
ORACLE®

Creating and Working with JSON in Oracle Database

Dan McGhan
Oracle Developer Advocate
JavaScript & HTML5
September, 2015

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

About me

- Dan McGhan
  - Oracle Developer Advocate
  - Focus on JavaScript and HTML5
- Contact Info
  - dan.mcghan@oracle.com
  - @dmcghan
  - jsao.io

Agenda

1. What is JSON?
2. Creating JSON
3. Working with JSON

Agenda

1. What is JSON?
2. Creating JSON
3. Working with JSON

What is JSON?

- JavaScript Object Notation
  - A simple data interchange format
  - Has its roots in JavaScript but is language independent
- Heavily used in JavaScript, both browsers and Node.js
  - Lighter weight and easier to use than XML

JSON's structure

- Based on 2 structures
  - object: {}
  - array: []
- Objects are made of name value pairs
- Values can be one of the following
  - string: "test"
  - number: 100
  - Boolean: true or false
  - structure: object or array
  - no value: null

One row from the departments table

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700

Same department in JSON

```
1  {
2    "DEPARTMENT_ID": 10,
3    "DEPARTMENT_NAME": "Administration",
4    "MANAGER_ID": 200,
5    "LOCATION_ID": 1700
6  }
```

Now with “cooler” keys

```
1  {
2    "id": 10,
3    "name": "Administration",
4    "managerId": 200,
5    "locationId": 1700
6  }
```

Several rows from the departments table

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Purchasing	114	1700

Those departments in JSON

```
[
  {
    "id": 10,
    "name": "Administration",
    "managerId": 200,
    "locationId": 1700
  },
  {
    "id": 20,
    "name": "Marketing",
    "managerId": 201,
    "locationId": 1800
  },
  {
    "id": 30,
    "name": "Purchasing",
    "managerId": 114,
    "locationId": 1700
  }
]
```

A row from department with related employees

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	20	Marketing	201	1800

	EMPLOYEE_ID	DEPARTMENT_ID	NAME	SALARY	HIRE_DATE
1	201	20	Michael Hartstein	13000	17-FEB-04
2	202	20	Pat Fay	6000	17-AUG-05

A department with nested employees in JSON

```json
{
  "id": 20,
  "name": "Marketing",
  "managerId": 201,
  "locationId": 1800,
  "employees": [
    {
      "id": 201,
      "name": "Michael Hartstein",
      "salary": 13000,
      "hireDate": "2004-02-17T00:00:00Z"
    },
    {
      "id": 202,
      "name": "Pat Fay",
      "salary": 6000,
      "hireDate": "2005-08-17T00:00:00Z"
    }
  ]
}
```

Other notes on JSON structure

- JSON is schemaless

```
1  [
2    "this is cool",
3    1,
4    [],
5    {}
6  ]
```

- There is no standard for handling dates
  - People often use epoch time

Agenda

1. What is JSON?
2. Creating JSON
3. Working with JSON

Options for creating JSON in Oracle

- Roll your own
- PL/JSON
- APEX_JSON
- ORDS
- Node.js

Our goal: departments with employees

```
{
  "id": 20,
  "name": "Marketing",
  "managerId": 201,
  "locationId": 1800,
  "employees": [
    {
      "id": 201,
      "name": "Michael Hartstein",
      "salary": 13000,
      "hireDate": "2004-02-17T00:00:00Z"
    },
    {
      "id": 202,
      "name": "Pat Fay",
      "salary": 6000,
      "hireDate": "2005-08-17T00:00:00Z"
    }
  ]
}
```

Roll your own

- JSON is easy, I can make it myself!
- Strings are not escaped
  - You'd have to write an escape function
- Limited use, can only *create* JSON
  - Want to write your own parser while you're at it? ☺

PL/JSON overview

- An open source library for working with JSON in Oracle
- Object based: JSON & JSON_LIST
  - Builds up an object that can be manipulated
- Many utility functions for all kinds of things
  - A little complicated

APEX_JSON overview

- A PL/SQL package aimed at helping APEX devs working with JSON
- Not object based
  - Writes to the htp buffer by default; can redirect to a clob
- Features for creating and working with JSON
  - Easy to learn and use
- Only available with APEX 5.0

ORDS overview

- Java based server designed to enable RESTful web services
  - Actually creates URI to serve JSON content
- Supports SQL to JSON (as do APEX_JSON and PL/JSON)
  - If SQL to JSON isn't sufficient, you can still use the other options with ORDS
- Can do much more
  - REST enable tables
  - APEX Listener
  - SODA

SQL/CURSOR to JSON limitations

- CURSOR expression control
  - Need a way to specify if results should be an array or an object
  - Always returning an array (like ORDS and APEX_JSON) is the next best thing
- Date handling
  - Formatting and time zone conversion support is lacking
- Boolean
  - Need a way to specify which values should be converted to Boolean
- Other things to consider
  - Binds
  - Pagination

Node.js overview

- Node.js allows one to run JavaScript on the server
  - Runs Google's V8 JavaScript engine (used in Chrome)
- Oracle has a database driver for Node.js
  - Release in January
  - Lot's of cool features in the works
- Queries to the driver return JavaScript objects
  - Very closely related to JSON

Agenda

1. What is JSON?
2. Creating JSON
3. Working with JSON

PL/JSON

- Robust capabilities for working with JSON

```
if(l_num_rows = 0) then
  return json_list();
else
  if(l_num_rows = 1) then
    declare ret json_list := json_list();
    begin
      ret.append(
        json(
          json(l_returnvalue).get('ROWSET')
        ).get('ROW')
      );
      return ret;
    end;
  else
    return json_list(json(l_returnvalue).get('ROWSET'));
  end if;
end if;
```

APEX_JSON

- Provides ability to parse and access values

```
DECLARE
    l_values apex_json.t_values;
BEGIN
    apex_json.parse (
        p_values => l_values,
        p_source => '{ "type": "circle", "coord": [10, 20] }' );
    sys.htp.p('Point at '||
        apex_json.get_number (
            p_values => l_values,
            p_path   => 'coord[1]')||
        ','||
        apex_json.get_number (
            p_values => l_values,
            p_path   => 'coord[2]'));
END;
```

Node.js

- Of course, it's native!

```
1  var str = '{"name": "Dan", "languages": ["sql", "pl/sql"]}';
2  var obj = JSON.parse(str);
3
4  obj.languages.push('javascript');
5
6  console.log(JSON.stringify(obj));
```

Oracle Database 12c

- 12.1.0.2 adds new JSON capabilities
- Simple dot-notation allows for easy access to data
  - JSON path expressions are also supported
- New functions
  - JSON_VALUE, JSON_QUERY, JSON_TABLE
- New conditions
  - JSON_EXISTS, IS JSON, IS NOT JSON, JSON_TEXTCONTAINS

Summary

- JSON is a simple data interchange format
  - Easy to learn and use
- There are many options for creating and working with JSON in Oracle
- Oracle Database 12c added JSON capabilities to the SQL engine
  - Very robust and performant


ORACLE®