



RM OUG

ROCKY MOUNTAIN ORACLE USERS GROUP

Training Days Conference
February 15 - 17, 2023

Be Informed and Inspired to Initiate new Ideas to Improve your work lives.

JSON, the Database & I



Robert Marz
DATABEE
Die IT-Architekten



Robert Marz – Independent Consultant

Primary Role

Senior Technical Architect
with database centric view of the world

DOAG (German Oracle User Group)

Active Member of Database Community
Responsible for Cloud Topics



@RobbieDabee



<https://robbie.databee.org>



robert.marz@databee.org



Oracle ACE
Pro



500+ technical experts helping peers globally

The **Oracle ACE Program** recognizes and rewards community members for their technical and community contributions to the Oracle community

3 membership tiers



For more details on Oracle ACE Program:
ace.oracle.com



Nominate
yourself or someone you know:

ace.oracle.com/nominate



A low-angle shot of a red-painted wooden ladder extending towards the top of the frame. A person's hand is visible on the right, gripping one of the rungs. The background is a clear blue sky with scattered white clouds. A horizontal bar with a red-to-orange gradient is positioned across the middle of the image, containing the word 'Introduction' in white text.

Introduction



An Overview

How to deal with JSON in the Database

Absolutly Not

All JSON Commands
Complete Syntax Diagrams
Evolution over Database Versions

A horror-themed background image featuring Jason Voorhees from the Friday the 13th franchise. He is wearing his signature white hockey mask with black eye holes and a dark, heavy jacket. He is holding a large machete in his right hand, which is raised. His left hand is extended forward, palm facing the viewer. The lighting is dramatic, with a blue glow on the left and a yellowish glow on the right. A semi-transparent orange and red gradient banner is overlaid across the middle of the image.

JSON Document Format



Setting the Stage: Who is Jason?

JSON Java Script Object Notation

Java Script Object Notation

Lightweight Format

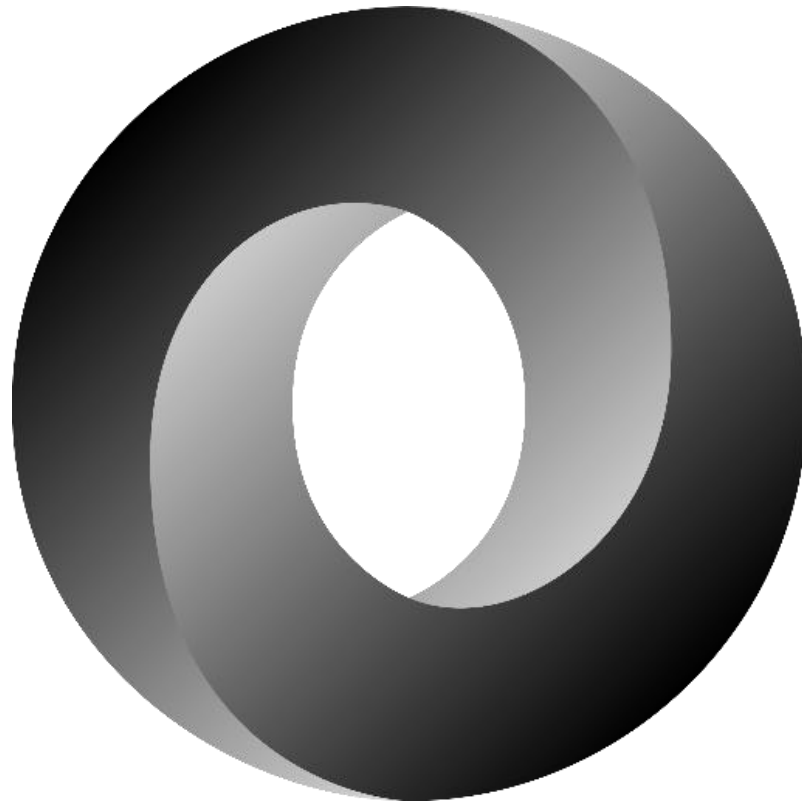
Text-based
Data Interchange
Document Format (can be stored as Files)

Originally designed

to persist JavaScript Objects

Standards:

[json.org](https://www.json.org)
[ECMA-404](https://www.ecma-international.org/standards/650/ECMA-404/)





©OYE IOV - stock.adobe.com

Structures

unordered
collection

“key”:“value” pairs

separated by commas

enclosed by braces { }

also realized as **object**, record, struct, dictionary, hash table, keyed list or associative array

separated by commas

ordered
list of
values

enclosed by square brackets []

also realized as **array**, vector, list or sequence



JSON: Types of Values

```
{  
  "string": "String are quoted",  
  "escape": "\n control chars",  
  "number": 1234,  
  "boolean": true,  
  "collection": {  
    "object": "fields",  
    "can": "nested"  
  },  
  "array": [ "one",  
            "two",  
            {"three": true} ]  
}
```

