JSON and PL/SQL: A Match Made in Database

Steven Feuerstein
Oracle Developer Advocate for PL/SQL
Oracle Corporation
steven.feuerstein@oracle.com
@sfonplsql
stevenfeuersteinonplsql.blogspot.com
Practically Perfect PL/SQL (YouTube)

ORACLE

Resources for Oracle Database Developers

- Official homes of SQL and PL/SQL - oracle.com/sql oracle.com/plsql
- Dev Gym: quizzes, workouts and classes - devgym.oracle.com
- Ask Tom - asktom.oracle.com – 'nuff said (**+ new: *Office Hours*!**)
- LiveSQL - livesql.oracle.com – script repository and 24/7 18c database
- SQL-PL/SQL discussion forum on OTN
  https://community.oracle.com/community/database/developer-tools/sql_and_pl_sql
- PL/SQL and EBR blog by Bryn Llewellyn - https://blogs.oracle.com/plsql-and-ebr
- Oracle Learning Library - oracle.com/oll
- oracle-base.com - great content from Tim Hall
- oracle-developer.net - great content from Adrian Billington

Some Questions for You

- Do you write code in the database?
- Do you write UI code as well?
- Do you work with UI developers?
- Do you fight with UI developers?
- Who has the ear of management, the database developers or the UI developers?

What is JSON?

- JavaScript Object Notation
  - A "lightweight", readable data interchange format. In other words, NOT XML. Squiggles instead of angle brackets. **WAY better!** ☺
  - Language independent, but widely used by UI developers, especially those working in JavaScript.
- Built on two structures:
  - Name-value pair collections
  - Order list of values: aka, arrays

```
Key-Value Pair: { “KEY” : “VALUE” }
Embedded: { “KEY” : { “KEY1” : “VAL1”, “KEY2” : “VAL2”} }
Arrays: { “KEY” : [ “VAL1”,”VAL2”,”VAL3”] }
```

What is JSON? (continued)

- JSON object - *unordered* set of name-value pairs
- JSON array - *ordered* collection of values.
- JSON value
  - String in double quotes, a number, Boolean literal, NULL, object or array
- Some terminology
  - Serialize: convert an object to another type. Most common: TO_STRING aka STRINGIFY.
  - Introspection: get information about the JSON objects. Example: IS_ARRAY
- "What constitutes well-formed JSON data is a gray area."

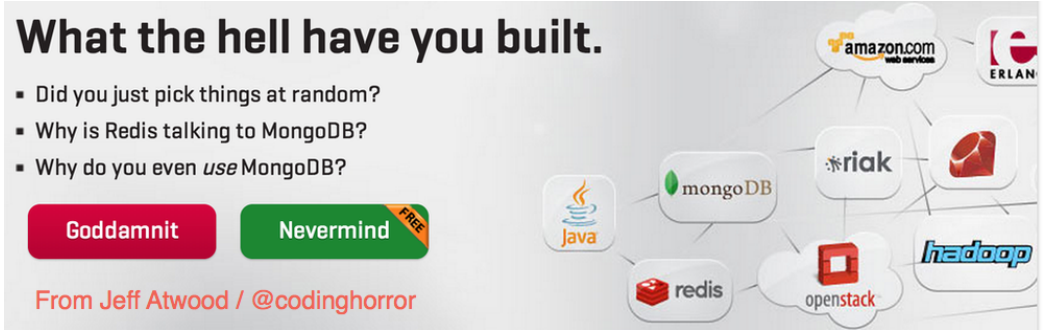
Should Database Developers Care About JSON?

- Do we really have to care about and learn yet another syntax for yet another non-relational chunk of data? [think XML]
- **Yes!**
- JSON is *the* (current) preferred method by which Javascript, Python and other developers interact with data.
- And these days, what **application (UI) developers** say, goes.
- The critical question for **database developers** is:

How can we help those UI developers succeed?

