

Writing Faster SQL

Bill Graziano

Copyright © 2011 scaleSQL Consulting, LLC

Thanks to Our Sponsors

Gold Sponsors



Silver Sponsors



Product Sponsors



Goals

Make TSQL statements fast

Run TSQL in a fast way

INDEX!

SARGABILITY

```
SELECT      SalesOrderID,  
            OrderDate  
FROM        Sales.SalesOrderHeader  
WHERE       YEAR(OrderDate) = 2003  
AND         MONTH(OrderDate) = 7  
GO  
  
SELECT      SalesOrderID,  
            OrderDate  
FROM        Sales.SalesOrderHeader  
WHERE       OrderDate  
            BETWEEN '7/1/2003' AND '7/31/2003'  
GO
```

What to Index?

```
SELECT      SUM(D.LineTotal) AS LineTotal
FROM        Sales.SalesOrderHeader H
JOIN        Sales.SalesOrderDetail D
            ON D.SalesOrderID = H.SalesOrderID
WHERE       H.CustomerID = 32345
AND         H.[Status] = 5
```

- Selective columns in the WHERE clause
- Columns used to JOIN tables
- 7 indexes per table is a good guideline

INCLUDE Columns

```
SELECT      SUM(D.LineTotal) AS LineTotal
FROM        Sales.SalesOrderHeader H
JOIN        Sales.SalesOrderDetail D
            ON D.SalesOrderID = H.SalesOrderID
WHERE       H.CustomerID = 32345
AND         H.[Status] = 5
```

- Index on ClientID and include Status
- Covering Index: Doesn't reference table

Filtered Indexes

```
SELECT      SUM(D.LineTotal) AS LineTotal
FROM        Sales.SalesOrderHeader H
JOIN        Sales.SalesOrderDetail D
ON          D.SalesOrderID = H.SalesOrderID
WHERE       H.CustomerID = 32345
AND         H.[Status] = 5
```

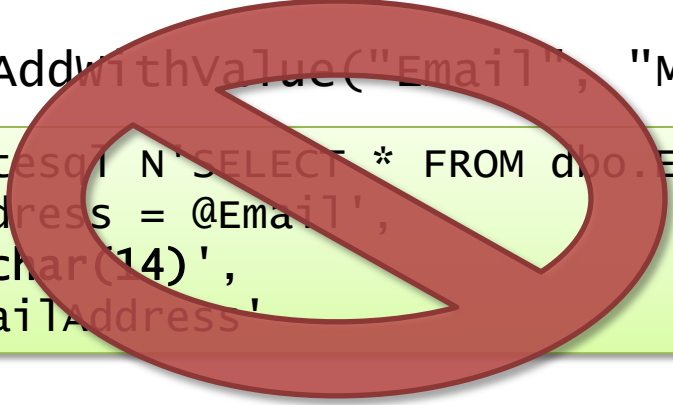
- Index on ClientID WHERE Status = 5
- Covering Index: Doesn't reference table
- Much smaller index

Indexing Tips

- Column order matters!
 - Most selective columns first
- Test on large datasets
- Index `PERSISTED`, `COMPUTED` columns
 - Must match the `WHERE` clause
 - Pre-2008 “date only” columns
- Always include a clustered index

Matching Data Types

```
cmd1.Parameters.AddWithValue("Email", "MyEmailAddress");
```



```
exec sp_executesql N'SELECT * FROM dbo.EmailLoad  
WHERE EmailAddress = @Email',  
N'@Email nvarchar(14)',  
@Email=N'MyEmailAddress'
```

```
cmd2.Parameters.Add("Email", SqlDbType.VarChar, 128).value =  
    "MyEmailAddress";
```

```
exec sp_executesql N'SELECT * FROM dbo.EmailLoad  
WHERE EmailAddress = @Email',  
N'@Email varchar(128)',  
@Email='MyEmailAddress'
```

Optional Parameters

SalesPersonID =
COALESCE(@SalesPersonID, SalesPersonID)

TABLE SCAN!

(CustomerID = @CustomerID
OR @CustomerID IS NULL)

TABLE SCAN!

Use sp_executeSQL and add your parameters

Indexes!

Finding Queries to Tune

- Profiler
 - Filter for high Reads, CPU or Duration
- SQL Server Management Studio
 - Server Management Reports – Top Queries
 - Activity Monitor – Recent Expensive Queries
- ClearTrace & TraceTune.com
 - Shows highest resource consumers in aggregate

EXISTS Short Circuits

```
DECLARE @Rows INT;
```

```
SELECT @Rows = COUNT(*)  
FROM Sales.SalesOrderHeader  
WHERE Status = 5;
```

```
IF @Rows > 0 PRINT 'Found!';  
GO
```

```
IF EXISTS (SELECT 1 FROM  
           Sales.SalesOrderHeader  
           WHERE Status = 5)  
    PRINT 'Found!';  
GO
```

MERGE Statement

- Single statement INSERT, UPDATE and DELETE
- Single pass through table
- Target a CTE for a subset of the target table

T-SQL Tips

- Use table variables under 100 rows
- Never use cursors
- Cursors should be read-only
- Table-valued parameters
- Use OUTPUT parameters
- Avoid NOT IN

