

Lab 17: Performance tuning with Indexes.

1. Create the following table:

```
CREATE TABLE TestForIndex
(
    TestID          INT          PRIMARY KEY,
    Column1         INT          NOT NULL
);
```

Answer:

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the 'ssingh11Database' expanded, with 'Tables' > 'dbo.TestForIndex' selected. The table's columns are listed: 'TestID (PK, int, not null)' and 'Column1 (int, not null)'. The main window shows the execution of the 'CREATE TABLE TestForIndex' script, which completed successfully. The Messages pane at the bottom confirms 'Command(s) completed successfully.' The status bar at the bottom indicates the connection is to 'MSSQL-01.ad.syr.edu ... ssingh11 (56) ssingh11Database' with 0 rows.

- Write a stored procedure that will insert 1 million records into the TestForIndex table. The values for both testId and column1 should start at 0 and increase by 1.

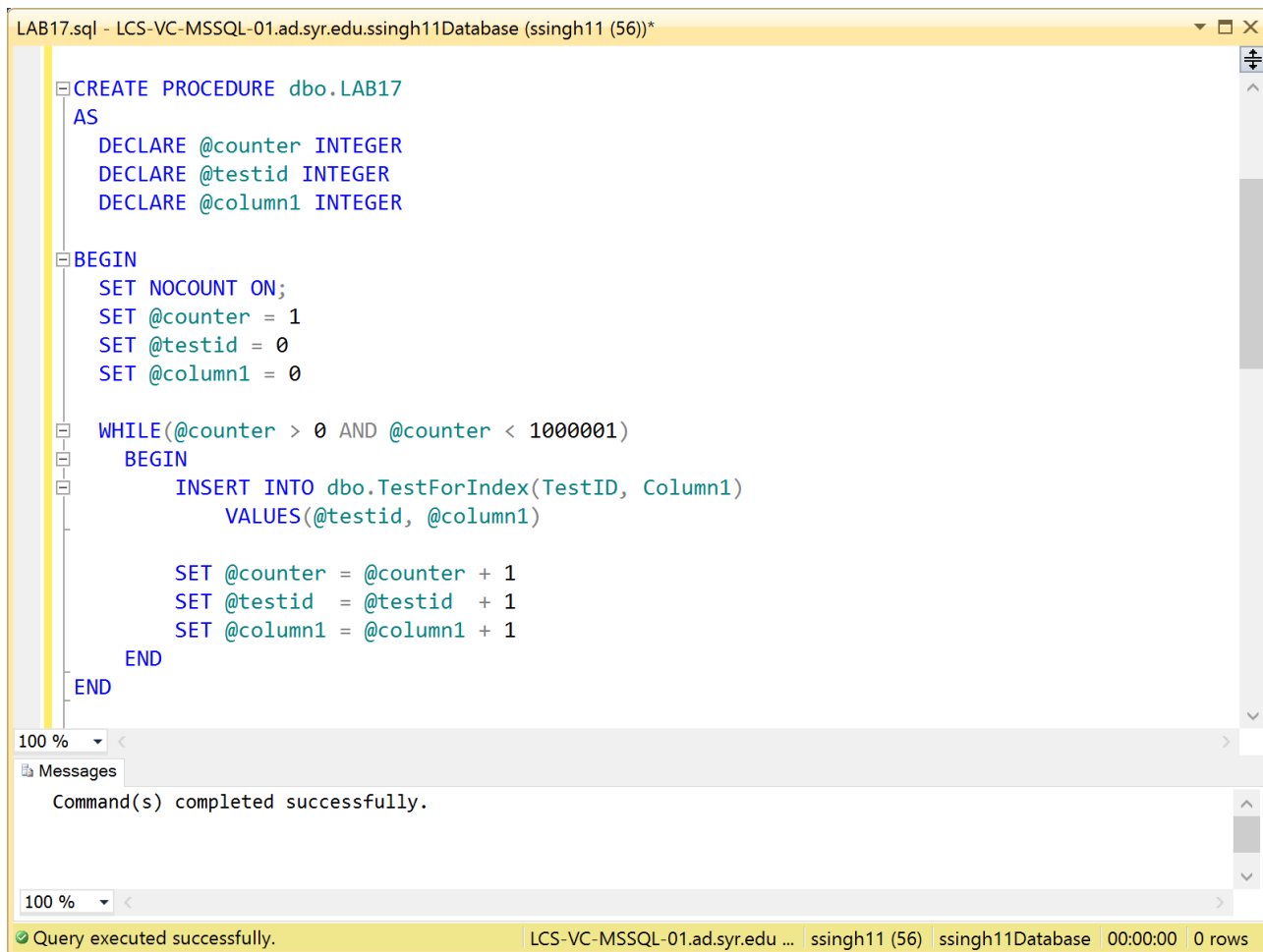
Answer:

```
CREATE PROCEDURE dbo.LAB17
AS
    DECLARE @counter INTEGER
    DECLARE @testid INTEGER
    DECLARE @column1 INTEGER

BEGIN
    SET NOCOUNT ON;
    SET @counter = 1
    SET @testid = 0
    SET @column1 = 0

    WHILE(@counter > 0 AND @counter < 1000001)
    BEGIN
        INSERT INTO dbo.TestForIndex(TestID, Column1)
            VALUES(@testid, @column1)

        SET @counter = @counter + 1
        SET @testid = @testid + 1
        SET @column1 = @column1 + 1
    END
END
```



