2.7 JAVASCRIPT OBJECT NOTATION (JSON)

JSON is a text-based data exchange format derived from JavaScript that is used in web services and other connected applications.

JSON Syntax

JSON defines only two data structures: objects and arrays. An object is a set of name-value pairs, and an array is a list of values. JSON defines seven value types: string, number, object, array, true, false, and null**.**

- Objects are enclosed in braces ({}), their name-value pairs are separated by a comma (,), and the name and value in a pair are separated by a colon (:). Names in an object are strings, whereas values may be of any of the seven value types, including another object or an array.

- Arrays are enclosed in brackets ([]), and their values are separated by a comma (,). Each value in an array may be of a different type, including another array or an object.When objects and arrays contain other objects or arrays, the data has a tree-like structure.

Uses of JSON

JSON is often used as a common format to serialize and deserialize data in applications that communicate with each other over the Internet. These applications are created using different programming languages and run in very different environments. JSON is suited to this scenario because it is an open standard, it is easy to read and write, and it is more compact than other representations.

JSON is built on two structures:

A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence

JSON Syntax

JSON syntax is considered as a subset of JavaScript syntax:

- Data is in name/value pairs: JSON data is written as name/value pairs. A name/value pair consists of a field name (in double quotes), followed by a colon.

  Example: “name”:”Cat”

- Data is separated by commas: **Example:** "first_name" : "Sun", "last_name" : "moon",

- Curly braces hold objects. Square brackets hold arrays. JSON keys are on the left side of the colon.They need to be wrapped in double quotation marks,

JSON Values:

In JSON, values must be one of the following data types:string, number, object (JSON object), array, Boolean and null

```
{ "firstName": "J", "lastName": "S", "age": 25, "children": [], "spouse": null,
 "address": {"street": "7504 TVS nagar", "city": "Tambaram","state": "Tamilnadu", "postalCode": "603203" },
 "phoneNumbers": [ {"type": "mobile", "number": "212 555-3346"},
```

```
{ "type": "fax", "number": "646 555-4567" } ]}
```

The first two name value pairs maps a string to another string. The third name value pair maps a string age with a number 25. The fourth pair maps a string children with an empty array []. The fifth pair maps a string spouse with null value. The sixth pair maps a string address with another JSON object. The seventh pair maps a string with array of JSON objects.

Function Files

There is no native support for defining functions in JSON. Commonly used approach is to define function as string and use eval() or new Function() to construct the function. The basic difference between these two are:

- eval() works within the current execution scope. It can access or modify local variables.
- new Function() runs in a separate scope. It cannot access or modify local variables.

HTTP Requests

JSON is most commonly used in asynchronous HTTP requests. This is where an application pulls data from another application via an HTTP request on the web.XMLHttpRequest is an API that provides scripted client functionality for transferring data between a client and a server. It enables to get data from an external URL without having to refresh the page. For example, a user could click a button that results in a small part of the page updating, rather than the whole page.

Artist.txt

```
{ "artists" : [
  { "artistname" : "Leonard Cohen", "born" : "1934"      },
  { "artistname" : "Joe Satriani", "born" : "1956" },
  { "artistname" : "Snoop Dogg","born" : "1971" } ]}
```

Below is a sample HTML page that retrieves that JSON data via HTTP, and uses JavaScript to wrap it in HTML tags and output it to the HTML document.

```
<!doctype html>
<title>Example</title><script>
// Store XMLHttpRequest and the JSON file location in variables
varxhr = new XMLHttpRequest();
varurl  =  "https://www.abc.com/json/Artists.txt";
```

```
// Called whenever the readyState attribute changes
xhr.onreadystatechange = function() {
  if (xhr.readyState == 4 &&xhr.status == 200) {  // Check if fetch request is done
varjsonData = JSON.parse(xhr.responseText); // Parse the JSON string
    showArtists(jsonData);// Call the showArtists(), passing in the parsed JSON string
  }};
// Do the HTTP call using the url variable we specified above
xhr.open("GET", url, true);
xhr.send();
// Function that formats the string with HTML tags, then outputs the result
function showArtists(data) {
var output = "<ul>"; // Open list
var i;    // Loop through the artists, and add them as list items
for (var i in data.artists) {
      output += "<li>" + data.artists[i].artistname + " (Born: " + data.artists[i].born +
")</li>"; }
output += "</ul>"; // Close list. Output the data to the "artistlist" element
document.getElementById("artistList").innerHTML = output;
}</script><!-- The output appears here -->
<div id="artistList"></div>
```

JSON-SQL

JSON functions in SQL Server enable to analyze and query JSON data, transform JSON to relational format, and export SQL query results as JSON text.

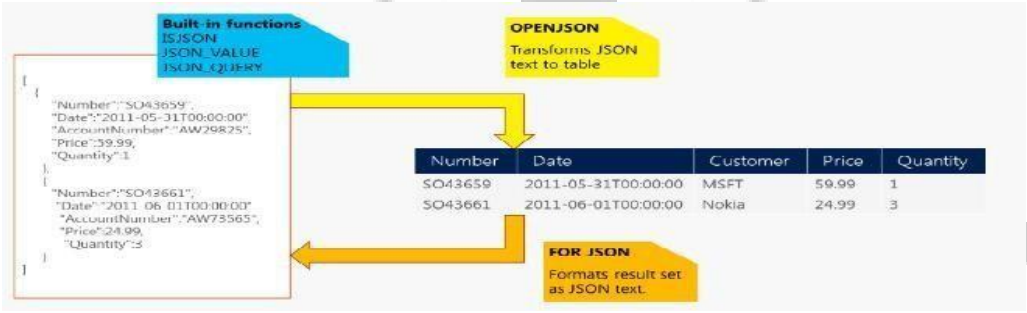

Fig 2.5: JSON with SQL

JSON text can be extracted from JSON or verify that JSON is properly formatted using built-in functions JSON_VALUE, JSON_QUERY, and ISJSON. For more advanced querying and analysis, the OPENJSON function can transform an array of JSON objects into a set of rows. Any SQL query can be executed on the returned result set. Finally, there is the FOR JSON clause that enables to format query results as JSON text.

Transact-SQL code, we will define a text variable to put JSON text:

```
DECLARE @json NVARCHAR(4000)
SET @json = N'{  “info”:{  “type”:1,
   “address”:{ “town”:”Bristol”,“county”:”Avon”, “country”:”England” },
   “tags”:[“Sport”, “Water polo”] },
  “type”:”Basic”}’
```

Extract values and objects from JSON text using the JSON_VALUE and JSON_QUERY functions:

```
SELECT
  JSON_VALUE(@json, ‘$.type’) as type,
  JSON_VALUE(@json, ‘$.info.address.town’) as town,
  JSON_QUERY(@json, ‘$.info.tags’) as tags
```

This query will return “Basic”, “Bristol”, and [“Sport”, “Water polo”] values. The JSON_VALUE function returns one scalar value from JSON text (e.g. strings, numbers, true/false) that is placed on a JSON path specified as the second parameter. JSON_QUERY returns an object or array on the JSON path. JSON built-in functions use JavaScript-like syntax to reference values and objects in JSON text via second parameter.The OPENJSON function enables to reference some array in JSON text and return elements from that array:

```
SELECT valueFROM OPENJSON(@json, ‘$.info.tags’)
```

The string values from the tags array are returned. However, the OPENJSON function can return any complex object.Finally, there is a FOR JSON clause that can format any result set returned by SQL query as JSON text:

```
SELECT object_id, nameFROM sys.tablesFOR JSON PATH
```