2021

SQL Server Coding Standard

ATENA VAHEDIAN (TEAM LEAD)
ADEYEMI OWOSENI
ASHLEY HUSCROFT
LEYSAN GILFANOVA
MAHMOUDREZA KARIMKHANINEJAD
MOHAMMAD MOSIHUZZAMAN

Table of Contents

1. Purpose of the Document

The purpose of this document is to define the coding standard to be used by all database developers working on the We Vote project.

2. Naming Convention

2.1. General Naming Rule

Use a unique name for SQL Server objects and avoid using SQL reserved keywords (ABSOLUTE, AFTER, BETWEEN, CACHE, DAY, EQUALS etc.) in the object names.
Avoid using spaces or any other invalid characters in object names; this may create issues in the code.
Note: Square brackets can be used for object names that include spaces, however, it may cause difficulties when reading the code, therefore it is not recommended.

Example:
Voting Station Type

Example:
VotingStationType

2.2. Naming Columns and Tables

Use "Pascal" notation to name objects such as Tables and their attributes. Use a singular noun for table names and plural for column names.

Example:
CandidateName
CandidateIDs

2.3. Naming Reference Columns

Include prefix PK_ and FK_ to indicate columns used as reference in the relational database. If the same column is used as foreign key in multiple tables, use numbers to distinguish each one.

Example:
PK_VotingStationIDs
FK_WardIDs
FK2_CommunityIDs

2.4. Naming Join Tables

Use "Pascal" notation to name objects such as Join Tables. The name should follow this structure: <table name>_Result when joining multiple normalized tables.

Example:
RegularVotingStation_Result

2.5. Naming Views

Use "Pascal" notation to name objects such as Views. The name should follow this structure: <view name>_vw when creating views.

Example:
MayorResults2017_vw

2.6. Naming Stored Procedure

Use "Pascal" notation to name objects such as Stored Procedures. The name should follow this structure: sp<action type> where action is: Get, Delete, Update, Write, Archive, Insert.

Example:
spGetPrecentage

2.7. Naming Indexes

Use the following naming convention for Indexes: IX_<tablename>_<volume name>.
Note: Clustered indexes created by Microsoft SQL Server on primary keys suffice for the purpose of this database.

Example:
IX_Community_CommunityNames

2.8. Naming Primary Key

Use the following naming convention for Primary Keys: PK_<column name>

Example:
PK_CommunityID
PK_ WardID

2.9. Naming Foreign Key

Use the following naming convention for Foreign Keys: FK_<column name>
Note: If the same column is used as a foreign key in multiple tables, use numbers to distinguish each one.

Example:
FK_VotingStationID
FK2_CommunityIDs

3. Coding Style

3.1. Comments

For clear readability, enclose comments in "--" for single line comments or "/* *comments*/ for descriptive comments.

Example:

-- This whole line is a comment --

```
/*
 All the text in this paragraph will be treated as comments
by SQL Server.
*/
```

3.2. SQL Code

Use “UPPER CASE” for all reserved SQL keywords and commands. Follow naming convention to call tables, columns, views, stored procedures. Always put semicolon (;) as an indication of the end of a statement. In addition, place each attribute name on a separate line when there are multiple attributes listed in the statement.

Example:

```
SELECT CommunityName,
        CommunityPopulation
FROM Communities;
```

3.3. Data Retrieval

Avoid use of “SELECT * ” in SQL queries. Instead use SELECT statement with a list of columns to retrieve FROM required table.
*Note: “SELECT *” should only be used when there is a need to specifically view all the columns in the table like to ensure all columns have been addressed, otherwise, refrain from using the statement.*

Example:

```
SELECT *
FROM CommunityInformation;
```

Example:

```
SELECT CommunityName,
        CommunityPopulation
FROM CommunityInformation;
```

3.4. Common Table Expression

Use CTE (Common Table Expression) to define a temporarily named result set.

Example:

```
    WITH cte_Ward AS (
       SELECT PK_WardIDs
          COUNT(*) Duplicates
     FROM Ward
     WHERE VotingStationIDs = 20);
```

3.5. Indentation

To improve readability of SQL Syntax, use proper indentation where necessary. Use a new line for each separate query. For queries with multiple columns, place each column on a new line. Surround the equals " = " operator with spaces.

