Load Data Faster into SQL Server Tables

Strategies and techniques to optimize bulk data loading operations in SQL Server for database professionals

Haripriya Naidu

She/Her

Lead Database Administrator

DBAVUG - 6/11/2025

Haripriya Naidu

- SQL Database Administrator
- 11 years
- AWS Certified Solutions Architect
- Performance Tuning
- SQL Server Engine Internals
- Speaker/Blogger

gohigh.substack.com

linkedin.com/in/haripriya-naidu1215/

github.com/haripriyasb/LoadDataFaster

Agenda

- Explore different ways to load bulk data
  - INSERT INTO
  - SELECT INTO
  - BULK INSERT
  - bcp utility
  - dbatools
  - Import/Export Wizard
- How to make them faster – hints, settings
- When to use which method

What is the problem I'm solving

- Worked on a project involving moving GBs of data
- Explored bulk loading options
- Needed to reduce runtime

What is Bulk Data

- Large volumes of data
- Move into or out of a SQL Server table quickly and efficiently

Examples:

- Moving data from database to another
- Loading a CSV file with 1 million customer records into a table

Performance Challenges

Why it is slow

- Row by row transaction logging
- Serial processing

How to speed it up

- Minimal logging – reduces transaction log overhead
- Parallel processing – multiple threads to load data

Why loading fast matters

- Reduce maintenance window
- Lower resource utilization
- Faster ETL processes
- Cost savings

Methods to Load Data

Insert Into
Select Into
Bulk Insert
bcp utility
dbatools
Import/Export Wizard

1. Insert into

```
INSERT INTO  dbo.TargetTable
SELECT * FROM dbo.SourceTable
```

Slower because:

- Fully logged
- Serial processing

Characteristics:

- Exclusive(X) lock
- No batchsize or rows per batch control

1.1 Insert into with Tablock

```
INSERT INTO  dbo.TargetTable WITH (TABLOCK)
SELECT * FROM dbo.SourceTable
```

- Speeds up the insert
- For bulk loads

What is TABLOCK hint

- Query hint you can add to insert statements
- Locks the whole table

```
INSERT INTO  TargetTable
SELECT * FROM SourceTable;

INSERT INTO  TargetTable WITH (TABLOCK)
SELECT * FROM SourceTable;
```

When to use TABLOCK

- Bulk loading into an empty or staging table/temp table
- Stored procedures
- Nightly ETL jobs
- Data archiving jobs

Benefits of TABLOCK hint

Regular INSERT

- Writes everything to the log file - Fully logged

INSERT WITH TABLOCK

- Writes almost nothing to the logfile - Minimally logged

Benefits of TABLOCK hint

Regular INSERT

- Serial operation - Only one thread inserts data

INSERT WITH TABLOCK

- Parallel operation - Multiple threads insert data concurrently

Locking behavior

- REGULAR INSERT
  - Row-level (RID or KEY) or page-level (PAG) locks.
  - Less blocking

- INSERT WITH TABLOCK HINT
  - Table-level lock (TAB)
  - Blocks other queries

Scenarios to use Tablock

- TABLOCK hint is **best** used
  - SIMPLE, BULK-LOGGED Recovery Model
  - Heap tables
  - Columnstore clustered indexes (OLAP)
  - Temp tables or Staging data
  - Huge amount of data

TABLOCK hint can still be used

- Efficiently in FULL Recovery Model

When NOT to use TABLOCK

- TABLOCK hint should **NOT** be used
  - When concurrency is important
  - Highly volatile tables in OLTP
  - Clustered and non-clustered indexes on tables

Recap – Insert into

- Insert into
  - Fully logged
  - Serial operation
- Make it faster by adding tablock hint
  - Minimally logged
  - Parallel operation
- Only one insert..select operation can run at a time on a table

2. Select into

```
SELECT id,a,b INTO dbo.TargetTable
FROM dbo.SourceTable;
```

- Produces a new table
- Minimally logged
- Supports parallel operation
- No hint needed, it is already fast

Limitation:

- Cannot control batch size or rows per batch
- Target table must reside on the default filegroup

3.1 Insert into- Openrowset

```
INSERT INTO  TargetTable
SELECT * FROM OPENROWSET(...);
```

- Access remote data by connecting to a remote data source
- Alternative to linked server
- Exclusive(X) lock

