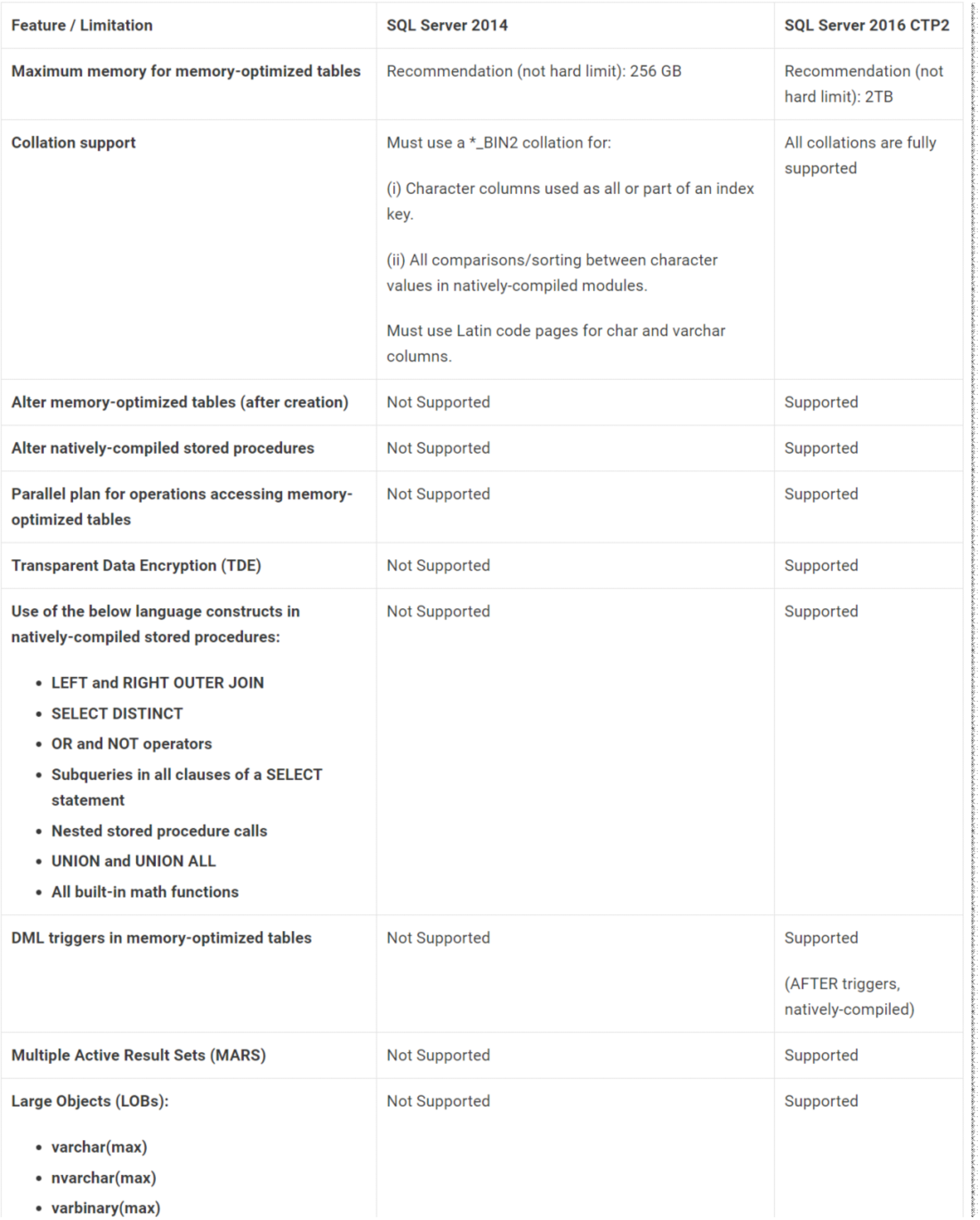
What it is…

Sunday, May 28, 2017 4:57 PM

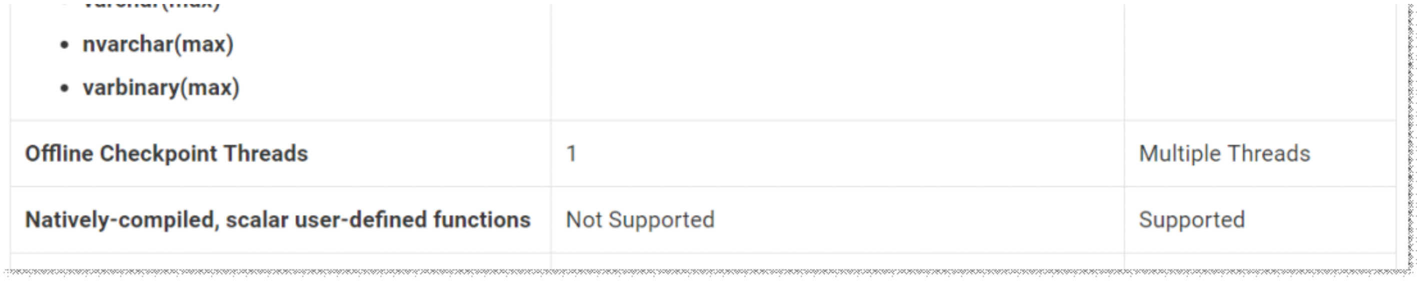
Memory-optimized tables do not have data pages or extents, they are just "data rows" written to memory sequentially in the order the transactions occurred

Version Differences

Sunday, May 28, 2017 4:32 PM



Feature / Limitation	SQL Server 2014	SQL Server 2016 CTP2
Maximum memory for memory-optimized tables	Recommendation (not hard limit): 256 GB	Recommendation (not hard limit): 2TB
Collation support	Must use a *_BIN2 collation for: (i) Character columns used as all or part of an index key. (ii) All comparisons/sorting between character values in natively-compiled modules. Must use Latin code pages for char and varchar columns.	All collations are fully supported
Alter memory-optimized tables (after creation)	Not Supported	Supported
Alter natively-compiled stored procedures	Not Supported	Supported
Parallel plan for operations accessing memory-optimized tables	Not Supported	Supported
Transparent Data Encryption (TDE)	Not Supported	Supported
Use of the below language constructs in natively-compiled stored procedures: • **LEFT and RIGHT OUTER JOIN** • **SELECT DISTINCT** • **OR and NOT operators** • **Subqueries in all clauses of a SELECT statement** • **Nested stored procedure calls** • **UNION and UNION ALL** • **All built-in math functions**	Not Supported	Supported
DML triggers in memory-optimized tables	Not Supported	Supported (AFTER triggers, natively-compiled)
Multiple Active Result Sets (MARS)	Not Supported	Supported
Large Objects (LOBs): • **varchar(max)** • **nvarchar(max)** • **varbinary(max)**	Not Supported	Supported



