
ORACLE®

Creating and Working with JSON in Oracle Database

Dan McGhan
Oracle Developer Advocate
JavaScript & HTML5
January, 2016

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 browser based & native mobile applications
  - Lighter weight and easier to use than XML

JSON overview

- Based on 2 structures (can be recursive)
  - object: {}
  - array: []
- Objects are made of key/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  }
```

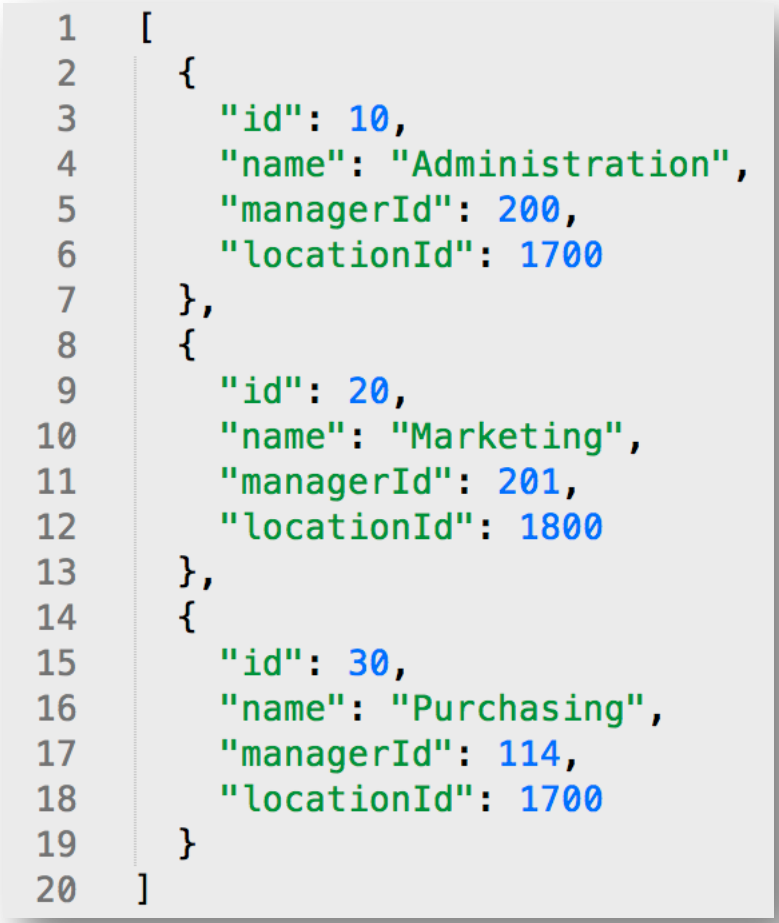
Now with “cooler” keys