3.2 Openrowset - Bulk hint

```
INSERT INTO  TargetTable
SELECT * FROM OPENROWSET(
BULK ‘C:\Data\SourceFile.csv’) AS a;
```

- Built-in BULK provider
- Imports a data file as a single batch
- Control where to start and end reading data
- Control rows per batch, default is whole file
- Need an alias for select with bulk hint

3.3 Openrowset – Order clause

```
INSERT INTO  TargetTable
SELECT * FROM OPENROWSET(
BULK ‘C:\Data\SourceFile.csv’,
ORDER (CustomerID ASC)
) AS a;
```

- If SourceFile.csv is ordered by clustered index CustomerID
- Add Order clause to improve insert performance
- Reduce sorting overhead during the load process

Quick tip - Order clause

- Data must already be ordered when it arrives from the source
- Sorting it before inserting will not increase speed

Concern:

- Sorting data during loading can consume a lot of memory

Solution:

- If source file is not sorted, first load it in a temp table
- Then create an index on temp table, matching the index on target table
- Now push data from temp table to target table

3.4 Openrowset – Tablock

```
INSERT INTO  TargetTable WITH (TABLOCK)
SELECT * FROM OPENROWSET(
BULK ‘C:\Data\SourceFile.csv’,
) AS a;
```

- Runs faster with tablock hint
- Takes X lock
- Not fully but efficiently logged
- Serial operation

Recap - Openrowset

- Add Bulk hint to load csv files
- In clustered tables, if source file is already sorted, add **Order** clause to improve insert performance
- In heap tables, add **Tablock** hint to minimally log and improve insert performance

DEMO

1. Insert..Select

   vs

   Insert..Select using TABLOCK hint

2. Select Into

4. Bulk Insert

```
BULK INSERT dbo.TargetTable
FROM ‘C:\Data\SourceFile.csv’
WITH (
    FIELDTERMINATOR = ',',    -- for a comma-separated CSV
    ROWTERMINATOR = '\n',     -- rows end with newline
    FIRSTROW = 2 ,            -- skip header row
);
```

- Import data from a text or csv file into a table or view
- No export
- Uses Sqlservr.exe - Fast way to load data into SQL Server
- Used in stored procedures
- Exclusive(X) lock on the table
- Minimally logged
- Serial operation
- Control batch size

4.1 Bulk Insert - Tablock

```
BULK INSERT dbo.TargetTable
FROM ‘C:\Data\SourceFile.csv'
WITH (
    FIELDTERMINATOR = ',',    -- for a comma-separated CSV
    ROWTERMINATOR = '\n',     -- rows end with newline
    FIRSTROW = 2 ,            -- skip header row
TABLOCK
);
```

- Runs twice as fast
- Takes Bulk Update(BU) lock
- Concurrent bulk inserts into the same table is possible

4.2 Bulk Insert – Native format

```
BULK INSERT dbo.TargetTable
FROM 'C:\Data\SourceFile.dat'
WITH (
    DATAFILETYPE = 'native',
    TABLOCK
     );
```

- Importing a native format file greatly improves insert performance

4.3 Bulk Insert – Indexed tables

```
BULK INSERT dbo.TargetTable
FROM 'C:\Data\SourceFile.dat'
WITH (
    DATAFILETYPE = 'native',
    ORDER (ObjectID ASC)
);
```

- In this case, source file is ordered by **ObjectID** in ascending order
- Target table has clustered index on **ObjectID**
- Use order clause to speed up insert performance
- Sorting overhead is reduced

Recap- Bulk Insert

- Only imports, no exports
- Import from csv or dat or txt files
- For faster performance:
  - Use native format as source file
  - Use ORDER clause for indexed tables if source file is sorted
    - Concurrent bulk inserts are possible
    - Have non-overlapping ranges in key column of input data
  - Use TABLOCK for heap tables
    - Concurrent bulk inserts are possible
    - Holds BU locks which are compatible with other BU locks

Quick Note - Insert Bulk

```
INSERT BULK dbo.MyTable (
    ID int,
    Name nvarchar(100),
    Salary float
)
```

- This is not the same syntax as ‘Bulk Insert’
- Insert Bulk - Internal syntax used by ‘data move’ software or monitoring software
- Not written for manual operations

