T-SQL Coding Convention & Architecture Guidelines

This information was drawn from Casper Nielsen's T-SQL Coding Standard, Vyas Kondreddi's TSQL Coding Conventions article, Brian Walker's Coding Standard document , and my own experience.

Coding

1. Avoid using "SELECT *"
2. Always use ORDER BY statement when row order matters. Don't depend on the natural data order in the table.
3. Always qualify field names with table name aliases when there are multiple tables involved in a T-SQL statement.

```
SELECT
 Table1.Column1,
 Table2.Column10,
 Table2.Column4
FROM
 Table1
 JOIN Table2 ON Table1.Column1 = Table2.Column1
```

4. Avoid using SQL keywords as object or column names. Ex: don't use "DateTime" for a column name.
5. Never round numeric values in the database. Rounding should be handled by the user interface.
6. Align related numbers into columns when they appear in contiguous lines of code, such as with a series of SUBSTRING function calls. This makes the list of numbers easier to scan.
7. Always specify the length of a character data type, and make sure to allow for the maximum number of characters users are likely to need, since characters beyond the maximum length are simply lost.
8. Always specify the precision and scale of the decimal data type, which otherwise defaults to an unspecified precision and integer scale.
9. Prefer try-catch error handling over checking @@error.
10. Avoid using "undocumented" features such as undocumented columns in system tables, undocumented functionality in T-SQL statements, or undocumented system or extended stored procedures.
11. Do not rely upon any implicit data type conversions. For example, don't assign a character value to a numeric variable assuming that T-SQL will do the necessary conversion. Instead, use the appropriate CONVERT function to make the data types match before doing a variable assignment or a value comparison. Another example: Although T-SQL does an implicit and automatic RTRIM on character expressions before doing a comparison, don't depend on that behavior because compatibility level settings and use of nchar complicate things.
12. Do not directly compare a variable value to NULL with comparison operators (symbols). If the variable could be null, use IS NULL or IS NOT NULL to do a comparison.
13. Don't use the STR function to perform any rounding; use it with integers only. Use the CONVERT function (going to a different scale) or the ROUND function before converting to a string if the string form of a decimal value is needed. CEILING and FLOOR are other options.
14. Be careful with mathematical formulas, since T-SQL may force an expression into an unintended data type. Add point zero (.0) to integer constants if a decimal result is desired.
15. Avoid using double quotes in T-SQL code. Use single quotes for string constants. If it's necessary to qualify an object name, use (non-ANSI SQL standard) brackets around the name.
16. Avoid the creation of temporary tables while processing data, as creating a temporary table means more disk I/O. Consider using advanced SQL, views, table variables, or derived tables, instead of temporary tables. For table variables: if the table variable will contain a large set of data, be aware that indexing is very limited (primary key index only).
17. Create temporary tables early in the routine, and explicitly drop them at the end. Interspersed DDL and DML statements can contribute to excessive recompile activity.
18. Be cautious with table-valued UDFs, because passing in a parameter with a variable, rather than a constant, can result in table scans if the parameter is used in a WHERE clause.
19. Avoid using the same table-valued UDF more than once in a single query. Table-valued UDFs do, however, have some dynamic compilation features that can be handy.
20. Almost every stored procedure should have SET NOCOUNT ON near the beginning, and it's good form to have SET NOCOUNT OFF near the end. This standard also applies to triggers.
21. Consider declaring an explicit transaction any time more than one database modification statement is used in a routine. This includes a single statement executed multiple times in a loop.
22. Always look for a set-based solution to a problem before implementing a cursor-based approach or a temporary table approach. A set-based approach is usually more efficient. If a cursor is unavoidable, consider using a WHILE loop instead.
23. All databases should have a table which stores a sequence of numbers. Our convention is a table named "Tally" with a single column named "N". This should be populated with numbers from 0 to 10,000 at least.
24. Understand how a CROSS JOIN works, and take advantage of it. For example, you can use a CROSS JOIN effectively between a table of working data and a table of sequence numbers; the result set contains a record for every combination of working data and sequence number.
25. Avoid wild card characters at the beginning of a word while searching using the LIKE keyword, as that results in an index scan, which defeats the purpose of an index. The following statement results in an index scan, while the second statement results in an index seek:

```
SELECT LocationID FROM Locations WHERE Specialties LIKE '%pples'
SELECT LocationID FROM Locations WHERE Specialties LIKE 'A%s'
```

26. Avoid searching using not equals operators (<> and NOT) as they result in table and index scans.
27. Avoid the use of GOTO. They are basically not needed after the introduction of TRY/CATCH and often indicate sloppy coding.
28. Avoid using batches (scripts) for recurring tasks, except creating and initializing the database. Use SSIS for transferring batches of data.
29. Always individually name fields in INSERT and UPDATE statements. Never use the * operator in such statements.

```
--Correct
INSERT INTO dbo.[User] (FirstName, LastName)
VALUES (@FirstName, @LastName)

--Avoid
INSERT INTO dbo.[User]
VALUES (@FirstName, @LastName)
```

30. Avoid using dynamic SQL in Procedures.
31. When using transactions, always set XACT_ABORT. If you have more than one database modifying action in a Procedure, decide if it needs to run atomically (most probably it does) - if so set XACT_ABORT ON and encapsulate all statements in a transaction. If it is not required to run atomically, explicitly set XACT_ABORT OFF.

```
--Correct
CREATE PROCEDURE dbo.uspMyProcedure (...)
AS
BEGIN
 SET XACT_ABORT ON
 BEGIN TRAN
 (...) --Transaction will be rolled back if anything fails here
 COMMIT TRAN
```

32. Terminate all SQL statements with a semicolon. This will be mandatory in future versions of SQL Server and is also required in other database systems.
33. Whenever reasonable, use ANSI Standard SQL (for example, use COALESCE instead of ISNULL).
34. When performing a division operation, make sure the denominator uses the NULLIF function if there is a possibility the denominator could hold a zero value, to prevent division-by-zero errors.

Database Design and Architecture

1. Always have unique indexes or primary keys on all tables without exception - never leave the integrity up to the application.
2. If you have enumerations in another layer that are persisted onto the database, map these in a table and ensure consistency by constraints.
3. If you allow NULL in a field, make sure it has a meaning apart from empty and zero.
4. Never use the TEXT or NTEXT types. Instead, use the MAX types.

```
--Correct
CREATE TABLE dbo.MyTable
(
 MyTextField varchar(MAX),
 MyUnicodeTextField nvarchar(MAX)
)

--Avoid
CREATE TABLE dbo.MyTable
(
 MyTextField text,
 MyUnicodeTextField ntext
)
```

5. Be cautious when using binary types (binary, varbinary, image, etc.) for storing files. Weigh the structure benefit with the added database load before deciding which approach to use.
   Examine where the file logically belongs: If it have significance to the database, store it there, otherwise store it on the file system.
   If you choose the file system as the data store for files, remember to backup the file store simultaneously with the database to counter synchronization issues.

6. Implement the database so it will keep itself consistent with the model it was designed for. This may involve using referential integrity checks extensively.
7. Avoid triggers if possible. Using Procedures as access points should decrease the need for triggers considerably.
8. Be careful when using collations - they handle differently. Find the most general collation and use this as the database default. For most European language scenarios this will be the SQL_Latin1_General_CP1_CI_AS (Case insensitive/Accent sensitive). When the need arises to use a specialized collation, do so on the specific field.