```
{
  "id": 10,
  "name": "Administration",
  "managerId": 200,
  "locationId": 1700
}
```

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:
    - ISO 8601: "2016-01-20T16:17:52.792Z"
    - Epoch time: 1453324612507

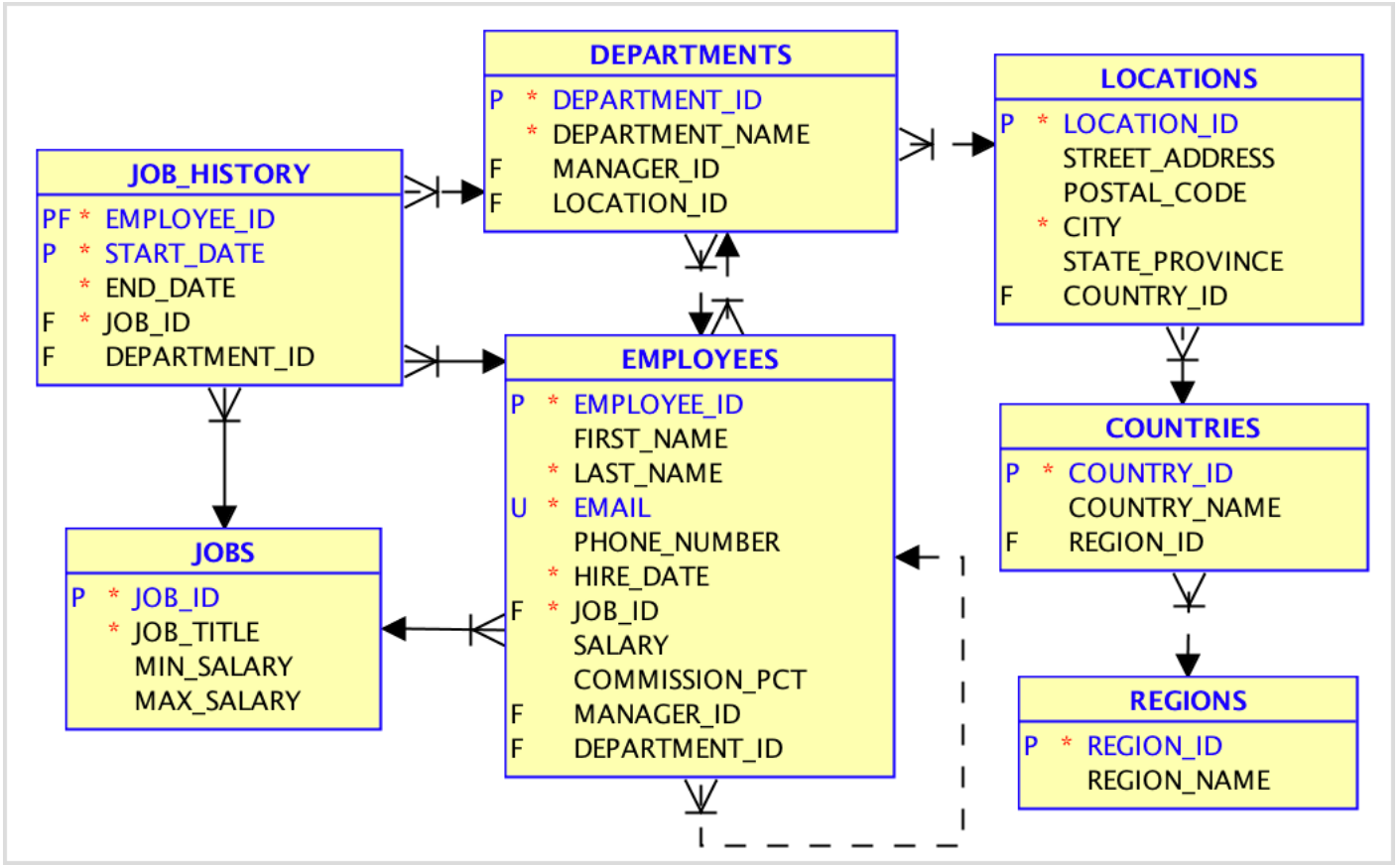
Agenda

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

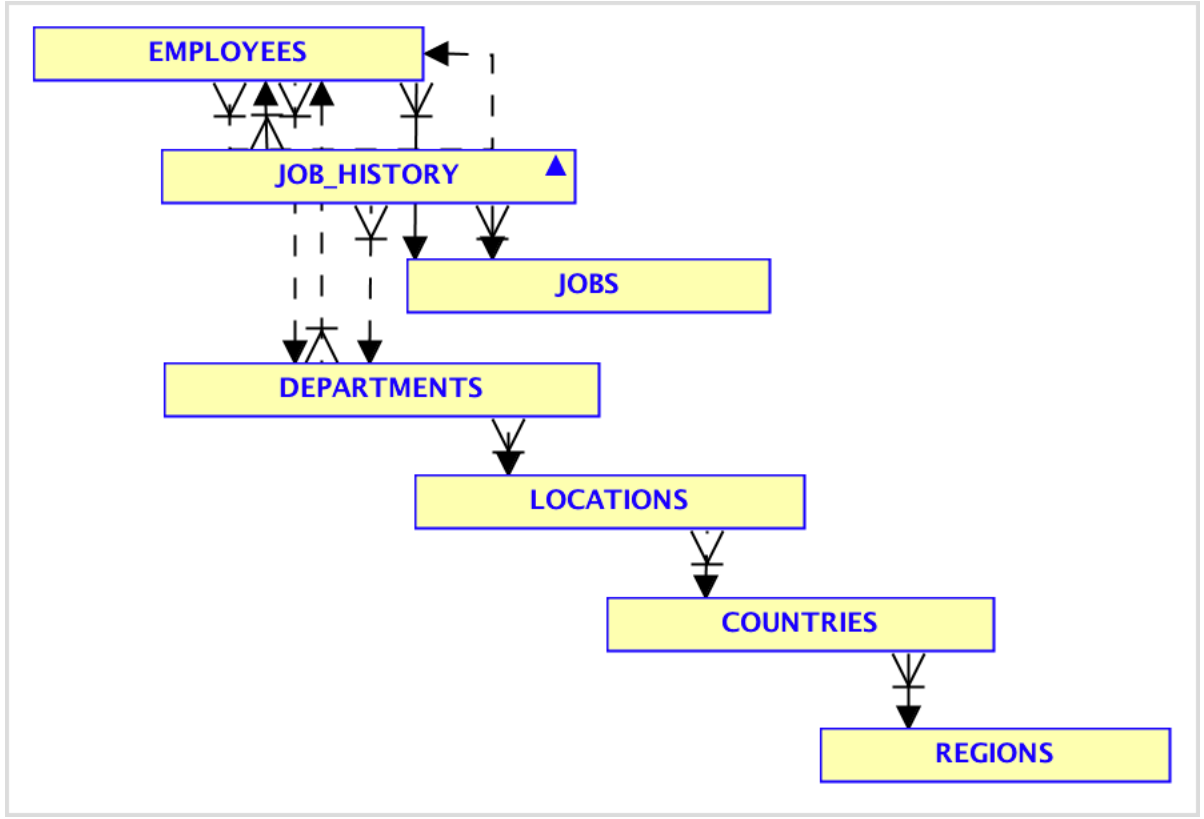
Options for creating JSON in Oracle

- Roll your own
- PL/JSON
- APEX_JSON
- ORDS
- Pick almost any language
  - Most have a driver for Oracle and features for JSON

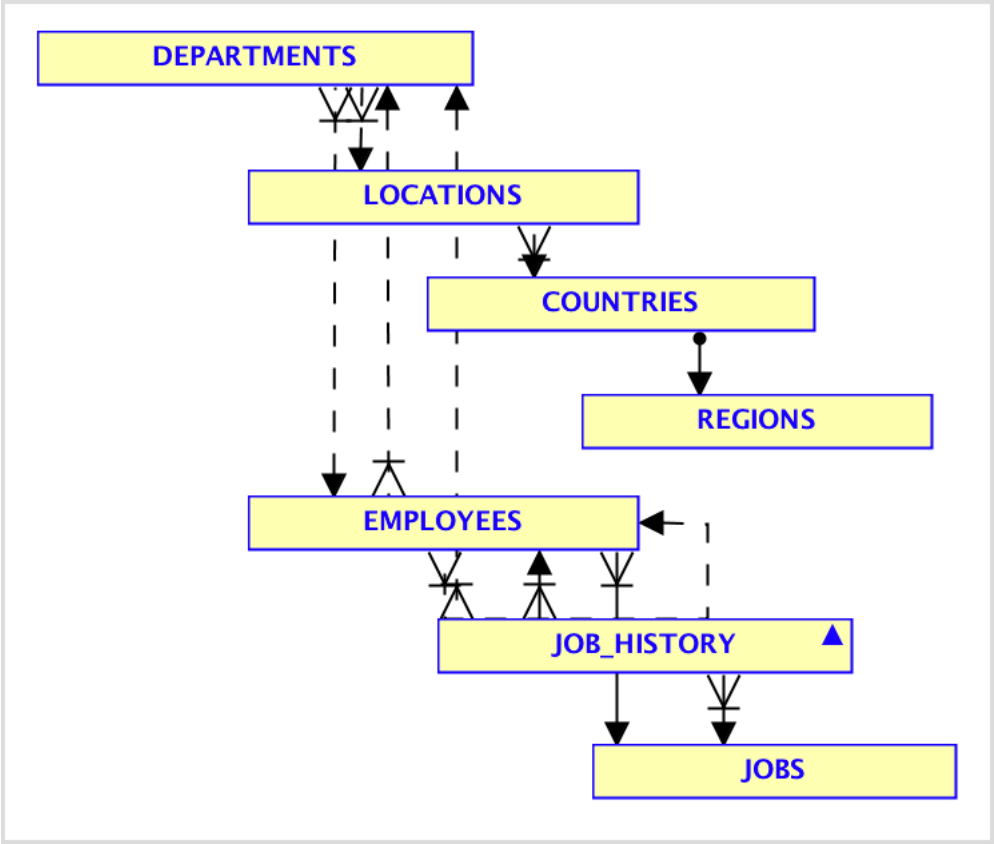
HR Schema



Traversing from employees down



Traversing from departments down



Resulting JSON