• **varchar(max)** • **nvarchar(max)** • **varbinary(max)**		
Offline Checkpoint Threads	1	Multiple Threads
Natively-compiled, scalar user-defined functions	Not Supported	Supported

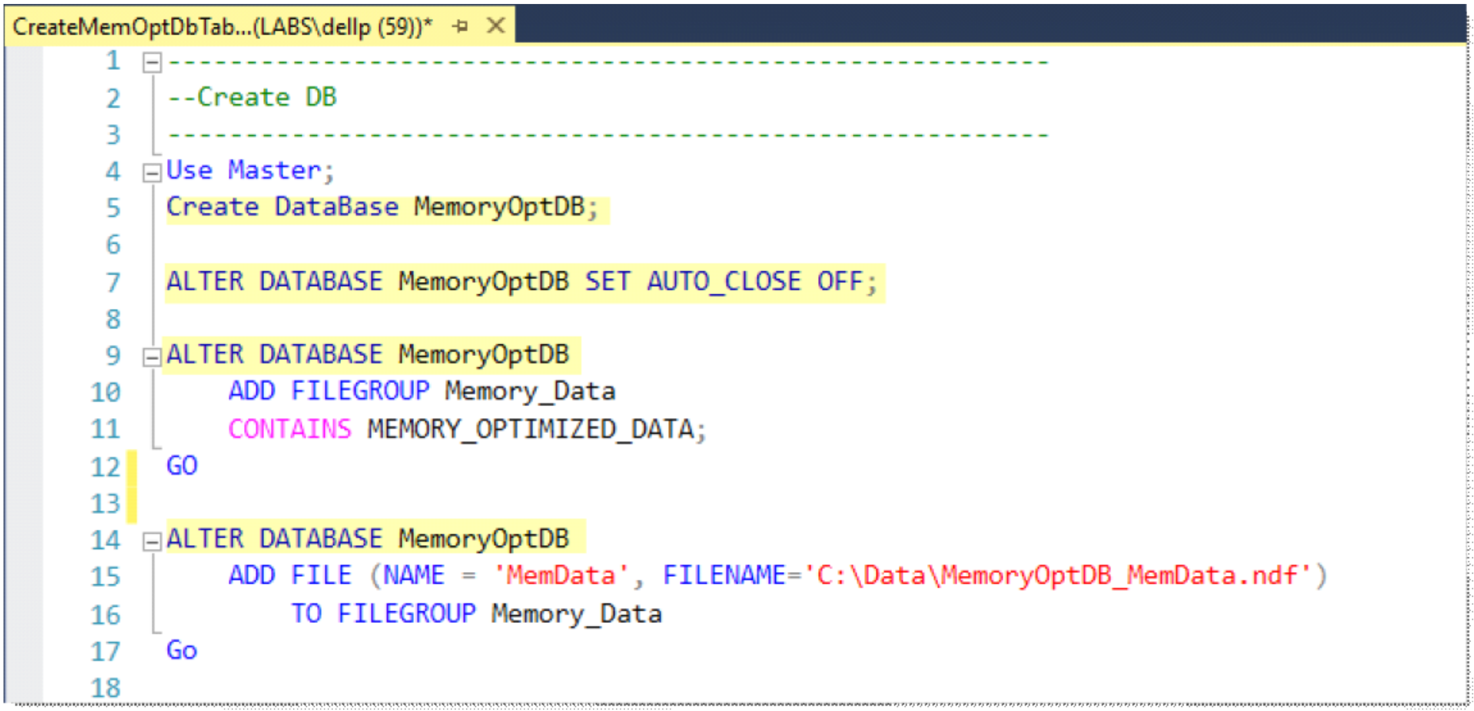
Possible Demo

Sunday, May 28, 2017 5:54 PM

In-Memory OLTP in SQL Server 2016

Mem. Opt. Database

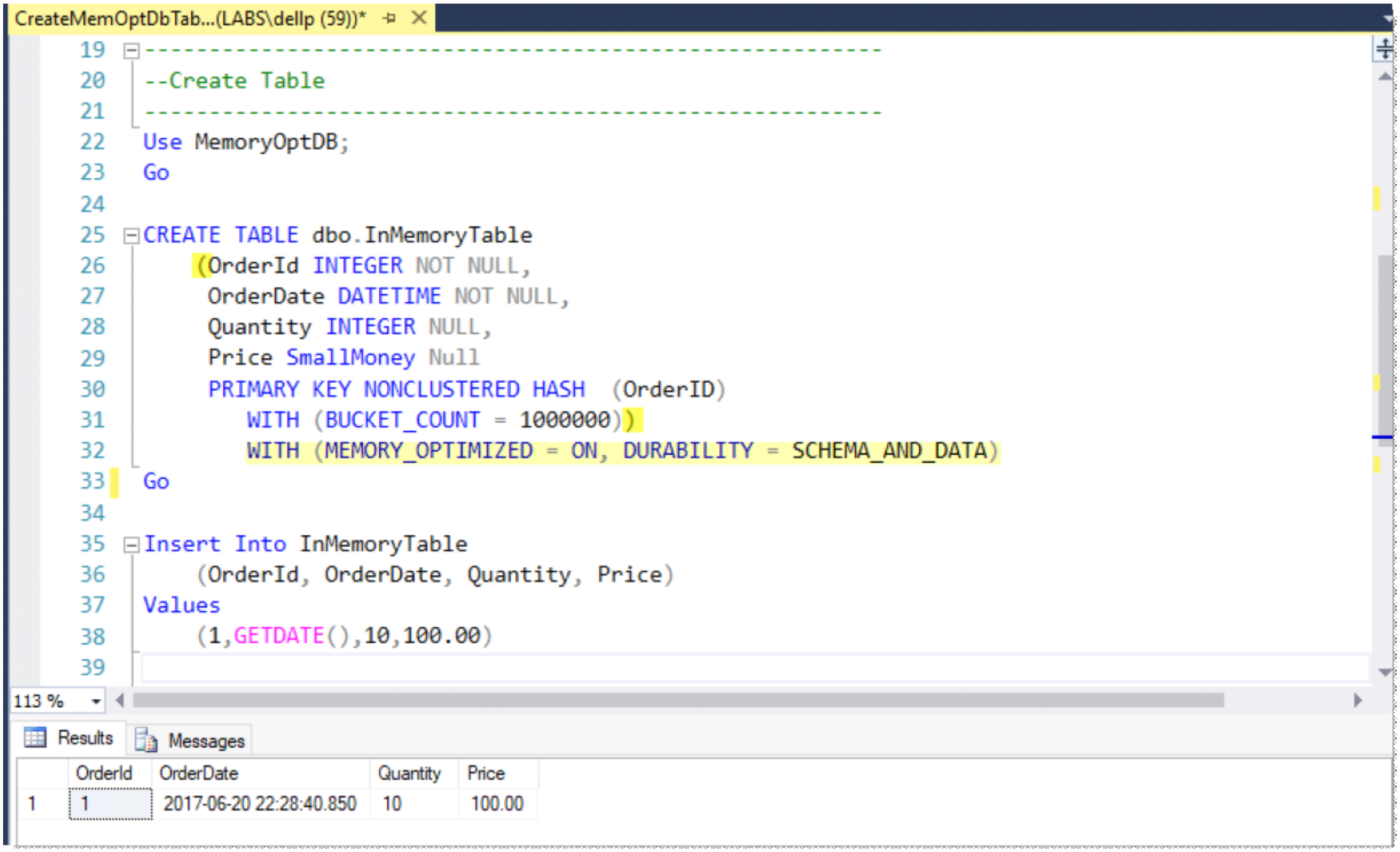
Sunday, May 28, 2017 6:04 PM



```
---------------------------------------------------------
--Create DB
---------------------------------------------------------
Use Master;
Create DataBase MemoryOptDB;

ALTER DATABASE MemoryOptDB SET AUTO_CLOSE OFF;

ALTER DATABASE MemoryOptDB
    ADD FILEGROUP Memory_Data
    CONTAINS MEMORY_OPTIMIZED_DATA;
GO

ALTER DATABASE MemoryOptDB
    ADD FILE (NAME = 'MemData', FILENAME='C:\Data\MemoryOptDB_MemData.ndf')
        TO FILEGROUP Memory_Data
Go