The screenshot shows a SQL Server Enterprise Manager window titled "LAB17.sql - LCS-VC-MSSQL-01.ad.syr.edu.ssingh11Database (ssingh11 (56))*". The main pane displays the SQL code for the stored procedure "dbo.LAB17", which is identical to the code provided in the previous block. The code is color-coded and includes a tree view on the left. Below the code pane, there is a "Messages" pane showing the message "Command(s) completed successfully." and a status bar at the bottom indicating "Query executed successfully." and "0 rows" affected.

```
LAB17.sql - LCS-VC-MSSQL-01.ad.syr.edu.ssingh11Database (ssingh11 (56))*
```

```
CREATE PROCEDURE dbo.LAB17
AS
    DECLARE @counter INTEGER
    DECLARE @testid INTEGER
    DECLARE @column1 INTEGER

BEGIN
    SET NOCOUNT ON;
    SET @counter = 1
    SET @testid = 0
    SET @column1 = 0

    WHILE(@counter > 0 AND @counter < 1000001)
    BEGIN
        INSERT INTO dbo.TestForIndex(TestID, Column1)
            VALUES(@testid, @column1)

        SET @counter = @counter + 1
        SET @testid = @testid + 1
        SET @column1 = @column1 + 1
    END
END
```

100 %

Messages

Command(s) completed successfully.

100 %

Query executed successfully. LCS-VC-MSSQL-01.ad.syr.edu ... ssingh11 (56) ssingh11Database 00:00:00 0 rows

3. Execute the SP. Once it finishes running, verify that the data was inserted.

Answer:

```
EXEC dbo.LAB17;
```

LAB17.sql - LCS-VC-MSSQL-01.ad.syr.edu.ssingh11Database (ssingh11 (56))*

```
EXEC dbo.LAB17;
```

100 %

Messages

Command(s) completed successfully.