```
{
  "id": 10,
  "name": "Administration",
  "location": {
    "id": 1700,
    "streetAddress": "2004 Charade Rd",
    "postalCode": "98199",
    "country": {
      "id": "US",
      "name": "United States of America",
      "regionId": 2
    }
  },
  "manager": {
    "id": 200,
    "name": "Jennifer Whalen",
    "salary": 4400,
    "job": {
      "id": "AD_ASST",
      "title": "Administration Assistant",
      "minSalary": 3000,
      "maxSalary": 6000
    }
  },
  "employees": [
    {
      "id": 200,
      "name": "Jennifer Whalen",
      "isSenior": true,
      "commissionPct": null,
      "jobHistory": [
        {
          "id": "AD_ASST",
          "departmentId": 90,
          "startDate": "17-SEP-1995",
          "endDate": "17-JUN-2001"
        },
        {
          "id": "AC_ACCOUNT",
          "departmentId": 90,
          "startDate": "01-JUL-2002",
          "endDate": "31-DEC-2006"
        }
      ]
    }
  ]
}
```

Roll your own

- JSON is easy, I can make it myself, right?
- 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
  - Creates URL to serve JSON content
- Supports SQL to JSON
  - Does have some limitations on the JSON that can be generated
  - 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

Agenda

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

Options for working with JSON in Oracle

- Roll your own
- PL/JSON
- APEX_JSON
- ORDS
- Pick almost any other language
  - Most have a driver for Oracle and features for JSON
- New features in 12c

JSON features in Oracle Database 12c (12.1.0.2)

- Simple dot-notation allows for easy access to data
- Functions
  - JSON_VALUE
  - JSON_QUERY
  - JSON_TABLE
- Conditions
  - IS (NOT) JSON
  - JSON_EXISTS
  - JSON_TEXTCONTAINS

Simple dot-notation

- An easy way to get the value of keys in the document
  - All values returned as VARCHAR2(4000)
- Gotchas
  - Must add the IS JSON check constraint to the column
  - Must alias the table and use it in the SELECT clause
  - JSON keys ARE case sensitive
  - Keys must be valid SQL identifiers
  - No support for Boolean

Typical usage of JSON functions & conditions