```

Memory Opt. Table

Tuesday, June 20, 2017 10:37 PM



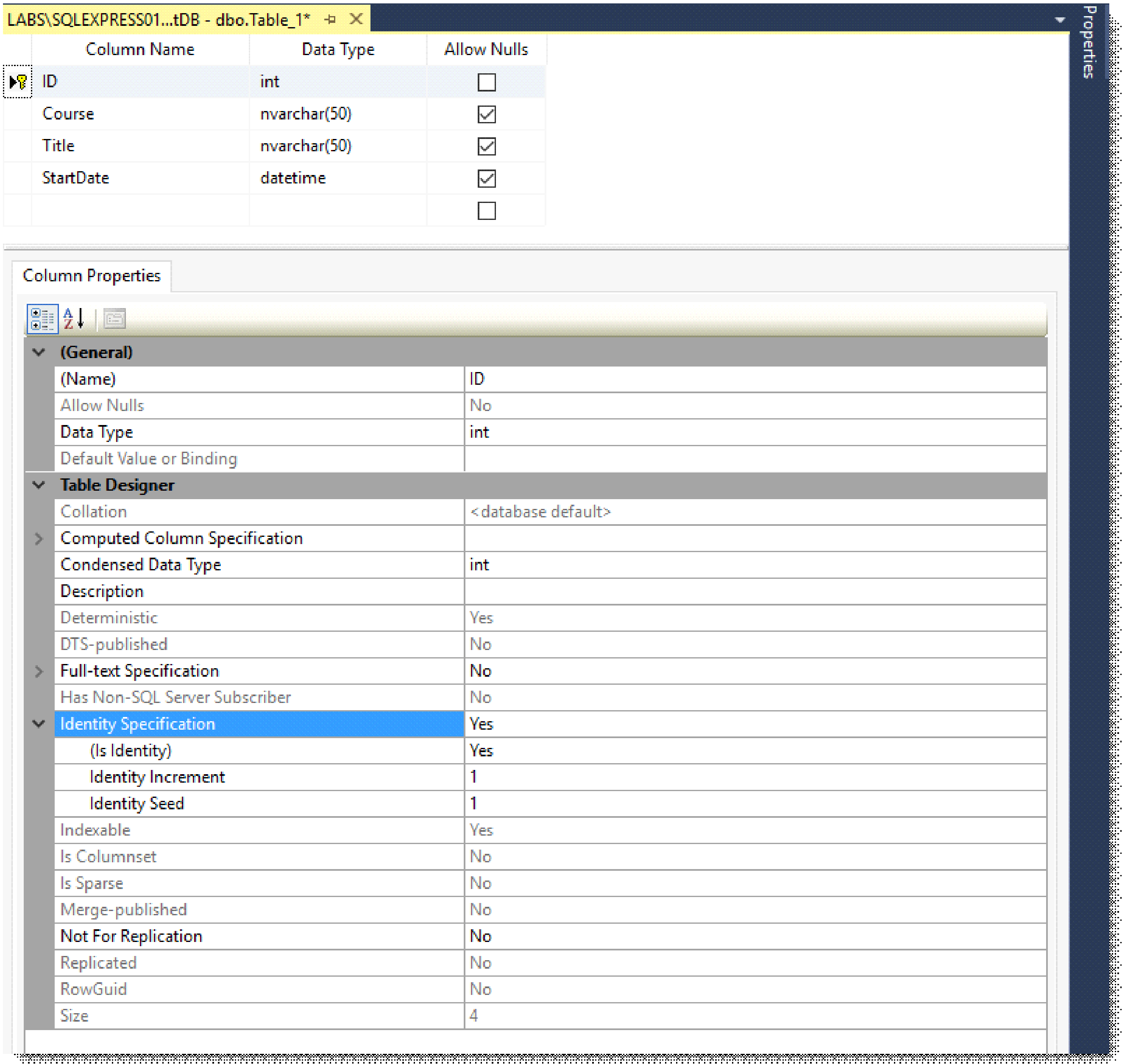
```sql
--------------------------------------------------------
--Create Table
--------------------------------------------------------
Use MemoryOptDB;
Go

CREATE TABLE dbo.InMemoryTable
    (OrderId INTEGER NOT NULL,
     OrderDate DATETIME NOT NULL,
     Quantity INTEGER NULL,
     Price SmallMoney Null
     PRIMARY KEY NONCLUSTERED HASH  (OrderID)
        WITH (BUCKET_COUNT = 1000000))
        WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA)
Go

Insert Into InMemoryTable
    (OrderId, OrderDate, Quantity, Price)
Values
    (1,GETDATE(),10,100.00)
```

	OrderId	OrderDate	Quantity	Price
1	1	2017-06-20 22:28:40.850	10	100.00

Create Mem. Opt. Table

Sunday, May 28, 2017 6:12 PM

□ Create Normal Table in Optimised DB

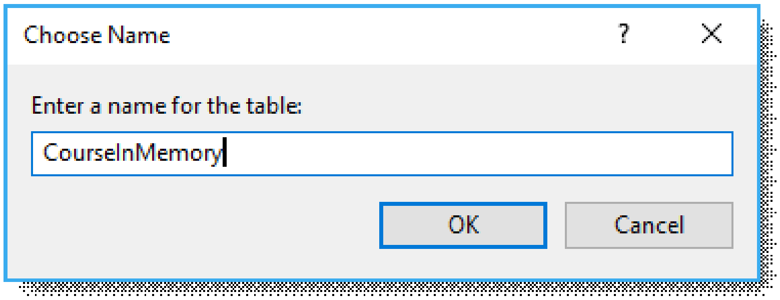


LABS\SQLEXPRESS01...tDB - dbo.Table_1*

Column Name	Data Type	Allow Nulls
ID	int	☐
Course	nvarchar(50)	☑
Title	nvarchar(50)	☑
StartDate	datetime	☑
		☐

