T-SQL static code analysis: Two branches in a conditional structure should not have exactly the same implementation

2 minutes

Having two branches in an `IF/ELSE IF` chain with the same implementation is at best duplicate code, and at worst a coding error.

If the same logic is truly needed for both instances, then in an `IF` chain they should be combined.

Noncompliant Code Example

```
IF @SortOrder = 1
  BEGIN
    SET @SortOrder = 0
    SELECT LastName FROM Employees ORDER BY
LastName
  END
ELSE IF @SortOrder = 2
  BEGIN
    SET @SortOrder = 0
    SELECT LastName FROM Employees ORDER BY
LastName -- Noncompliant
  END
```

```
ELSE
  BEGIN
    SET @SortOrder = -1
    SELECT LastName FROM Employees
  END
GO
```

Exceptions

Branches in an `IF/ELSE IF` chain with implementation that contains a single line of code are ignored.

```
IF @SortOrder = 1
  BEGIN
    SELECT LastName FROM Employees ORDER BY
LastName
  END
ELSE IF @SortOrder = 2
  BEGIN
    SELECT LastName FROM Employees
  END
ELSE
  BEGIN
    SELECT LastName FROM Employees ORDER BY
LastName -- No issue, usually this is done on purpose to
increase the readability
  END
GO
```

But this exception does not apply to `IF` chains without ELSE-s when all branches have the same single line of code. In case of `IF` chains with ELSE-s rule {rule:tsql:S3923} raises a bug.

```
IF @SortOrder = 1    -- Noncompliant, this might have been
```

```
done on purpose but probably not
  BEGIN
    SELECT LastName FROM Employees ORDER BY
LastName
  END
ELSE IF @SortOrder = 2
  BEGIN
    SELECT LastName FROM Employees ORDER BY
LastName
  END
GO
```