

# SQL SERVER BEST PRACTICES

Ahmedabad SQL Server User Group

Ahmedabad, Gujarat, India

July 19, 2008

**Pinal Dave**

Microsoft MVP – SQL Server

Founder – [SQLAuthority.com](http://SQLAuthority.com)

[pinal@SQLAuthority.com](mailto:pinal@SQLAuthority.com)

## Speaker Profile

- Microsoft Most Valuable Professional – SQL Server
- Masters of Science (Computer Networks) – University of Southern California, Los Angeles, USA.
- Bachelors of Engineering (Electronics & Communication) – Nirma Institutes of Technology, Ahmedabad, India.
- MCP, MCDBA, MCAD
- Founder – SQLAuthority.com
- 5+ Years of experience as Database Administrator

# || Agenda

- Introduction
- Speaker Profile
- Agenda
- 10 Best Practices for SQL Server
- Summary
- Questions & Answers
- References
- End Note

## 10 Best Practices for SQL Server

- 1) Order of Join
- 2) Usage of Indexes
- 3) Usage Scalar Functions on Conditions
- 4) Usage of WITH (NOLOCK) query hint
- 5) SET NOCOUNT ON/OFF
- 6) Paging in SQL Server
- 7) Separate Hard Drive for TempDB, Indexes
- 8) Normal or De-Normal Database Design
- 9) Several Small Tips
- 10) The 10<sup>th</sup> One

## 1) Order of Join

- If there is more than one table in JOIN operation, the order of joining matters. Smaller (least rows) table should be base table and additional joined table should have increasing capacity (rows, size).
- If smaller table is used earlier in joins it filters out most of the rows in early stage of query, which helps query optimizer to better use indexes for quicker results.

## 2) Usage of Indexes

- Indexes should be considered on all columns that are frequently accessed by the WHERE, ORDER BY, GROUP BY, TOP, and DISTINCT clauses.
- When indexes are created separate table containing index and their pointer rows are created.
- Creating too many indexes may improve performance for few of the queries however; performance of query which contains UPDATE, INSERT or DELETE operation gets reduced.
- Drop indexes which seem not useful.

## 3) Usage Scalar Functions on Conditions

- Using functions forces query optimizer to do table scan and not use indexes available on table. Function results are available to query optimizer at run time and it does not utilize indexes for query results.
- Use calculated columns or computed columns instead of functions. Indexes can be created on computed columns and query performance will be increased.
- E.g. `LTRIM(RTRIM(FieldName))` does not utilize index created on `FieldName`

## 4) Usage of WITH (NOLOCK) query hint

- If table contains data which may not be changing frequently or is non-sensitive data, use query hint WITH (NOLOCK) in the SELECT statement.
- This hint does not lock the database table or database page at all.
- Use of this hint should be done responsibly as it reads dirty data as well.

```
SELECT *  
FROM Production.Product WITH (NOLOCK)  
WHERE ProductID = 1  
GO
```

## 5) SET NOCOUNT ON/OFF

- SET NOCOUNT ON should be at the top of every stored procedure.
- SET NOCOUNT prevents DONE\_IN\_PROC messages (which are almost always discarded, anyway) from being sent to the client for each statement executed in a stored procedure.

***SET NOCOUNT ON***

*SELECT \**

*FROM Production.Product*

*WHERE ProductID = 1*

***SET NOCOUNT OFF***

## 6) Paging in SQL Server

- Returning large resultset to application from database is very resource intensive and network congesting operation.
- Usage of new RANKING (DENSE\_RANK (), RANK () etc) function along with TOP clause can create paging in SQL Server.
- This way SQL Server will return only rows which are required by application.
- Reference : Search “**Paging**” at <http://search.SQLAuthority.com>

## 7) Separate Hard Drive for TempDB, Indexes

- Use of different hard drive controller for tempdb, indexes or different file-group.
- TempDB or Indexes are frequently used in database operations. If tempDB or Indexes lies in the same hard drive as other database objects, database have to share resources associated with single hard drive controller, which sometime deteriorate whole input/output operations of the database.
- Putting them on separate hard drive controller can improve the performance as both the data can be retrieved simultaneously.

## 8) Normal or De-Normal Database Design

- 3<sup>rd</sup> normal form is definitely most recommended form of the database.
- Any higher degree of normalization can affect negatively to database performance.
- If your operations are related to Data Warehousing (DW) or Business Intelligence (BI), do not hesitate to use de-normalized database structure.

## 9) Several Small Tips

- Always have clustered Index on your table.
- Do not use `SELECT * FROM TableName`. Write the name of the column required to application, instead of \*(star).
- Avoid using cursors at any cost. Use `INSERT INTO SELECT` statement structure instead of using cursors to loop over dataset and insert into table.
- Rebuild database at regular interval.
- Take database backup and test it by restoring them often.

# 10) The 10<sup>th</sup> One

Your Suggestions

# Summary



# Questions and Answers



# References

- [Microsoft Book Online](#)
- [SQLAuthority.com](#)
- [SQL Server 2008](#)



## || End Note

- Thank You!
- For any questions or queries contact me at [pinal@SQLAuthority.com](mailto:pinal@SQLAuthority.com)





Microsoft®  
**SQL Server® 2008**

The logo for Microsoft SQL Server 2008, featuring a stylized red and grey wireframe sphere on the left, followed by the text "Microsoft®" in a smaller font and "SQL Server® 2008" in a large, bold font.