Column Properties

(General)	
(Name)	ID
Allow Nulls	No
Data Type	int
Default Value or Binding	
Table Designer	
Collation	<database default>
Computed Column Specification	
Condensed Data Type	int
Description	
Deterministic	Yes
DTS-published	No
Full-text Specification	No
Has Non-SQL Server Subscriber	No
Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1
Indexable	Yes
Is Columnset	No
Is Sparse	No
Merge-published	No
Not For Replication	No
Replicated	No
RowGuid	No
Size	4



Choose Name

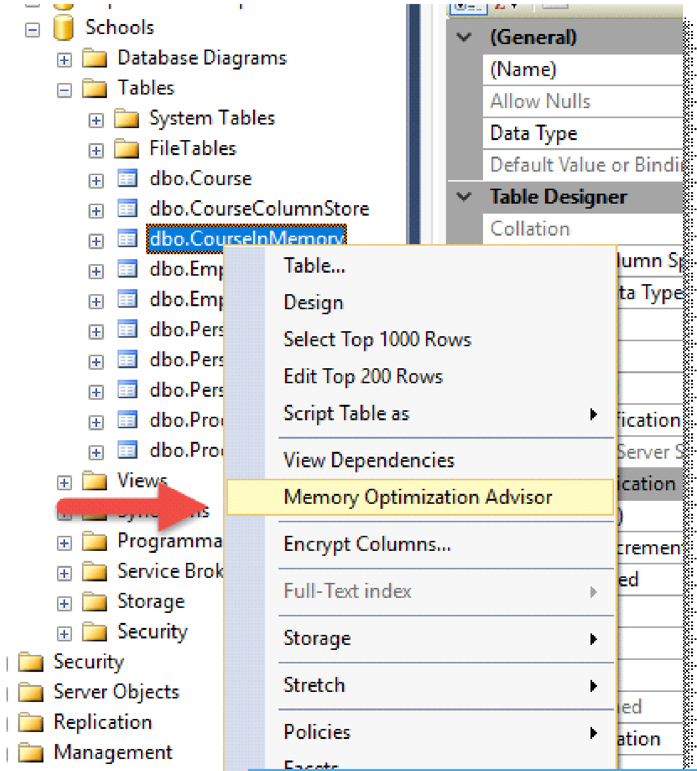
Enter a name for the table:

