Exploring The New {JSON} Features In SQL Server

Who is this guy?

Eric Cobb

Database Development Manager

MCSE: Data Platform | MCSE: Data Management and Analytics

1999-2013: “Webmaster“, Programmer, Developer

2013+: SQL Server Database Administrator

GitHub: https://github.com/ericcobb

Blog: http://www.sqlnuggets.com

Twitter: @sqlnugg
@cfgears

What Tools Are We Using?

- SQL Server
  - 2017 Developer Edition (Free Download)
- Stack Overflow Database
  - 10 GB and 100+ GB versions (Free Download)
- SQL Operations Studio
  - Windows, macOS, and Linux (Free Download)

What Are We Learning?

- How to easily return JSON from your existing queries
- How to parse JSON text and find or extract specific objects
- How to parse JSON text and turn the data into rows and columns
- How to store and retrieve JSON documents in SQL Server

Returning JSON From T-SQL Queries

- Add the FOR JSON clause to a SELECT statement to format query results as JSON
  - Use FOR JSON AUTO to automatically format the JSON output based on the structure of the SELECT statement
  - Use FOR JSON PATH to maintain full control over the format of the JSON output

STOP: {DEMO TIME}

RETURNING JSON FROM T-SQL QUERIES

Using SQL Server’s JSON Functions

Built-in JSON Functions

- ISJSON tests whether a string is valid JSON
- JSON_VALUE extracts a scalar value from a JSON string
- JSON_QUERY extracts an object or an array from a JSON string
- JSON_MODIFY changes a value in a JSON string
- OPENJSON parses JSON text and returns rows and columns

JSON Path Expressions

- Use JSON path expressions to reference specific records in JSON objects
  - Functions seek into the JSON text at the specified position and parse only the referenced fragment
- Path is specified via a set of path steps:
  - The default value for path is ‘$’, which represents the context root
  - Key names - for example, $.user.firstname
  - Array elements - for example, $.user[1]
  - The dot operator – for example, $.user[1].firstname
- A path expression has an optional path mode, with a value of **lax** (default) or **strict**

JSON_VALUE & JSON_QUERY

- **JSON_VALUE** - Returns a single text value of type NVARCHAR(4000)
  - If the value is greater than 4000 characters:
    - In lax mode, JSON_VALUE returns null
    - In strict mode, JSON_VALUE returns an error
  - If you have to return values greater than 4000 characters, use OPENJSON instead of JSON_VALUE
- **JSON_QUERY** - Returns a JSON fragment of type NVARCHAR(MAX).
  - If the returned value is not an object or an array:
    - In lax mode, JSON_QUERY returns null
    - In strict mode, JSON_QUERY returns an error

OPENJSON

- Table-valued function used in FROM clause
- Only available under compatibility level 130 (SQL 2016) or higher
    - If compatibility level is lower than 130, SQL Server can't run OPENJSON
    - Other JSON functions are available at all compatibility levels, provided you are on SQL Server 2016 or higher

STOP: {DEMO TIME}

USING JSON FUNCTIONS

Storing JSON Data In SQL Server

JSON Data Types In SQL Server

- There are no JSON Data Types in SQL Server!

- JSON is stored as an NVARCHAR
  - Uses native JSON functions to parse JSON documents using T-SQL
  - NVARCHAR(MAX) lets you store JSON documents that are up to 2 GB in size
    - Recommend that you use NVARCHAR(4000) or below for performance reasons

STOP: {DEMO TIME}

STORING JSON DATA

Eric Cobb

GitHub: https://github.com/ericcobb

Blog: http://www.sqlnuggets.com

Twitters: @cfgears, @sqlnugg