You've encountered a common behavior when using Contains with a list of values in Entity Framework (EF) against SQL Server. Here's a breakdown of why EF generates a query with OPENJSON and how it works:

Why OPENJSON is Used:

1. **Parameterization Limitations:** SQL Server doesn't directly support passing a list of values as a single parameter for use in an IN clause. EF needs to find a way to represent your listFilters (which is a C# collection) as a set of values that SQL Server can understand.
2. **OPENJSON for JSON Parsing:** SQL Server's OPENJSON function allows you to parse a JSON string and extract values from it. EF leverages this to convert your C# list into a JSON array, which can then be used in the IN clause.

How it Works:

1. **C# List to JSON:** EF takes your listFilters (e.g., List<int>) and serializes it into a JSON array string. For example, if listFilters contains [1, 2, 3], it's converted to the JSON string "[1,2,3]".
2. **Parameterization:** This JSON string is passed to SQL Server as a parameterized value (in your example, @__listFilters_0). This is important for security and performance.
3. **OPENJSON Parsing:** The SQL query uses OPENJSON(@__listFilters_0) to parse the JSON string.
4. **WITH ([value] int '$') AS [l]:** This part specifies how to extract the values from the JSON.
    a. WITH ([value] int '$') defines a schema for the parsed JSON. It indicates that you want to extract a column named value of type int from the root of each JSON element (indicated by $).
    b. AS [l] assigns an alias l to the resulting table from OPENJSON.
5. **IN Clause:** The WHERE [p].[RandomNumber] IN (SELECT [l].[value] FROM ...) clause filters the Posts table based on whether RandomNumber matches any of the parsed values from the JSON array.

In essence, EF is doing a workaround to translate your C# Contains operation into a SQL query that SQL Server can efficiently execute.

Advantages of this Approach:

- **Parameterization:** Prevents SQL injection vulnerabilities.
- **Performance:** SQL Server's OPENJSON is generally efficient for parsing JSON.

Potential Considerations:

- **SQL Server Version:** OPENJSON is available in SQL Server 2016 and later. If you're using an older version, EF might use a different approach (which could be less efficient).
- **Performance with Large Lists:** If your listFilters contains a very large number of values, the JSON string could become large, and the OPENJSON parsing might take longer. In such cases, consider alternative approaches, such as table-valued parameters (TVPs), if they are applicable.
- **Type Compatibility:** Ensure that the types in your C# list and the corresponding column in the SQL Server table are compatible. EF will attempt to convert them, but type mismatches can lead to errors.

In summary, the OPENJSON behavior is a mechanism EF uses to bridge the gap between C# collection operations and SQL Server's query capabilities, allowing you to use Contains with a list of values effectively.