**• Use "Pascal" notation for SQL server Objects Like Tables, Views, Stored Procedures. Also tables and views should have ending " s".**

 Example:

 UserDetails

 Emails

**• If you have big subset of table group than it makes sense to give prefix for this table group. Prefix should be separated by \_.**

 Example:

Page\_ UserDetails

Page\_ Emails

**• Use following naming convention for Stored Procedure . sp <Application Name> \_ [<group name> \_] <action type> <table name or logical instance> Where action is: Get, Delete, Update, Write, Archive, Insert ... ie verb**

Example:

spApplicationName\_GetUserDetails

spApplicationName\_UpdateEmails

**• Use following Naming pattern for triggers: TR\_ <TableName> \_ <action> <description>**

Example:

TR\_Emails\_LogEmailChanges

TR\_UserDetails\_UpdateUserName

**• Indexes: IX\_ <tablename> \_ <columns separated by\_>**

Example:

IX\_UserDetails\_UserID

**• Primary Key: PK\_ <tablename>**

Example:

PK\_UserDetails PK\_ Emails

**• Foreign Key: FK\_ <tablename\_1> \_ <tablename\_2>**

Example:

FK\_UserDetails\_Emails

**• Default: DF\_ <table name> \_ <column name>**

Example:

 DF\_ UserDetails \_UserName

**• Avoid use of SELECT \* in SQL queries. Instead practice writing required column names after SELECT statement**.

Example:

SELECT Username ,  
               Password  
FROM    UserDetails;

**• Avoid using temporary tables and derived tables as it uses more disks I / O. Instead use CTE (Common Table Expression); its scope is limited to the next statement in SQL query.**

**• Use SET NOCOUNT ON at the beginning of SQL Batches, Stored Procedures and Triggers. This improves the performance of Stored Procedure.**

**• Properly format SQL queries using indents.**

 Example: Correct Format

SELECT Username ,

              Password

FROM   UserDetails ud

INNER JOIN Employee e ON e.EmpID = ud.UserID;

**• Use upper Case for all SQL keywords.**

 Example:

SELECT, UPDATE, INSERT, WHERE, INNER JOIN, AND, OR, LIKE.

   • There must be PRIMARY KEY in all the tables of database with column name ID.

   • If "One Table" references " Another Table" than the column name used in reference should use the following rule: Column of Another Table: <OneTableName> ID

Example:

If User table references Employee table than the column name used in reference should be UserID where User is table name and ID primary column of User table and UserID is reference column of Employee table.

    • Columns with Default value constraint should not allow NULLs.

    • Practice using PRIMARY key in WHERE condition of UPDATE or DELETE statements as this will avoid error possibilities.

    • Always create stored procedure in same database where its relevant table exists otherwise it will reduce network performance.

    • Avoid server-side Cursors as much as possible, instead use SELECT statement. If you need to use cursor then replace it with WHILE loop.

   • Instead of using LOOP to insert data from Table B to Table A, try to use SELECT statement with INSERT statement.

      INSERT INTO Table A (column1, column2 )

      SELECT column1, column2

      FROM   Table B

      WHERE  ...

     • Avoid using spaces within the name of database objects; this may create issues with front-end data access tools and applications . If you need spaces in your database object name then will accessing it surround the database object name with square brackets.

     Example:

           [Order Details]

   • Do not use reserved words for naming database objects, as that can lead to some unpredictable situations.

   • Practice writing comments in stored procedures, triggers and SQL batches, whenever something is not very obvious, as it will not impact the performance.

   • Do not use wild card characters at the beginning of word while search using LIKE keyword as it results in Index scan.

   • While using JOINs in your SQL query always prefix column name with the table name.

   • Default constraint must be defined at the column level. All other constraints must be defined at the table level.

   • Avoid using rules of database objects instead use constraints.

   • Always put the DECLARE statements at the starting of the code in the stored procedure. This will make the query optimizer to reuse query plans.

   • Put the SET statements in beginning ( after DECLARE) before executing code in the stored procedure.

   • Use BEGIN ... END blocks only when multiple statements are present within a conditional code segment.

    • To express apostrophe within a string, nest single quotes (two single quotes).

     Example:

           SETsExample =   'SQL''s Authority '

   • When working with branch conditions or complicated expressions, use parenthesis to increase readability.

    Example:

     IF ((SELECT 1

                FROM TableName

                WHERE 1 = 2) ISNULL)

   • To mark single line as comment use (--) before statement. To mark section of code as comment use (/\*...\*/).

  • Avoid the use of cross joins if possible.

   • If there is no need of resultset then use syntax that does not return a resultset.

           IF EXISTS(SELECT 1

                 FROM UserDetails

                 WHERE UserID = 50)

      IF EXISTS(SELECT COUNT(UserID)

               FROM UserDetails

               WHERE UserID = 50)

   • Use graphical execution plan in Query Analyzer or SHOWPLAN\_TEXT or SHOWPLAN\_ALL commands to analyze SQL queries. Your queries should do an "Index Seek "instead of an" Index Scan "or a" Table Scan ".

   • Do not prefix stored procedure names with "SP\_", as "SP\_" is reserved for system stored procedures.

    Example:

       SP <App Name> \_ [< Group Name> \_] <Action> <table / logical instance>

   • Do not query / manipulate the data directly in your front end application, instead create stored procedures, and let your applications to access stored procedure.

   • Avoid using ntext, text, and image data types in new development work. the Use nvarchar The (max) , A varchar (max) , and a varbinary (max) instead.

   • Do not store binary or image files (Binary Large Objects or BLOBs) inside the database.

   • Use the CHAR datatype for a non -nullable column, as it will be the fixed length column, NULL value will also block the defined bytes.

   • Avoid using dynamic SQL statements. Dynamic SQL tends to be slower than static SQL, as SQL Server generate execution plan every time at runtime.

   • Minimize the use of Nulls. Because they incur more complexity in queries and updates.   ISNULL and COALESCE functions are helpful in dealing with NULL values

   • Use Unicode datatypes, like NCHAR, NVARCHAR or NTEXT if it needed, as they use twice as much space as non-Unicode datatypes.

   • Always use column list in INSERT statements of SQL queries. This will avoid problem when table structure changes.

   • Perform all referential integrity checks and data validations using constraints instead of triggers, as they are faster. Limit the use of triggers only for auditing, custom tasks, and validations that can not be performed using constraints.

   • Always access tables in the same order in all stored procedure and triggers consistently. This will avoid deadlocks.

   • With Begin and End Transaction always use global variable@@ERROR, immediately after data manipulation statements (INSERT / UPDATE / DELETE), so that if there is an Error the transaction can be rollback.

   • Do not use column numbers in the ORDER BY clause; it will reduce the readability of SQL query.

Example:

      SELECT     UserID ,

                         UserName ,

                         Password

      FROM        UserDetails

      ORDER BY UserName;

   • The RETURN statement is meant for returning the execution status only, but not data. If you need to return data, use OUTPUT parameters.

   • Effective indexes are one of the best ways to improve performance in a database application.

   • BULK INSERT command helps to import a data file into a database table or view in a user-specified format.

   • Use MERGE Statement to implement multiple DML operations instead of writing separate INSERT, UPDATE, DELETE statements.

   • When some particular records are retrieved frequently , apply Filtered Index to improve query performace, faster retrieval and reduce index maintenance costs.

   • Using the NOLOCK query optimizer hint is considered good practice in order to improve concurrency on a busy system.