```
1   select JSON_VALUE(),
2     JSON_QUERY()
3   from JSON_TABLE()
4   where JSON_VALUE() = ...
5     and JSON_EXISTS()
6     and JSON_TEXTCONTAINS()
```

Overview of JSON path expressions

- Example: `$.location.streetAddress`
- Allow access to JSON documents via SQL
  - An extension of JSON syntax
    - They are case sensitive
    - Add wildcards, array ranges, predicates
- Can address
  - entire document
  - scalar values
  - arrays
  - objects

Basic syntax of JSON path expressions

- Example: `$.employees[0].name`
- Context item
  - Always starts with a dollar sign (`$`), representing the document
- Object step
  - A period (`.`) followed by a key name
- Array step
  - Left bracket, index expression, followed by a right bracket
  - Examples: `[*]`, `[0]`, `[0-2]`, `[0,1, 3 to 5]`

Clauses used in JSON functions and conditions

- RETURNING Clause
  - Optional Keywords
    - PRETTY – For pretty-printed JSON
    - ASCII – For escaping non-ASCII Unicode characters
- Wrapper clauses
  - WITH WRAPPER
  - WITHOUT WRAPPER
  - WITH CONDITIONAL WRAPPER
- Error clauses

JSON_VALUE

- Returns a single scalar value from a JSON document
- Takes in 2 arguments
  - JSON column
  - JSON path expression
- Has optional modifiers to
  - Specify data types returned
  - Control error handling

JSON_QUERY

- Only returns an array or object from a JSON document
- Takes in 2 arguments
  - JSON column
  - JSON path expression
- Has optional modifiers to
  - Wrap results
  - Pretty results
  - Control error handling

JSON_EXISTS

- Returns a Boolean indicating a match of a JSON path expression
  - Allows differentiation between missing keys, null values, and empty values
- Takes in 2 arguments
  - JSON column
  - JSON path expression

JSON_TABLE

- Enables the creating of inline relational view of JSON data
  - A powerful means work with JSON using relational tools
  - Includes many options for controlling output
- Uses a driving table which contains the JSON documents
  - No need to specify a join condition, it's implicit
- Avoid joins on data created by JSON_TABLE via the same documents
  - Bring all required values through as needed

JSON_TEXTCONTAINS

- Enables full-text searching of JSON documents
- Takes in 3 arguments
  - JSON column
  - JSON path expression
  - The string to search for
- Must be used with JSON Oracle Text Index
  - CTXSYS.JSON_SECTION_GROUP

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
  - Provides the simplicity of JSON with the power of Oracle Database
- Next steps
  - Read the docs JSON in Oracle Database
  - Checkout the HOL in the Database App Development VM
  - Read more about Relational to JSON in Oracle Database


ORACLE®