100 %

Query executed successfully. LCS-VC-MSSQL-01.ad.syr.edu ... ssingh11 (56) ssingh11Database 00:13:08 0 rows

```
SELECT * FROM TestForIndex;
```

LAB17.sql - LCS-VC-MSSQL-01.ad.syr.edu.ssingh11Database (ssingh11 (51))*

```
SELECT * FROM TestForIndex;
```

100 %

Results Messages

	TestID	Column1
1	0	0
2	1	1
3	2	2
4	3	3
5	4	4
6	5	5
7	6	6
8	7	7
9	8	8
10	9	9
11	10	10
12	11	11
13	12	12
14	13	13
15	14	14
16	15	15
17	16	16

Query executed successfully. LCS-VC-MSSQL-01.ad.syr.edu ... ssingh11 (51) ssingh11Database 00:00:03 1000000 rows

4. Run the following selects from the table, take note of the time it took to select the data. Run each statement 10 times, Average out the rest of the runs (for both the CPU time and the overall times).

```
SET STATISTICS TIME ON
SELECT *
    FROM TestForIndex a, TestForIndex b
    WHERE a.testID = b.column1;
SET STATISTICS TIME OFF
```

INDEX	CPU TIME	ELAPSED TIME
1	1124 ms	6914 ms
2	1295 ms	3830 ms
3	1248 ms	3462 ms
4	1294 ms	3780 ms
5	1232 ms	4243 ms
6	1311 ms	3719 ms
7	1591 ms	5578 ms
8	1310 ms	4672 ms
9	1326 ms	4174 ms
10	1311 ms	4106 ms
Average	1304.2	4447.8

The screenshot shows a SQL Server Enterprise Manager window titled "SQLQuery2.sql - LCS-VC-MSSQL-01.ad.syr.edu.ssingh11Database (ssingh11 (53))*". The query editor displays the following SQL code:

```
--1
SET STATISTICS TIME ON
SELECT *
    FROM TestForIndex a, TestForIndex b
    WHERE a.testID = b.column1;
SET STATISTICS TIME OFF
```

The Results pane shows the execution results:

(1000000 row(s) affected)

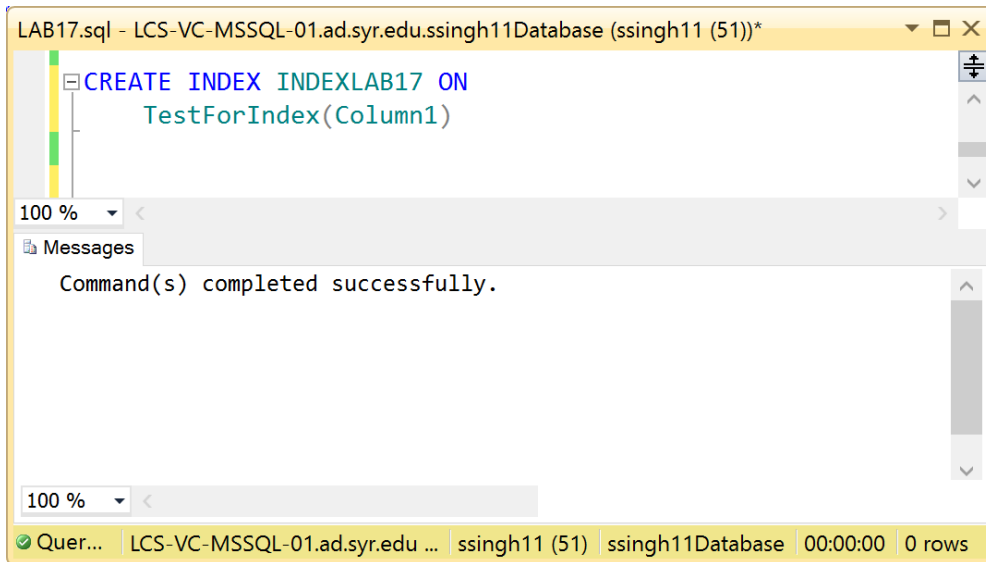
SQL Server Execution Times:
CPU time = 1124 ms, elapsed time = 6914 ms.

The status bar at the bottom indicates: "Query executed successfully. LCS-VC-MSSQL-01.ad.syr.edu ... ssingh11 (53) ssingh11Database 00:00:09 1000000 rows".