5. bcp

```
bcp DBATest.dbo.TargetTable1 out "C:\Data\SourceTable.dat" -n -S .\SQL2022 -T
```

- Command-line tool
- Export to file from SQL server table and vice versa
- Import into SQL Server table from file and vice versa
- The tool is built using the bulk API

5.1 bcp – Native Format

```
bcp DBATest.dbo.TargetTable1 out "C:\Data\SourceTable.dat" -n -S .\SQL2022 –T
```

- Use native format to export, so the import next time can be faster
- This writes a binary native format .dat file to disk
- Faster ‘bulk insert’ with .dat file

5.2 bcp – Order to Export

```
bcp “DBATest.dbo.TestTable” out “C:\Data\Employees.dat” -n -c –t, -S
.\SQL2022 –T
```

Ordering is not guaranteed

```
bcp “SELECT id, text FROM DBATest.dbo.TestTable ORDER BY ObjectId” queryout
"C:\Data\ResultFile.dat" -n -c –t, -S .\SQL2022 –T
```

ORDER BY – ordering is guaranteed

Now ResultFile is ordered by ObjectID

When this file is used as source to import, it improves performance in case of indexed tables

5.3 bcp – Order to Import

```
bcp DBATest.dbo.TestTable in C:\Data\ResultFile.dat -c –t, -S .\SQL2022 -T -E -h ORDER(ObjectID)
```

- Performance improves when
  - If source data file is sorted according to clustered index on target table
  - And when ORDER clause is used
  - Reduces sorting overhead

- Sorted file + ORDER clause = faster insert
- Unsorted file + ORDER clause = won't run faster
- By default, bcp assumes the data file is unordered

5.4 bcp - Tablock hint

```
bcp DBATest.dbo.HeapTable IN C:\Data\SortedFile.bcp -b 5000 -h "TABLOCK"
-S .\SQL2022
```

- Improves performance via minimal logging
- Holds Bulk Update(BU) lock
- Concurrent loads are possible

Recap - bcp

- Export using Native format and ‘queryout’ with ‘order by’ clause
- Import using formatted file for faster insert
- Use Order hint for clustered tables
- Use Tablock hint for heap tables

6. dbatools

- Open-source PowerShell module to automate SQL Server tasks
- Data migration, backups, restores etc.

Data Move:

- Write-DbaDbTableData
- Copy-DbaDbTableData
- Import-DbaCsv

6.1 Write-DbaDbTableData

```
$DataTable = Invoke-DbaQuery -SqlInstance sql1 -Database dbatest -Query "SELECT *
FROM t1"

Write-DbaDbTableData -SqlInstance sql2 -InputObject $DataTable -Database dbatest -
Table dbo.t1
```

- Writes data to SQL Server table using SQL Bulk Copy
- Import into table from csv or from table to table
- This will buffer the contents of that table in memory of the machine running the commands.
- By default, this operation will lock the destination table while running.

6.2 Copy-DbaDbTableData

```
Copy-DbaDbTableData -SqlInstance Server1 -Destination Server2 -Database dbatest
-Table dbo.t1
```

- Copies data between SQL Server tables using SQL Bulk Copy.
- Faster than Write-DbaDbTableData
- Least resource-intensive way
- Does not buffer the contents in memory of the machine running commands
- By default, this operation will lock the destination table while running.
- Specify –NoTableLock if needed, but makes it slower
- Default batchsize = 50000

6.4 Import-DbaCsv

```
PS C:\> Import-DbaCsv -Path C:\SourceFile.csv -SqlInstance sql1 -Database mydb
```

- .NET's super fast SqlBulkCopy class to import CSV files into SQL Server.
- The entire import is performed within a transaction
- Default is row locks and does not lock the entire table
- Use –TableLock parameter to obtain Bulk Update Lock

Recap - dbatools

- Write-DbaDbTableData
  - From csv or table to table
- Copy-DbaDbTableData
  - Faster to copy data from one server to other
- Import-DbaCsv
  - Faster option to import from csv to table

DEMO

7. Import and Export Wizard

- Regular insert like operation – fully logged and serial operation
- Ad-hoc loads from one server to another
- Save SSIS package if needed

Summary

Key Factors to Fast Load

- Minimal logging
  - Write less/nothing to logfile
- Parallel insertion
  - Multiple threads insert data
- Bulk Update Lock
  - Concurrent inserts
  - Bcp, bulk insert
- Import or export ordered data for indexed tables