Types of Values

string

“always enclosed in double quotes”

number

Integer	-256
Float	256.123
E-notation	2.3e3

Boolean

true
false
null

Object

Nested Objects
Arrays

Array

Simple Types
Objects



JSON Document Format: Odds & Ends



Characteraset

always Unicode

Number encoding

always English, Decimal Point

Whitespace

between Tokens is ignored
space, linefeed, carriage return, tab

Dates are represented as strings

usually ISO 8601 Zulu (UTC) Time
"2019-11-19T15:13:23Z"

JSON is Schemaless

no constraints for your implementation
Developers Hell - when dealing with documents not produced by your code
[JSON Schema](#) addon – not yet a standard

©whitehouse - stock.adobe.com



Storing JSON Data



Storing JSON Docs in the Database



Text-based columns

- Varchar2
- CLOB
- BLOB

BLOBs

- eliminate NLS-Layer (JSON is always UTF-8)
- Works well since 12.2
- Database character set should be AL32UTF8

Check constraints (is_json)

- Sanity
- enables dot Notation for querying

Native JSON Data Type “OSON” Format

- 21c New Feature



Speeding things up –Function Based Indexes

```
select *  
  from departments_json  
 where json_value ( department_data  
                  , '$.department' ) = 'Sales'
```

Function Based Index – Useful when searching for a specific attribute

```
create index dept_department_name_i on  
  departments_json (  
    json_value (  
      department_data, '$.department'  
      error on error  
      null on empty  
    )  
  )
```



Speeding things up – JSON Search Indexes

create

search index

dept_json_i

on departments_json

(department_data)

for json;

select *

from departments_json d

where json_textcontains

(department_data, '\$'

, 'Public');

Oracle Text

Index

Can be used for

json_value

json_textcontains

Indexes

whole Document for every row

[Chris Saxon: How to Store, Query, and Create JSON Documents in Oracle Database](#)



When do I store JSON Docs



©vipman4 - stock.adobe.com

Original Document

needed for

- Reference
- Archive
- Redistribution

JSON structure

Varies

and may be queried later on



When Do I convert JSON to relational?

Whenever I need to work with the contents

JSON is an Interchange Format

Almost always

A photograph of a warehouse loading dock. A worker wearing a yellow hard hat and safety vest is operating a blue forklift. The forklift is positioned on a metal ramp leading to the back of a white truck. A large pallet of goods is being loaded onto the truck. The scene is illuminated by bright sunlight, creating a strong lens flare effect. A semi-transparent orange and red banner is overlaid across the middle of the image, containing the text 'Feeding JSON to the DB'.

Feeding JSON to the DB



Loading JSON into the Database: Insert

Datatype
BLOB



```
insert into departments_jsonjson
values ( 110, utl_raw.cast_to_raw (
'{"department": "Accounting",
"employees": [
{
"name": "Higgins, Shelley",
"job": "Accounting Manager",
"hireDate": "2002-06-07T00:00:00Z"
},
{
"name": "Gietz, William",
"job": "Public Accountant",
"hireDate": "2002-06-07T00:00:00Z"
}
]
}'));
```

JSON Documents

are Text

Insert / Update

works fine
as long as Documents size < 4000 /
32000

Check Constraint

is_json
rejects non-json Texts



Loading JSON into the Database: External Table / SQL*Loader

create table

```
json_dump_file_contents
  (json_document blob)
organization external
( type oracle_loader
  default directory json_dump_dir
  access parameters
  ( records delimited by 0x'0a'
    disable_directory_link_check
    fields (json_document raw)
  )
  location ('JSONdocs.dmp')
)
parallel
reject limit unlimited;
```

JSON Dump Files

contain one JSON Doc per row
usually produce by NoSQL
DBs
e.g. MongoDB

Load via

SQL *Loader
External Table



Loading JSON into the Database: SQLcl

```
var HashMap = Java.type("java.util.HashMap");
var bindmap = new HashMap();

print("\nreading file: " + args[1]);
var filePath=args[1];

var blob=conn.createBlob();
var bstream=blob.setBinaryStream(1);

java.nio.file.Files.copy(
    java.nio.file.FileSystems.getDefault().getPath(filePath)
        , bstream );
bstream.flush();

bindmap.put("inhalt", blob);
bindmap.put("pfad", filePath);

if(!util.execute( "insert into dokument "
    + "(datei_inhalt,datei_pfad) "
    + " values(:inhalt, :pfad)"
    , bindmap)
){ print("insert failed, exiting");
  exit;
}
```