Example: Wrong Format

```
UPDATE  VotingStation SET VotingStationName = 'Elboya School' WHERE VotingStationID = 3304;
```

Example: Correct Format

```
UPDATE  VotingStation
    SET VotingStationName = 'Elboya School'
WHERE VotingStationID = 3304;
```

3.6. Conditions

Practice using WHERE condition with UPDATE or DELETE statements to avoid accidental changes of the dataset.

Example:

```
INSERT INTO CommunityInformation (CommunityNames, CommunityPopupalions)
        SELECT CommunityNames,
                CommunityPopupalions
FROM MasterData
WHERE CommynityNames = 'Acadia';
```

3.7. ANSI JOINs

When joining tables in SQL, always use a column name along with the abbreviation of the table name.

Example:

```
SELECT C.CommunityNames,
            HO.NumberHomeOwners,
            HO.NumberRenters
FROM Community AS C
RIGHT JOIN HomeOwner AS HO
ON C.CommunityID = HO.CommunityID;
```

3.8. Parenthesis

Use parentheses with branch conditions and complicated expressions to improve code readability.

Example:

```
SELECT CommunityID,
        NumberCitizen
IF (NumberCitizen>5000, “SATISFIED”, “NOT SATISFIED”)
FROM Citizenship;
```

3.9. New Values

List column names in the INSERT statements of SQL queries when populating tables, to ensure data is entered accurately.

Example:

```
INSERT INTO PoliticalScale (PK_PoliticalIDs,
                            PoliticalLevels
VALUES (1, 'Left');
```

3.10. Sorting

Use ORDER BY followed by ASC/DESC to sort records in ascending or descending order.

Example:

```
SELECT CommunityIDs,
        NumberHomeowners
FROM HomeOwner
ORDER BY NumberHomeowners  ASC;
```

3.11. Indexes

Clustered or non-clustered indexes are one of the best ways to improve performance in a database application.
Note: Clustered indexes created by Microsoft SQL Server on primary keys suffice the purpose of this database.

3.12. Data Importing

BULK INSERT command imports a data file into a database table or view in a user specified format.
Example: .csv file

4. Security

4.1. Use Referential Integrity to avoid accidental loss of tables or mistakes in data manipulations. Place BEGIN TRAN before a SQL statement. If an execution is done correctly, use COMMIT command to complete the query, otherwise use ROLLBACK to return to the original result.
Note: Always use BEGIN TRAN, COMMIT/ROLLBACK with UPDATE, DELETE, DROP, TRUNCATE.

Example:

```
BEGIN TRAN                                  - doesn't save a procedure in a database
DELETE FROM VotingStation
WHERE VotingStationNames = 'Haysboro School'
COMMIT;                                     - complete a procedure
ROLLBACK;                                   - discard changes
```

4.2. Avoid deadlocks, by accessing tables in the same order in all stored procedures and triggers consistently. A deadlock occurs when two (or more) transactions block each other by holding locks on the resources each of the transaction's needs.

5. Data Types and NULL Values

5.1. Use appropriate data types for the column value to allow for efficient storage. Avoid using maximum length where it is unnecessary.
Example: VARCHAR (max), NVARCHAR (max) can be replaced with VARCHAR (appropriate size), NVARCHAR(appropriate size).

5.2. Consider using TINYINT and SMALLINT for Primary Keys and Numeric data types without decimals.

5.3. Use VARCHAR for English textual data type. A Unicode character NVARCHAR can be used for other languages or when special characters are present.

5.4. Allow for NULL values in columns only when NULL values are required. Minimize the use of NULLs.

5.5. CHAR, VARCHAR, NCHAR, NVARCHAR, TEXT, NTEXT allow the use of an empty string as a default. NULL values are not required.

6. Database Design Basis

6.1. Microsoft SQL Server is the relational database management system used in this project to process big datasets.

6.2. Database Normalization needs to be applied to exclude redundant data and to efficiently store data. Third Normal Form (3NF) will satisfy the database functionality and improve performance.

6.3. Consider using stored procedures for repetitive commands and calculation.