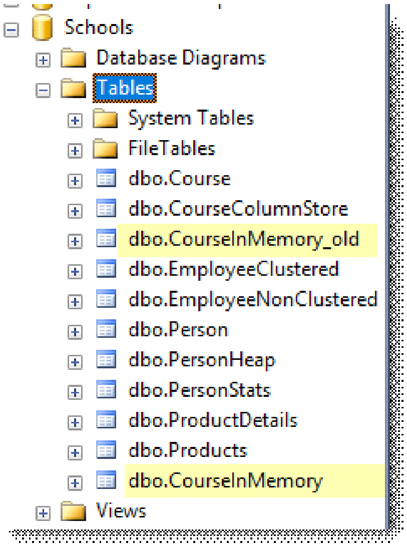
CourseInMemory

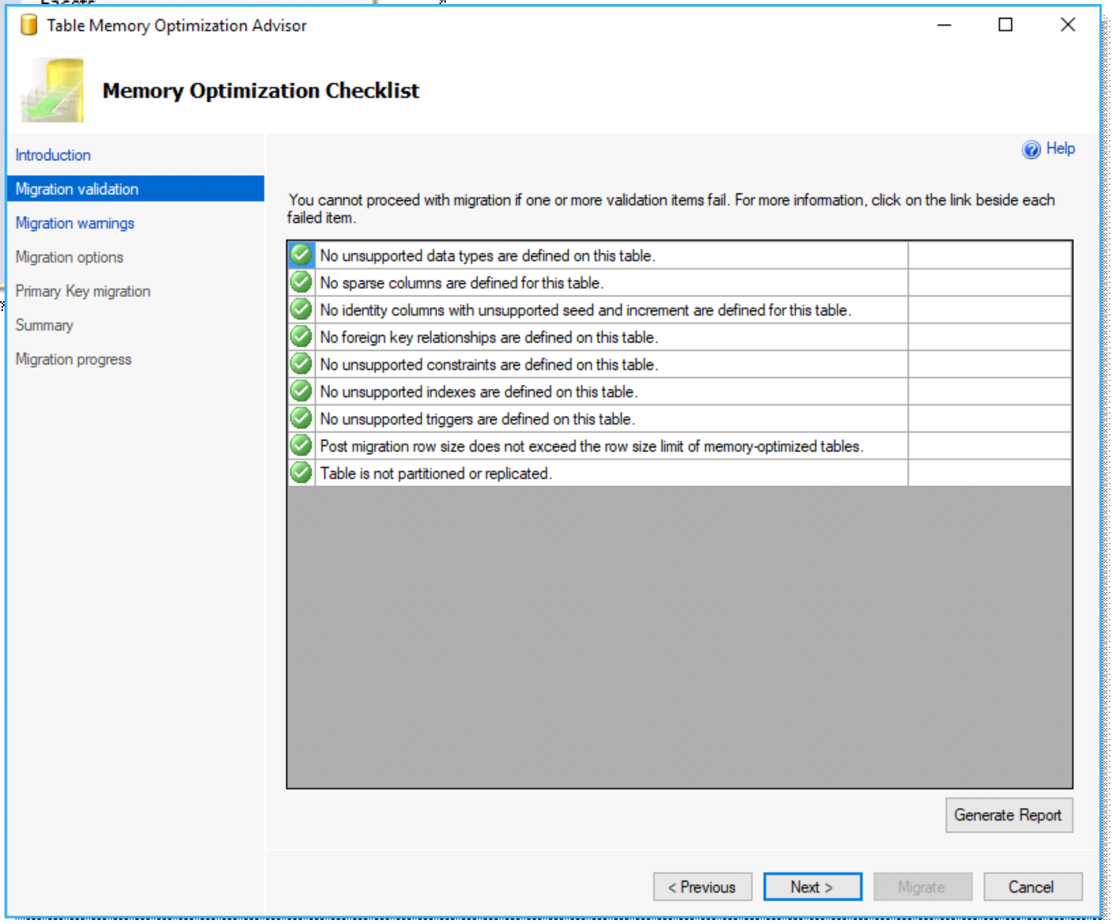
Convert to In-Memory

Sunday, May 28, 2017 6:23 PM

□ Convert to In-Memory



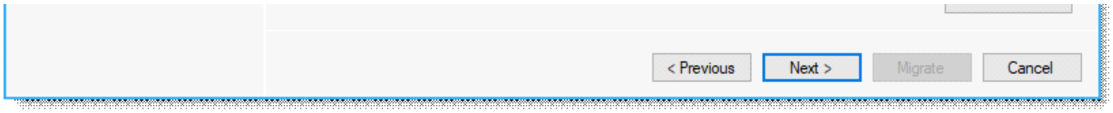


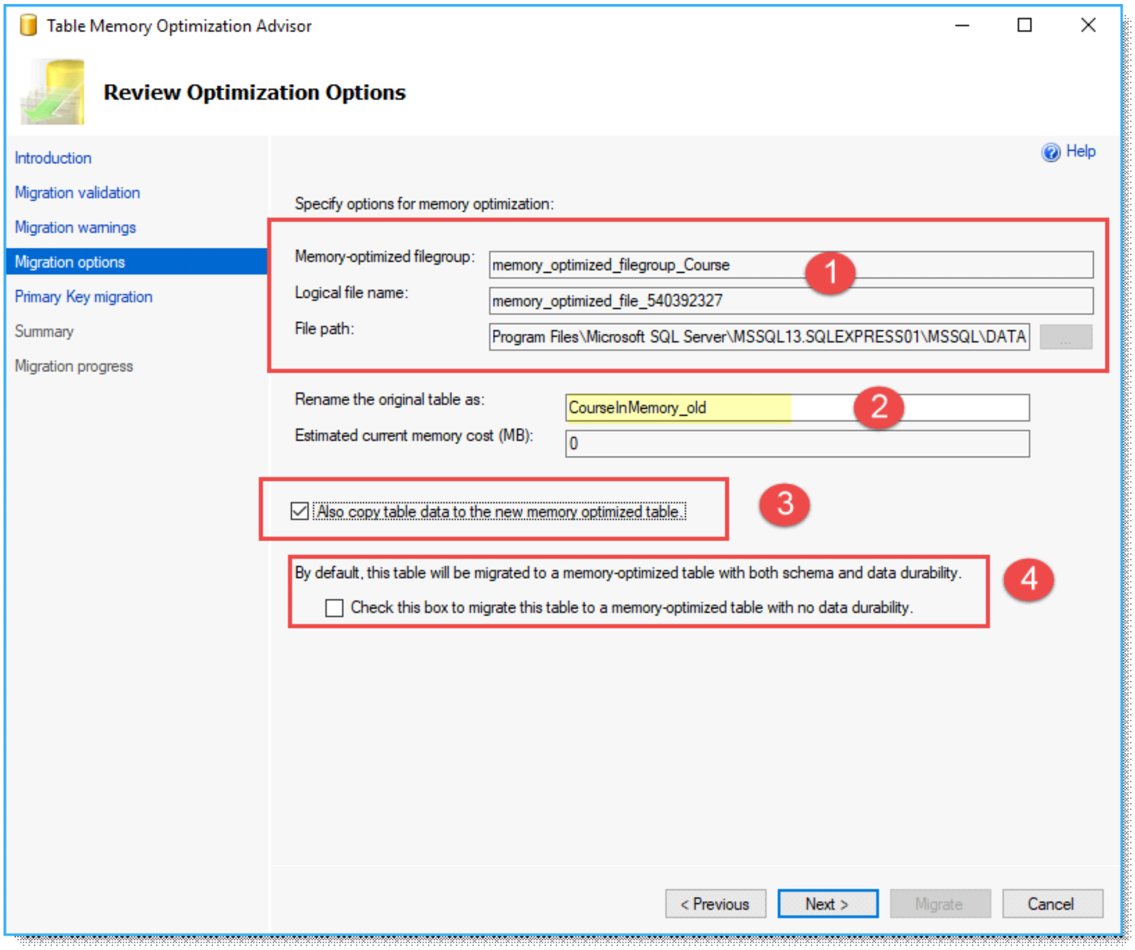


Memory Optimization Checklist

You cannot proceed with migration if one or more validation items fail. For more information, click on the link beside each failed item.

✓ No unsupported data types are defined on this table.	
✓ No sparse columns are defined for this table.	
✓ No identity columns with unsupported seed and increment are defined for this table.	
✓ No foreign key relationships are defined on this table.	
✓ No unsupported constraints are defined on this table.	
✓ No unsupported indexes are defined on this table.	
✓ No unsupported triggers are defined on this table.	
✓ Post migration row size does not exceed the row size limit of memory-optimized tables.	
✓ Table is not partitioned or replicated.	





Review Optimization Options

Specify options for memory optimization:

Memory-optimized filegroup: memory_optimized_filegroup_Course

Logical file name: memory_optimized_file_540392327

File path: Program Files\Microsoft SQL Server\MSSQL13.SQLEXPRESS01\MSSQL\DATA

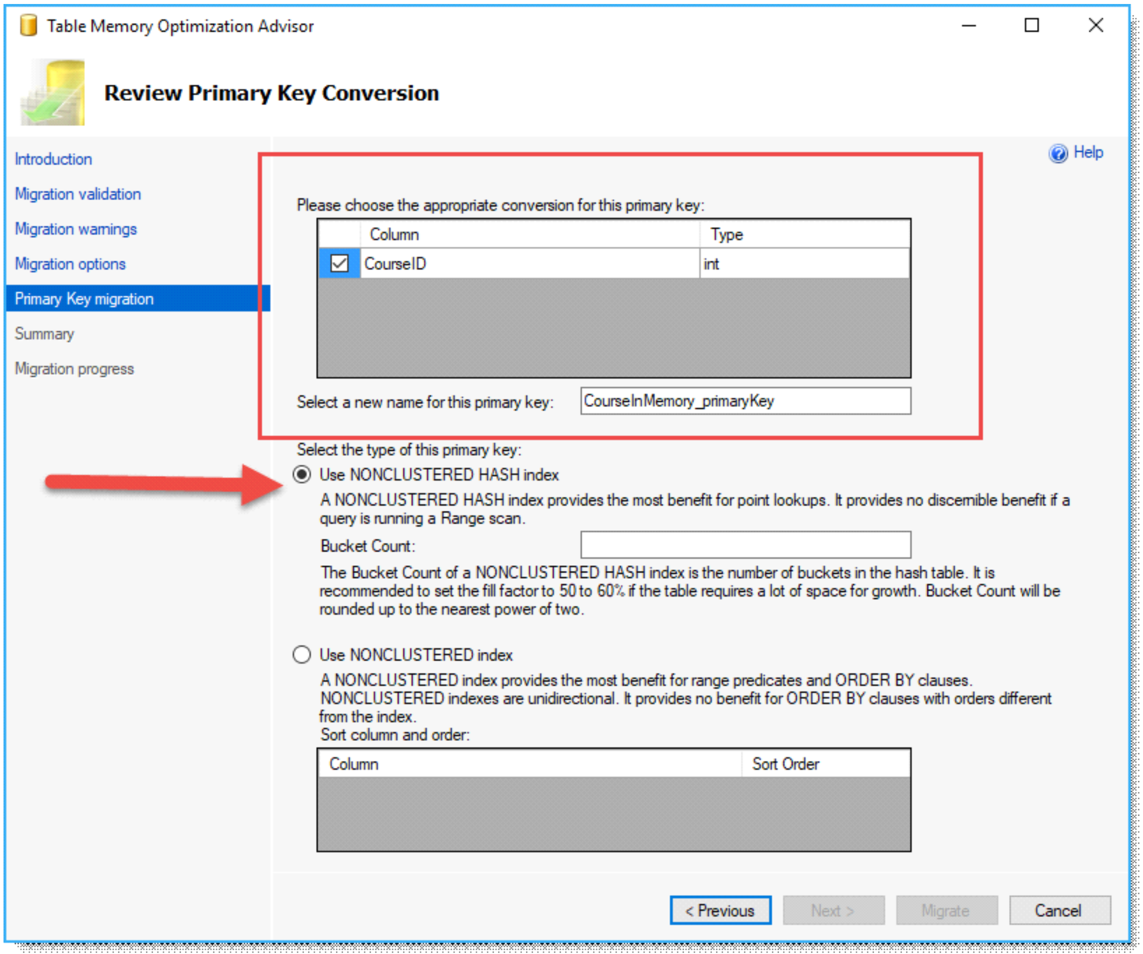
Rename the original table as: CourseInMemory_old