SQLcl

SQLDeveloper Worksheet as CLI
99,8% SQL*Plus compatible

Modern CLI

Tab-Completion
Editor
History

Enhancements

Liquibase & HashiCorp Vault integration
DDL Command
lots more

Scripting

in JSR-223 Languages
(e.g. JavaScript, Python, etc)
[Oracle DB Tools GitHub Examples](#)



Loading JSON into the Database: REST Service



Oracle REST Data Services ORDS

Free
Java-Based
evolved from APEX Listener

REST enables Database

very easy
SQL Developer Wizards
ORDS REST Services are pure Metadata

Developed & Deployed in Minutes

[POST Up a BLOB to an Oracle Table via REST - Jeff Smith](#)



Loading JSON into the Database: REST Service Example

The screenshot displays the Oracle SQL Developer interface. On the left, the 'REST Data Services' tree is visible, showing a 'POST' service under a 'media' module. The main window shows an SQL Worksheet with the following PL/SQL code:

```
1 declare
2   image_id integer;
3 begin
4   insert into gallery (title,content_type,image)
5     values (:title,:content_type :body)
6     returning id into image_id;
7   :status := 201;
8   :LOCATION := IMAGE_ID;
9 end;
```

The `:body` parameter in the `values` clause is highlighted with a red box.

[POST Up a BLOB to an Oracle Table via REST - Jeff Smith](#)



Loading JSON into the Database: utl_http



Oracle Database

can do `http_requests`
e.g. for using REST Services
Package `utl_http`

Prerequisites

ACL/ACE for Network Access
Wallet for https-Requests

PL/SQL implementation

straight forward
Example: REST DB Links



Querying JSON



JSONPath - Reference

JSONPath is Xpath for JSON

Used all over the place

Uses dot-Notation: `$.store.book[0].title`

Returns JSON Document or single element

[JSONPath Online Evaluator](#)

Expression

Description

\$

the root object / element

@

the current object / element

. or []

child operator

[start:end:step]

array slice operator

?()

filter expression

JSONPath	Result
<code>\$.store.book[*].author</code>	the authors of all books in the store
<code>\$.author</code>	all authors
<code>\$.store.*</code>	all things in store, which are some books and a red bicycle.
<code>\$.store..price</code>	the price of everything in the store.
<code>\$.book[2]</code>	the third book
<code>\$.book[(@.length-1)]</code> <code>\$.book[-1:]</code>	the last book in order.
<code>\$.book[0,1]</code> <code>\$.book[:2]</code>	the first two books
<code>\$.book[?(@.isbn)]</code>	filter all books with isbn number
<code>\$.book[?(@.price<10)]</code>	filter all books cheaper than 10
<code>\$.*</code>	All members of JSON structure.



Querying JSON: Dot-Notation

```
select d.department_data.department
from departments_json d
```

```
select d.department_data.employees[0].name
from departments_json d
where department_id = 110;
```

EMPLOYEES
Gietz, William

```
select d.department_data.employees[*].name
from departments_json d
where department_id = 110;
```

EMPLOYEES
["Gietz, William", "Higgins, Shelley"]

```
-- Column needs is_json constraint
-- or "treat as json" function (18c)
```

```
with j_data as (
  select treat (
    d.department_data as json
  ) as department_data
from departments_json d
where department_id = 110
)
select j.department_data.department
from j_data j
where department_data is json;
```

DEPARTMENT
Accounting



JSON_TABLE

Lives inside the SQL-From-Clause

Produces **Rows** and **Columns**

Accepts LOBs with JSON data

Included in SQL:2016 Standard



Querying JSON: The JSON_TABLE Operator (2/2)

The JSON Document

```
select wert
  from json_table( ['Eins", "Zwei", "Drei",
                  "Vier", "Fünf", "Sechs"]
                  , '$[*]'
                  columns wert varchar2 path '$'
                )
/
```

Produces rows
(JSONPath Object)

Produces columns
(JSONPath Element)

WERT

Eins
Zwei
Drei
Vier
Fünf
Sechs

6 rows selected

Elapsed: 00:00:00.011



Dates in JSON – Dealing with ISO 8601 Zulu

JSON Dates usually ISO 8601 Zulu (UTC) Time Strings
“2019-11-19T15:33:54Z”

```
-- Converting in Oracle is tricky
cast( to_timestamp_tz
      ( to_char(
          to_date(timestamp, 'YYYY-MM-DD"T"HH24:MI:SS"Z"')
          , 'YYYY-MM-DD HH24:MI:SS "UTC"')
      , 'YYYY-MM-DD HH24:MI:SS TZR')
      at time zone sessiontimezone as date ) timestamp
```

```
-- Oracle 18c has new date function
select TO_UTC_TIMESTAMP_TZ('2019-11-19T15:33:54Z') timestamp
from dual;
```

A blacksmith is shown working with a piece of metal. The metal is held in a vice on an anvil. The blacksmith is using a hammer to shape the metal. The background is dark and out of focus, suggesting a forge or workshop environment. A semi-transparent orange and red banner is overlaid on the left side of the image, containing the text "Working with JSON".