Locks

- Exclusive(X) locks
  - Only one insert at a time
  - X lock not compatible with any lock
  - Insert..Into
  - Select..Into
- Bulk Update(BU) locks
  - Concurrent inserts
  - BU lock compatible with other BU locks
  - bcp
  - Bulk insert

Load faster for FULL recovery model

- Change recovery model
  - If FULL, set to bulk_logged
  - Perform insert
  - Revert to FULL

```
-- Switch to BULK_LOGGED to perform minimal logging for bulk insert
ALTER DATABASE DB SET RECOVERY BULK_LOGGED;

-- Perform BULK INSERT after switching to BULK_LOGGED
BULK INSERT dbo.Table
FROM 'C:\Data\SourceFile.dat'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    TABLOCK
);

-- Revert to FULL recovery model after bulk insert
ALTER DATABASE DB SET RECOVERY FULL;
```

Load faster for indexed tables

- For indexed tables
  - Disable indexes
  - Perform insert
  - Enable indexes

```
-- Disable all indexes or Clustered index
ALTER INDEX ALL ON Orders DISABLE;

-- Perform BULK INSERT after disabling indexes
BULK INSERT dbo.Table
FROM 'C:\Data\SourceFile.dat'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    TABLOCK
);
-- Rebuild to enable all indexes
ALTER INDEX ALL ON Orders REBUILD;
```

Fast Load - By Method & Optimization

Insert Into

- Add tablock hint for minimal logging and parallel operation

Select..Into

- Already fast

Openrowset

- Add Bulk hint to load csv
- Use ORDER clause for indexed tables
- Use TABLOCK hint for heap tables

Fast Load - By Method & Optimization

Bulk Insert

- Use native format as source file
- Use ORDER clause for indexed tables
- Use TABLOCK hint for heap tables

bcp

- Export using Native format and ‘queryout’ with ‘order by’ clause
- Import using formatted file and ORDER hint for clustered tables
- Use TABLOCK hint for heap tables

Fast Load - By Method & Optimization

dbatools

- Copy-DbaDbTableData
  - Faster to copy data from one server to other
- Import-DbaCsv
  - Faster option to import from csv to table

Fast Load – By Scenario

- From one table to another table in **same** server
  - Insert Into..with tablock
  - Select Into
  - dbatools: Copy-DbaDbTableData
- From one table to another table in **different** server
  - dbatools: Copy-DbaDbTableData
- From file to table
  - Bulk Insert
  - Bcp
  - dbatools: Import-DbaCsv
- Ad-hoc load
  - Import/Export Wizard

Method	INSERT INTO	BULK INSERT	SELECT INTO	BCP Utility	dbatools	Import/Export Wizard
Use TABLOCK?	Improves performance	Improves performance	No	Improves performance	Default for Copy-Data	No
Concurrent Inserts	No	Yes (with TABLOCK)	No	Yes (with TABLOCK)	No	No
Batch Size Control	No	Yes	No	Yes	Yes	Yes
Supports Import & Export	Insert only	Import only	Insert only	Both Import and Export	Both Import and Export	Both
From/To CSV	No/No	Yes/No	No/No	Yes/Yes	Yes/Yes	Yes/Yes
Bulk Load from SQL to SQL	Yes	No	Yes	No	Yes	Yes but smaller loads
Bulk Load from File to SQL	No	Yes	No	Yes	Yes	Yes but smaller loads

Resources

- The Data Loading Performance Guide | Microsoft Learn
- https://gohigh.substack.com/p/insert-with-tablock-hint-faster?r=vmpc0
- https://gohigh.substack.com/p/insert-with-tablock-hint-tables-with?r=vmpc0
- https://learn.microsoft.com/en-us/sql/t-sql/statements/insert-transact-sql?view=sql-server-ver16#using-insert-intoselect-to-bulk-import-data-with-minimal-logging-and-parallelism
- https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms190203(v=sql.105)
- https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms191244(v=sql.105)
- https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2008/dd425070(v=sql.100)?redirectedfrom=MSDN

Questions?

Thank you!

 gohigh.substack.com

 linkedin.com/in/haripriya-naidu1215/

 github.com/haripriyasb/LoadDataFaster