Estimated current memory cost (MB): 0

☑ Also copy table data to the new memory optimized table.

By default, this table will be migrated to a memory-optimized table with both schema and data durability.

□ Check this box to migrate this table to a memory-optimized table with no data durability.



Review Primary Key Conversion

Please choose the appropriate conversion for this primary key:

	Column	Type
☑	CourseID	int

Select a new name for this primary key: CourseInMemory_primaryKey

Select the type of this primary key:

● Use NONCLUSTERED HASH index

A NONCLUSTERED HASH index provides the most benefit for point lookups. It provides no discernible benefit if a query is running a Range scan.

Bucket Count:

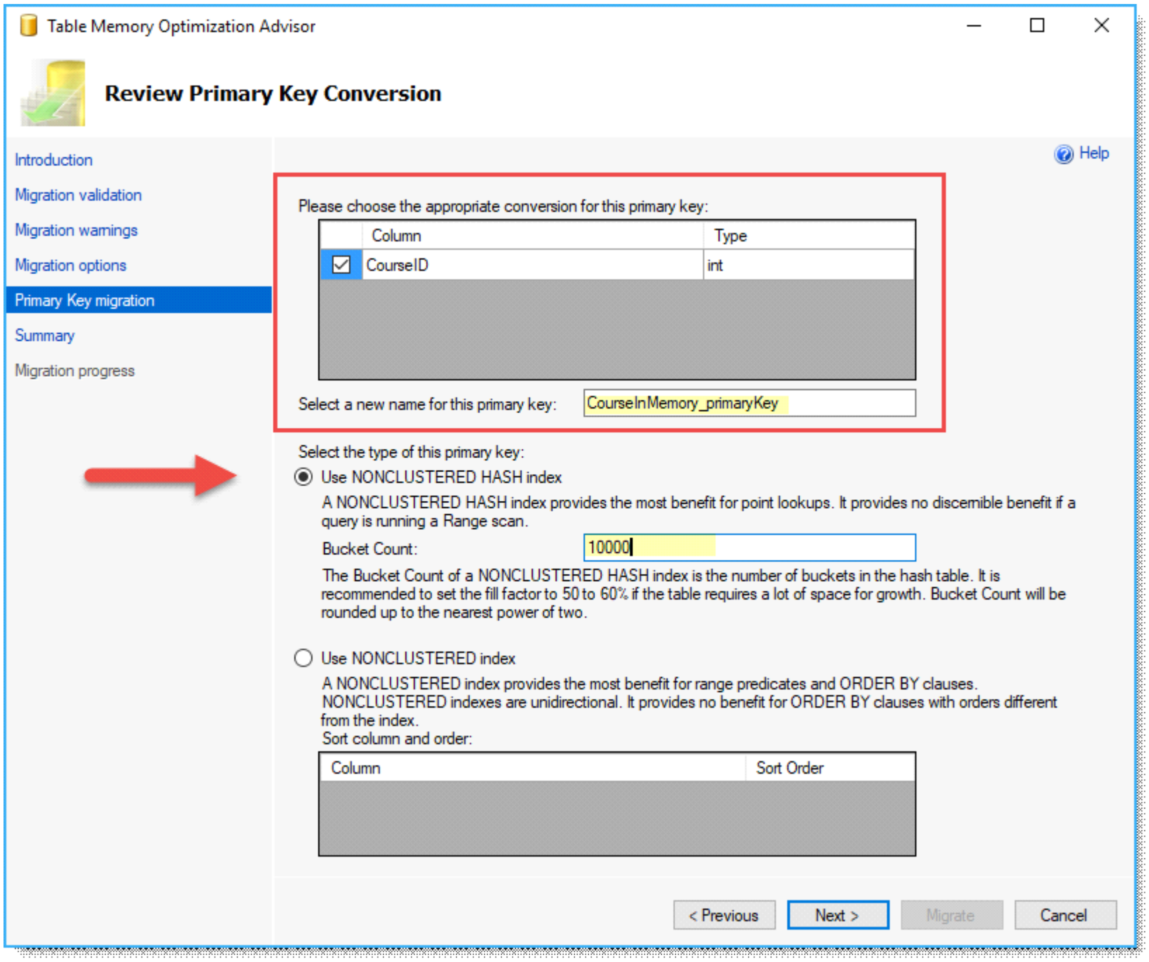
The Bucket Count of a NONCLUSTERED HASH index is the number of buckets in the hash table. It is recommended to set the fill factor to 50 to 60% if the table requires a lot of space for growth. Bucket Count will be rounded up to the nearest power of two.

○ Use NONCLUSTERED index

A NONCLUSTERED index provides the most benefit for range predicates and ORDER BY clauses. NONCLUSTERED indexes are unidirectional. It provides no benefit for ORDER BY clauses with orders different from the index.

Sort column and order:

Column	Sort Order



Review Primary Key Conversion

Please choose the appropriate conversion for this primary key:

	Column	Type
☑	CourseID	int

Select a new name for this primary key: CourseInMemory_primaryKey

Select the type of this primary key:

● Use NONCLUSTERED HASH index

A NONCLUSTERED HASH index provides the most benefit for point lookups. It provides no discernible benefit if a query is running a Range scan.