Working with JSON



Comparing JSON Documents

```
select case
  when json_equal({'one': 1, 'two': ['three', 4]}
    , {'two': ['three'
      , 4]
      , 'one': 1 }
    )
  then 'EQUAL'
  else 'DIFFERENT'
end compare
from dual;
```

COMPARE

EQUAL

```
select case
  when json_equal({'one': 1, 'two': ['three', 4]}
    , {'one': 1, 'two': [4, 'three']})
  then 'EQUAL'
  else 'DIFFERENT'
end compare
from dual;
```

COMPARE

DIFFERENT



Updating JSON Docs – `json_merge_patch`

`JSON_merge_patch()`

- – new Feature in 19c

Creates / Updates /
Delete

- Objects in JSON Docs
Works like Unix patch or SQL merge

```
json_mergepatch({'FirstName':"Eric", "LastName':"Cartman"}, {'LastName':',,Fox'})  
{'FirstName':"Eric", "LastName':"Fox"}
```

```
json_mergepatch({'FirstName':"Eric", "LastName':"Cartman"}, {'Salary':1000})  
{'FirstName':"Eric", "LastName':"Cartman", "Salary":1000}
```

```
json_mergepatch({'FirstName':"Eric", "LastName':"Cartman"}, {'FirstName':null})  
{'LastName':"Cartman"}
```

A photograph of a disassembled motor. On the left is a large, circular stator assembly with multiple layers of copper windings. In the center is the main motor housing, partially cut away to reveal internal components like the rotor and brushes. On the right is a separate gear assembly with a green plastic housing and a metal gear. The background is dark, and a red-to-orange gradient banner is overlaid across the middle.

Generating JSON



Generate JSON Documents

SQL-Functions starting with
12gR2

JSON_Object

JSON_Array

JSON_ArrayAgg





Generate JSON Documents – JSON_Object

```
select json_object ('id2' value cust_id,  
                  'name' value (first || ' ' || last),  
                  'joined' value join_date) customers  
from customers;
```

CUSTOMERS

```
{"id2":1,"name":"Eric Cartman","joined":"2017-07-10T16:26:15"}  
{"id2":2,"name":"Kenny McCormick","joined":"2017-07-10T16:26:15"}  
{"id2":3,"name":"Kyle Brofloski","joined":"2017-07-10T16:26:15"}  
{"id2":4,"name":"Stan Marsh","joined":"2017-07-10T16:26:15"}
```



Generate JSON Documents JSON_Array

```
select json_array(first, last) customers  
from customers;
```

CUSTOMERS

```
["Eric", "Cartman"]  
["Kenny", "McCormick"]  
["Kyle", "Brofloski"]  
["Stan", "Marsh"]
```



Generate JSON Documents JSON_ArrayAgg

```
select json_arrayagg(  
    json_object('id'    value item_id,  
               'name'  value name,  
               'quantity' value stock_quantity)  
    ) stock_items  
from items  
where stock_quantity > 10;
```

STOCK_ITEMS

```
[{"id":345,"name":"Border Patrol Costume","quantity":20},  
 {"id":429,"name":"Air Guitar","quantity":14}]
```

Publishing JSON



Publishing JSON

Publishing JSON

utl_file

let the database
write files

sql*plus

spool files

SQLcl

script your JSON Docs

REST
Service

let the clients request
Documents on demand
deliver via http(s)



Conclusion





Topics covered

What is JSON

Storing JSON content

Loading JSON

Querying JSON

Manipulating JSON

Generating JSON

Publishing JSON

Didn't mention

PLSQL APIs

JSON Data Guide

APEX JSON Packages

Native JSON Data Type, OSON Format (not in depth)

SODA

Oracle Database API for MongoDB

23c JSON Duality Views



JSON, the Database and I

JSON is a flexible Interchange Format

- Lingua Franca of modern Web Apps

Loading JSON is easy

- It's just Text

Features evolve quickly

- 20c JSON Datatype, JSON Dictionary

The Database can be turned into the Backbone of RESTful Services

PLEASE

**DO
TRY THIS
AT HOME**