1. What is Dynamic SQL?

Dynamic SQL, is a batch of SQL statement(s) that is generated within T-SQL and executed using the `EXECUTE` (or `exec`) statement or, preferably, via the `sp_executesql` system stored procedure.

2. Analysis – Pros and Cons of Dynamic SQL

The following section provides the pros and cons of dynamic SQL when compared with the Static SQL statements.

2.1 Benefits of Dynamic SQL -

**Predicate Optimization**: The real benefit of dynamic SQL is that the execution plans generated for each invocation of the query will be optimized for only the predicates that are actually being used at that moment. The main issue with the static SQL solutions, aside from maintainability, was that the additional predicates confused the query optimizer, causing it to create inefficient plans. Dynamic SQL gets around this issue by not including anything extra in the query.

**Single Query Plan Caching**: For every stored procedure there is one cached query plan and an additional ad hoc plan cached for each invocation of the stored procedure. (This can be verified using the view `sys.dm_exec_cached_plans`) This means that every time a new argument is passed to the stored procedure, a compilation occurs, which is clearly going to kill performance. The dynamic query is not being parameterized and is therefore producing duplicate query plans for different arguments.
2. 2 Drawbacks of Dynamic SQL -

**Speed**: Dynamic SQL tends to be slower than static SQL, as SQL Server must generate an execution plan every time at runtime. Alternative, ‘IF’ and ‘CASE’ statements come in handy to avoid dynamic SQL.

**Permissions**: Dynamic SQL requires the users to have direct access permissions on all accessed objects like tables and views. Generally, users are given access to the stored procedures which reference the tables, but not directly on the tables. In this case, dynamic SQL will not work.

**Syntax**: One distinct advantage of writing stored T-SQL procedures is that you get a syntax check directly. With dynamic SQL, a trivial syntax error may not show up until run time. Even if you test your code carefully, there may be some query, or some variation of a query, that is only run in odd cases and not covered in your test suite.

3. Best Practices while using Dynamic SQL

Using sp_ExecuteSQL as compared to Execute (or Exec)

1. Query parameterization
   a. With Exec statements the query cannot be parameterized effectively.
      Examples:

      ```
      DECLARE @String NVARCHAR(60)
      SET @String = 'Stone'
      DECLARE @sql NVARCHAR(MAX)
      SET @sql = '' +
      'SELECT AddressId ' +
      'FROM Person.Address ' +
      'WHERE AddressLine1 LIKE ''%'' + @String + ''%'''
      
      EXEC sp_executesql
      @sql,
      N'@String NVARCHAR(60)',
      @String
      ```

      If this is run using Execute (or exec) statement

      Msg 137, Level 15, State 2, Line 1
      Must declare the scalar variable "@String".
b. Ease of passing Multiple and Optional parameters. For multiple parameters, simply comma-delimit their data type definitions in the second parameter, and then pass as many outer parameters as necessary to define every variable listed in the second parameter.

2. Prevention of SQL injection

4. References

- Great Debate: Store Procedure vs. Dynamic SQL
- The Curse and Blessings of Dynamic SQL
  [http://www.sommarskog.se/dynamic_sql.htm](http://www.sommarskog.se/dynamic_sql.htm)
- Coding Conventions
  [http://vyaskn.tripod.com/coding_conventions.htm](http://vyaskn.tripod.com/coding_conventions.htm)