T-SQL Tips

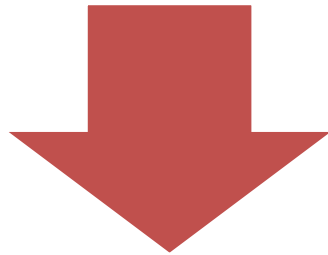
- Be careful with DISTINCT
- Use user-defined functions carefully
 - Don't SELECT!
- Use RANKing functions
- Prefix object name with schema
- OR can slow the WHERE clause

Set Based Processing



One UPDATE

Five Rows



Five UPDATES

One Row Each

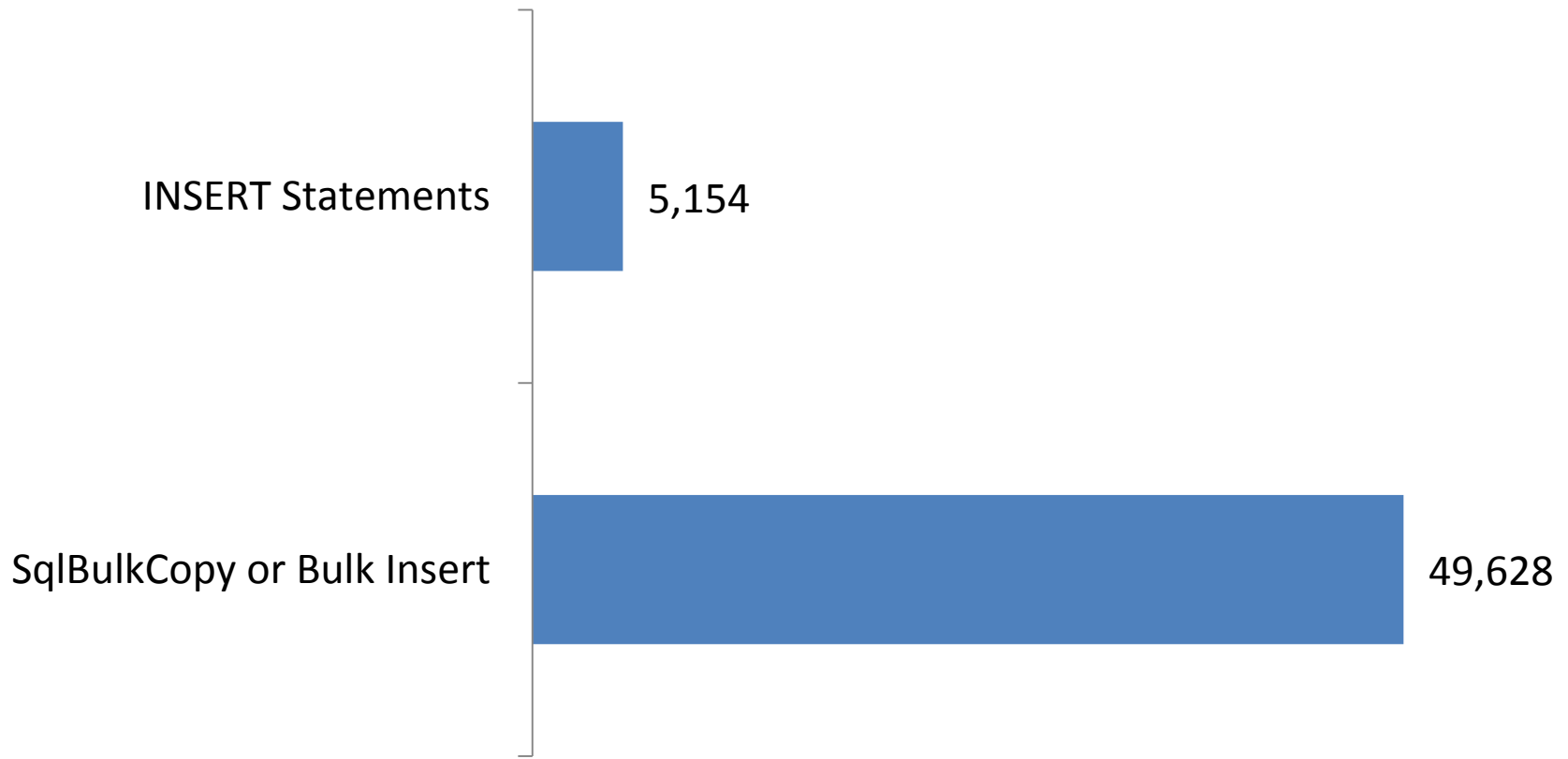
- Prefer derived tables & CTE to temp tables
- Avoid writing loops

Reduce Query Compiles

- PerfMon
 - Plan Cache :: Cache Hit Ratio (90% +)
 - SQL Statistics :: SQL Compilations / second
- Profiler
 - Performance :: Showplan XML for Query Compile
- Use stored procedures
- Parameterize queries at the client
- Forced parameterization

Insert in Bulk

Rows Inserted per Second



DELETE in Batches

```
WHILE EXISTS (SELECT * FROM dbo.EmailLoad)
BEGIN
    DELETE TOP (500) FROM dbo.EmailLoad;
    WAITFOR DELAY '00:00:01';
END
```

Process Asynchronously

- Change Tracking
- Change Data Capture (Enterprise Edition)
- Service Broker
- READPAST Query hint

The SQL that never runs
is the fastest SQL of all.

CACHE

Thank You

billg@scaleSQL.com

