Statistics Properties - IX_SalesOrderDetail_ProductID
Select a page
General
Details
Filter
Script
Help
Connection
View connection properties
Progress
OK
Cancel

Table Name: SalesLT.SalesOrderDetail

Statistics Name: IX_SalesOrderDetail_ProductID

```
Statistics for INDEX 'IX_SalesOrderDetail_ProductID'.
----------------------------------------------------------------------------


                        Name                          Updated
----------------------------------------------------------------------------
   IX_SalesOrderDetail_ProductID             Aug 20 2019  1:09PM

                 All Density                Average Length
----------------------------------------------------------------------------
                 0.007042253                              4
                 0.001845018                              8           Pr
                 0.001845018                             12ProductID,

             Histogram Steps
                RANGE_HI_KEY                     RANGE_ROWS
----------------------------------------------------------------------------
                         707                              0
                         708                              0
                         711                              0
                         712                              0
                         714                              0
                         715                              0
                         716                              0
                         718                              2
                         722                              0
                         738                              0
                         739                              0
                         742                              0
                         743                              0
                         747                              0
                         748                              0
```

Server:
jdsql-one.database.windows.net

Connection:
johndeardurff

Ready

Microsoft