SQL for NoSQL Databases: Déjà Vu (Part 2)

Christoph Bussler
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065

Christoph.Bussler@oracle.com

1. THE ALTERNATIVE APPROACH TO QUERYING JSON WITH SQL: JSON SQL

Providing a query interface for querying "schema-less" JSON documents using SQL as declarative query language [1] is finally gaining momentum (e.g. [4], [5]).

The approach proposed in [2] called JSON SQL, however, represents an alternative approach as it adheres to the following principles:

- extends the SQL 92 grammar ([3]) without changing the existing grammar rules, but by strictly adding new rules or extending existing rules
- supports a syntax for specifying query results as documents or tables (!) with the results as documents precisely matching the stored documents' structure
- defines a clear semantics based on the relational model
- implements the complete set of JSON data types, including JSON true, JSON false and JSON null

In JSON SQL no assumption is made about the JSON document structure except that it must comply with the JSON standard. Documents are grouped into collections (which imply neither a specific semantics nor specific constraints).

2. EXAMPLES OF JSON SQL

A few examples are shown and explained next with more detail to be found in [2]:

`select {a, b} from exampleColl`
return a JSON document for each JSON document in exampleColl with properties "a" and/or "b" if present, if none is present, return the empty document

`select a, b from exampleColl`
return a table with two columns with the value of property "a" and/or the value of property "b" populated if present

`select {a AS a.b} from exampleColl`
project property "a" but relocate it to "a"."b" in the result document

`select {a AS a.b.c, b AS a.b} from exampleColl`
illegal as "a"."b" would remove "a"."b"."c" from the result

`select {*} from exampleColl where a.x.y = true`
return all documents completely where the value of property "a"."x"."y" is of literal JSON true

`select {*} from exampleColl where a = 'true'`
return all documents where the value of property "a" equals to String "true"

`select {*} from exampleColl where "true" < 'true'`
return all documents where the value of property "true" is less than the String "true".

`select * from exampleColl where a.b = c.[3]`
return all properties as columns where the value of property "a"."b" is equal to the 4th array element value of array "c" in the same document

`select {*} from exampleColl where a = [7, null]`
return all documents where the value of property "a" corresponds to the JSON array [7, null]

`select {c.[2]} from exampleColl`
project the 3rd array element; this requires a "filler" representation (<>) to return an array as e.g. [<>, <>, 5] indicating that the first and second array elements are unknown (since not projected). This is the only extension required for representing JSON values (or the absence of).

3. SEMANTICS

If a property or a path to a property does not exist, then the property will not participate in projection or selection.

Property value comparisons with values or JSON literals must have matching types (every JSON type has a literal representation, e.g., true, false, null, 'abc', {"a":5} and [0, 1, 'three', null]).

4. THE WAY FORWARD: KEEP GOING

The implementation outlined in [2] is not complete yet, but adheres to the principles outlined in Section 1. The implementation work continues and the step-wise progress will be reported in [2].

5. REFERENCES

[1] Bussler, C.: SQL for NoSQL Databases: Déjà Vu. 7th Biennial Conference on Innovative Data Systems Research (CIDR '15) January 4-7, 2015, Asilomar, California, USA.

[2] http://realprogrammer.wordpress.com

[3] http://savage.net.au/SQL/sql-92.bnf.html

[4] Microsoft Azure DocumentDB: https://azure.microsoft.com/en-us/documentation/articles/documentdb-sql-query/

[5] Couchbase: http://www.couchbase.com/n1ql

9th Biennial Conference on Innovative Data Systems Research (CIDR '17) January 8-11, 2017, Chaminade, California, USA.