5. Create an index on the column1 column.

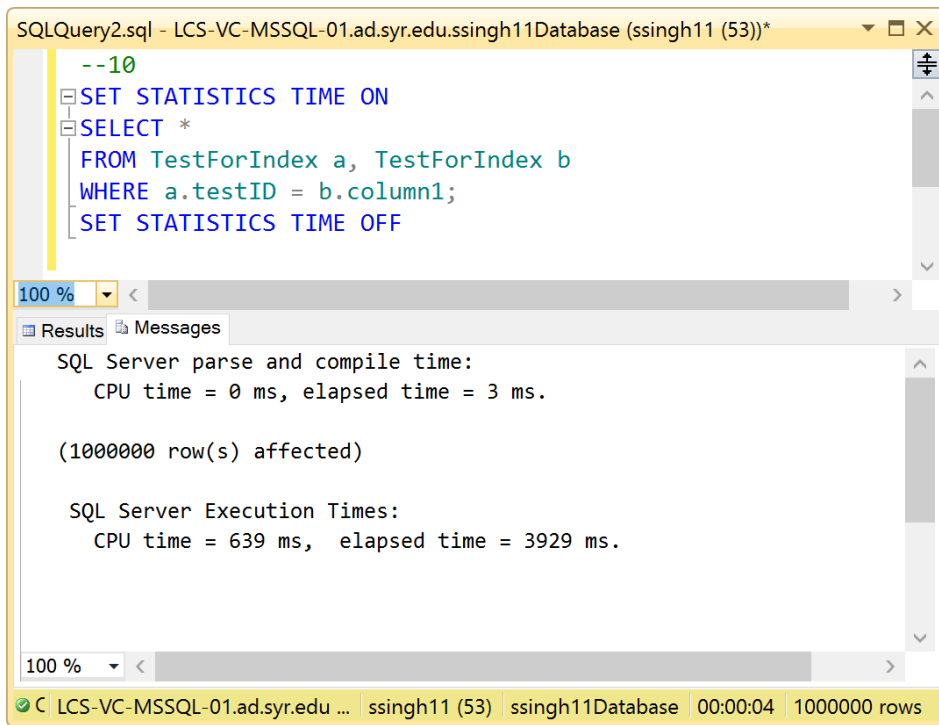
Answer:

```
CREATE INDEX INDEXLAB17 ON  
TestForIndex(Column1)
```



6. Run the selects again, following the same process as in #4.

INDEX	CPU TIME	ELAPSED TIME
1	656	4718
2	578	3705
3	390	4160
4	421	3521
5	577	4432
6	593	4353
7	515	4347
8	531	5614
9	593	4094
10	639	3929
Average:	549.3	4287.3



The screenshot shows a SQL Server Enterprise Manager window titled "SQLQuery2.sql - LCS-VC-MSSQL-01.ad.syr.edu.ssingh11Database (ssingh11 (53))*". The query editor contains the following SQL code:

```
--10
SET STATISTICS TIME ON
SELECT *
FROM TestForIndex a, TestForIndex b
WHERE a.testID = b.column1;
SET STATISTICS TIME OFF
```

Below the query editor, the "Results" tab is selected, displaying the following output:

```
SQL Server parse and compile time:
  CPU time = 0 ms, elapsed time = 3 ms.

(1000000 row(s) affected)

SQL Server Execution Times:
  CPU time = 639 ms,  elapsed time = 3929 ms.
```

The status bar at the bottom indicates the connection is "LCS-VC-MSSQL-01.ad.syr.edu ...", the user is "ssingh11 (53)", the database is "ssingh11Database", the execution time is "00:00:04", and "1000000 rows" were affected.

7. Compute the difference (percentage change and actual change) between the indexed and non-indexed runs. Make sure that the performance increased as expected.

% change in CPU Time = $(1304.2 - 549.3) / 1304.2 = 754.9 / 1304.2 = 0.57882 = 0.57882 * 100 = \underline{\underline{57.8\%}}$

% change in Elapedd Time = $(4447.8 - 4287.3) / 4447.8 = 0.03608 * 100 = \underline{\underline{3.6\%}}$