Bucket Count: 10000

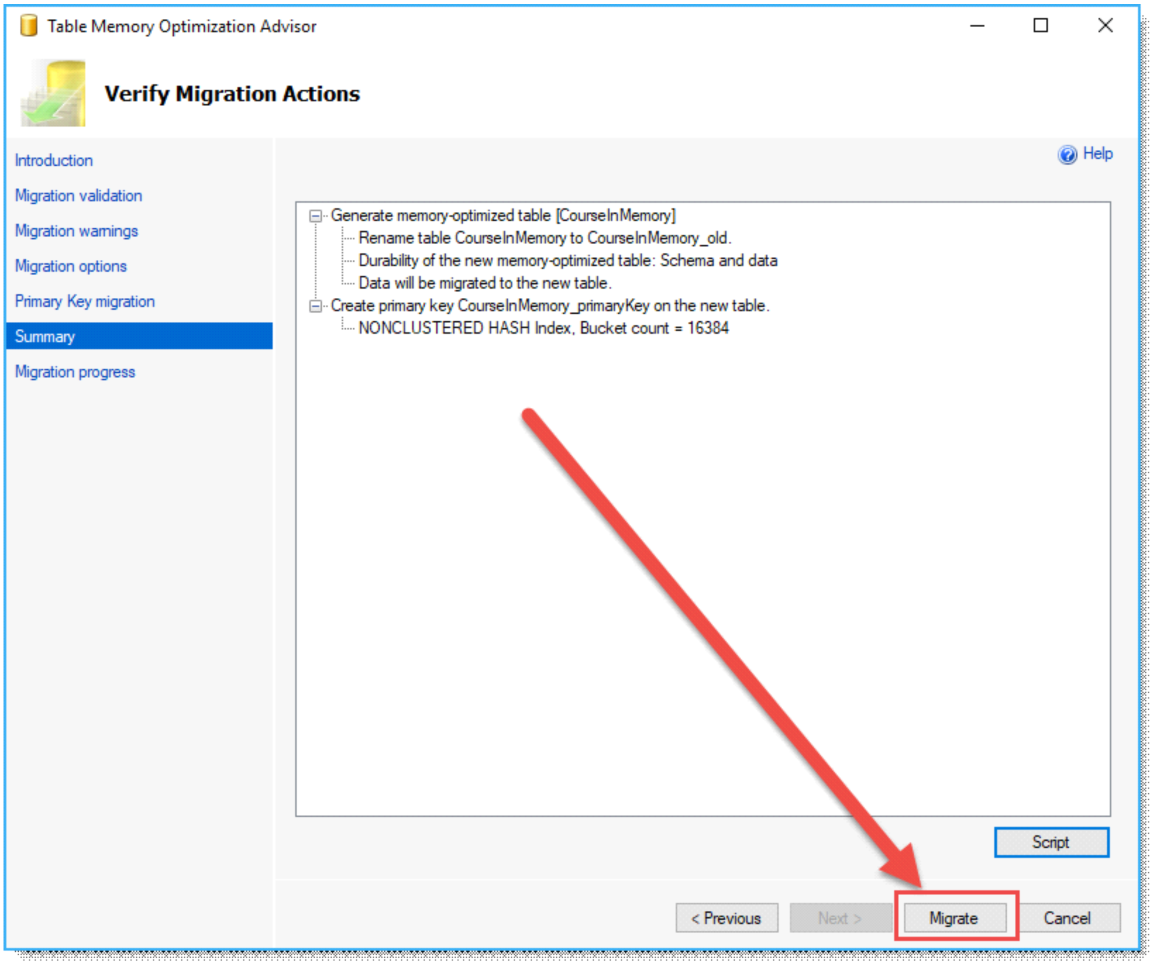
The Bucket Count of a NONCLUSTERED HASH index is the number of buckets in the hash table. It is recommended to set the fill factor to 50 to 60% if the table requires a lot of space for growth. Bucket Count will be rounded up to the nearest power of two.

○ Use NONCLUSTERED index

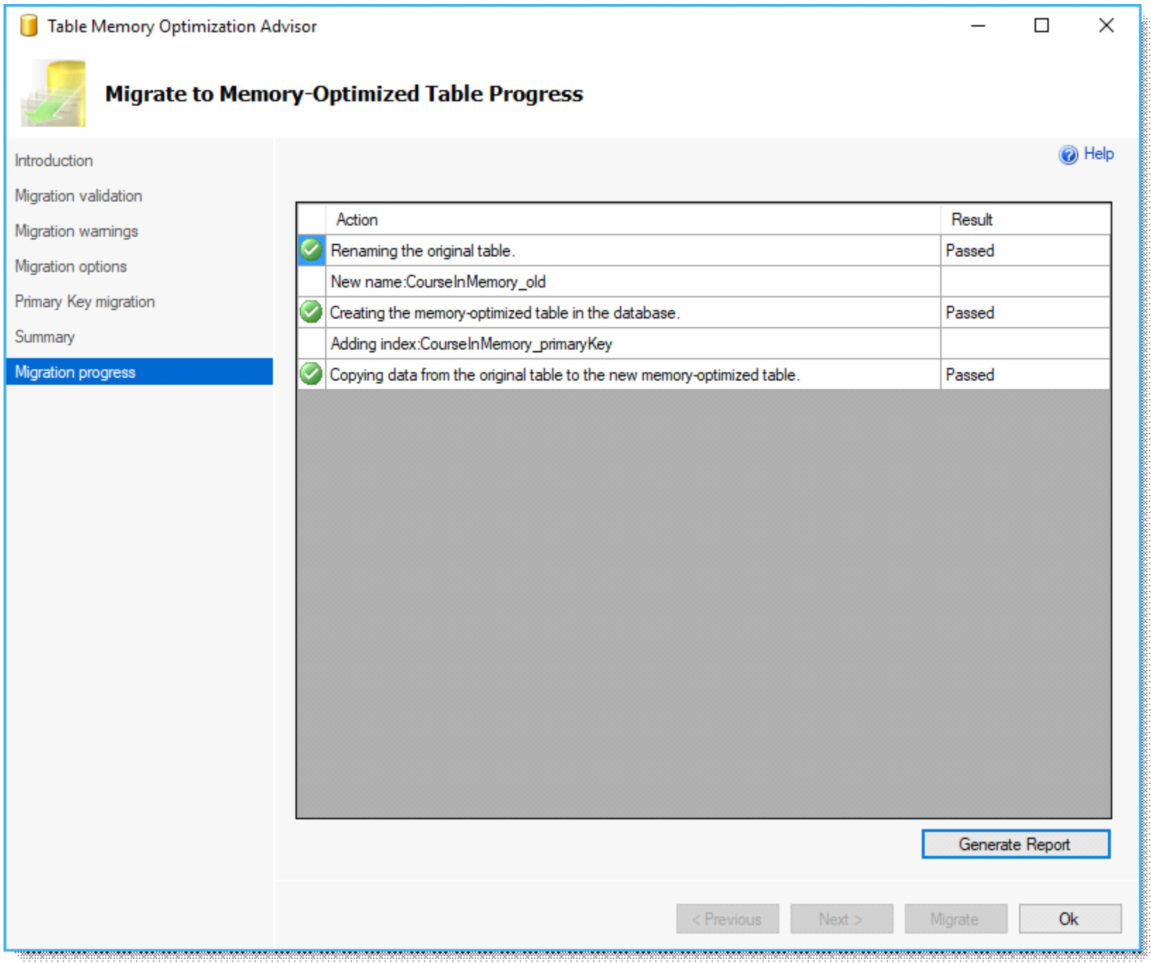
A NONCLUSTERED index provides the most benefit for range predicates and ORDER BY clauses. NONCLUSTERED indexes are unidirectional. It provides no benefit for ORDER BY clauses with orders different from the index.

Sort column and order:

Column	Sort Order



```
⊟ Generate memory-optimized table [CourseInMemory]
    Rename table CourseInMemory to CourseInMemory_old.
    Durability of the new memory-optimized table: Schema and data
    Data will be migrated to the new table.
⊟ Create primary key CourseInMemory_primaryKey on the new table.
    NONCLUSTERED HASH Index, Bucket count = 16384
```

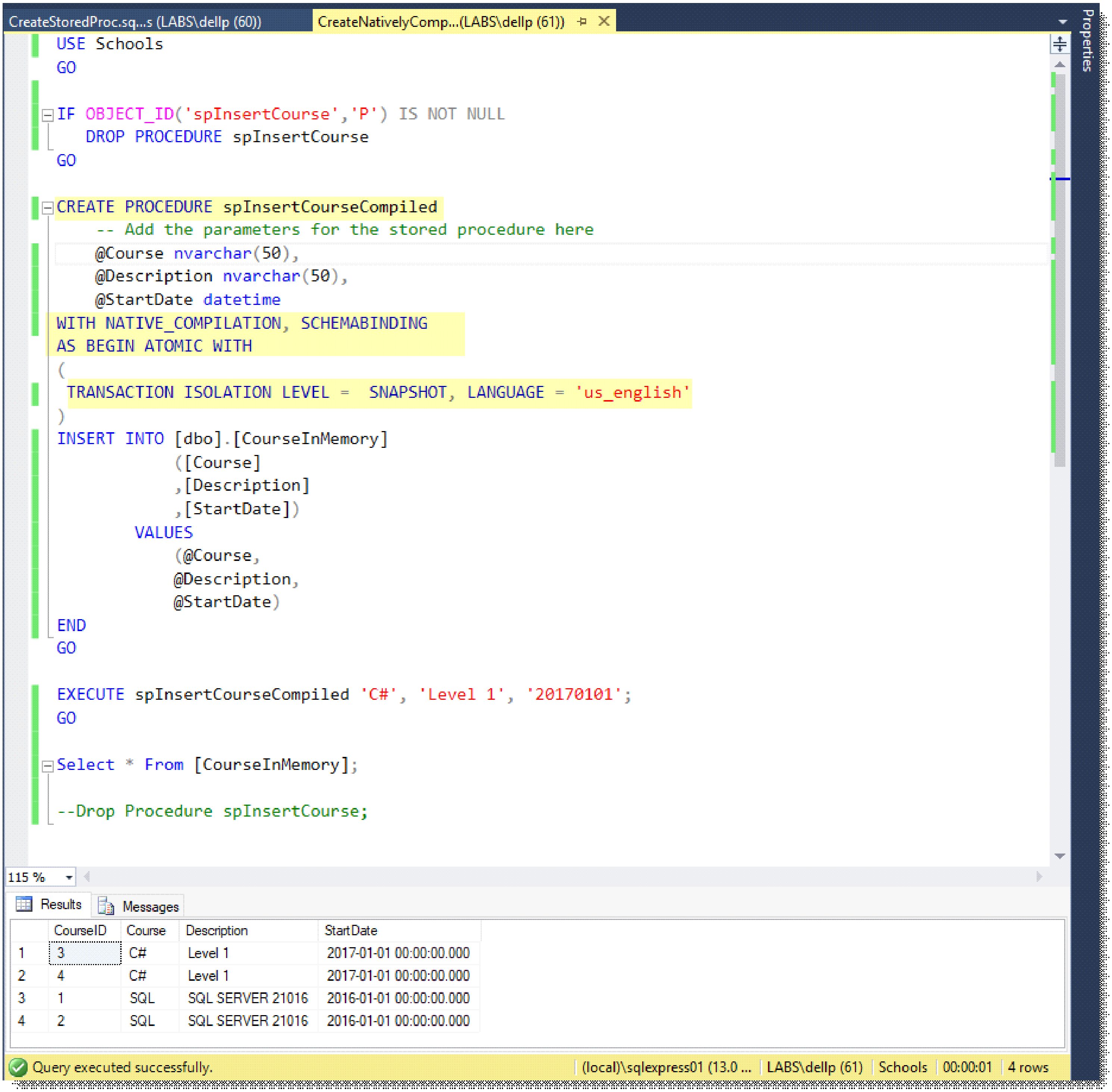


Migrate to Memory-Optimized Table Progress

	Action	Result
✓	Renaming the original table.	Passed
	New name:CourseInMemory_old	
✓	Creating the memory-optimized table in the database.	Passed
	Adding index:CourseInMemory_primaryKey	
✓	Copying data from the original table to the new memory-optimized table.	Passed

Convert to Natively Compiled

Sunday, May 28, 2017 11:14 PM


CreateStoredProc.sq...s (LABS\dellp (60))
CreateNativelyComp...(LABS\dellp (61))
Properties
115 %
Results
Messages

```sql
USE Schools
GO

IF OBJECT_ID('spInsertCourse','P') IS NOT NULL
   DROP PROCEDURE spInsertCourse
GO

CREATE PROCEDURE spInsertCourseCompiled
    -- Add the parameters for the stored procedure here
    @Course nvarchar(50),
    @Description nvarchar(50),
    @StartDate datetime
WITH NATIVE_COMPILATION, SCHEMABINDING
AS BEGIN ATOMIC WITH
(
 TRANSACTION ISOLATION LEVEL =  SNAPSHOT, LANGUAGE = 'us_english'
)
INSERT INTO [dbo].[CourseInMemory]
            ([Course]
            ,[Description]
            ,[StartDate])
        VALUES
            (@Course,
            @Description,
            @StartDate)
END
GO

EXECUTE spInsertCourseCompiled 'C#', 'Level 1', '20170101';
GO

Select * From [CourseInMemory];

--Drop Procedure spInsertCourse;
```

	CourseID	Course	Description	StartDate
1	3	C#	Level 1	2017-01-01 00:00:00.000
2	4	C#	Level 1	2017-01-01 00:00:00.000
3	1	SQL	SQL SERVER 21016	2016-01-01 00:00:00.000
4	2	SQL	SQL SERVER 21016	2016-01-01 00:00:00.000

Query executed successfully. | (local)\sqlexpress01 (13.0 ... | LABS\dellp (61) | Schools | 00:00:01 | 4 rows