We've Got It (Relatively) Easy

There's a reason for the Framework Insanity of JavaScript



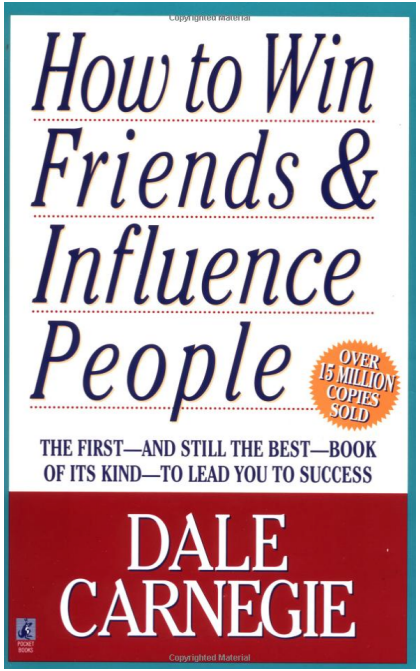
- User interfaces are tied directly and tightly to culture. Uh oh.
- Lots and lots of code (compared to, say, Application Express)
- Microservices, bots, containers, asynchronous communication....
- Endless demand for changes to UIs, since we need to hide all that ever-increasing complexity

So they’ve got it tough, so what?

- That depends on what’s important.
- The only thing that matters – **that should matter** - when it comes to software development is building successful applications.
- Which means **we are all in this together**.
- Let’s start acting like that.

Let's Change the Message to *How can I help?*

- And drop the attitude. ☺
- Don't be so eager to point out where others are *wrong.*
  - You could even admit *you* are wrong.
- Find developer pain points. These come to mind:
  - Performance of DB access
  - Headaches wrestling with SQL
  - Needs JSON-based APIs
- Then offer solutions, of which you have lots.



We can help UI developers – a LOT.

And 12.2 makes it easier than ever before.

- You hate SQL? No problem, we love it and are good at it.
  - *Get* really good at it!
- You want APIs? We've got the best data APIs.
  - PL/SQL is *the* best performing, most secure and productive language for creating APIs to the database, through packages.

- You want JSON?
  - Oracle Database offers native JSON support via SQL and PL/SQL.
- You will only talk REST? No problem.
  - Easy, secure REST APIs (often generated) through Oracle REST Data Services.

Watch ORDS in action at 2:45 with Jeff Smith

JSON and SQL in Oracle Database

https://v.gd/oradbjson

- Oracle Database 12c Release 1 added many great features for native support of JSON in tables and via SQL.
- "IS JSON" constraint for existing types – there is no *JSON type*
  - (N)VARCHAR2, (N)CLOB, BLOB, RAW
- JSON operators in SQL
  - JSON_VALUE, JSON_QUERY, JSON_TABLE, JSON_EXISTS, IS JSON
- JSON Dataguide
  - Discover information about structure and content of JSON documents
- Index JSON data – scalar values and in 12.2 the Search Index.

Use Case: JSON for Flexfields
https://livesql.oracle.com search "flex"

ORACLE

Changing JSON Data in Tables

- Oracle Database offers lots of ways to extract information from JSON documents stored in tables.
- Changes to JSON requires a replacement of the entire document.
  - You cannot, for example, do an "in place" removal of a name-value pair or and element from an array.
- Instead:
  - 1. You *serialize* the JSON data into a PL/SQL variable.
  - 2. Change the JSON data as needed.
  - 3. Run the usual DML statements to modify the table.
- Ah...but how do you go about changing that JSON data?

JSON and PL/SQL in Oracle Database

- Oracle Database 12c **Release 2** built upon the fantastic start in 12.1 with more SQL features and a set of object types to manipulate JSON in PL/SQL.
- The JSON* types provide an in-memory, hierarchical representation of JSON data. Use them to...
  - Check structure, types or values of JSON data. Validate rules, etc.
  - Transform JSON data the "smart way."
  - Construct JSON data programmatically

Not on 12.2?
Check out APEX_JSON and PL/JSON for similar functionality.

PL/SQL JSON Object Types

- JSON_ELEMENT_T
  - Supertype of all those below. Rarely used directly.
- JSON_OBJECT_T
  - Manipulate JSON objects (set of name-value pairs)
- JSON_ARRAY_T
  - Manipulate JSON arrays
- JSON_SCALAR_T
  - Work with scalar values associated with a key
- JSON_KEY_LIST
  - Array of key names, returned by GET_KEYS method

Some JSON Object Type Basics

- Use the *parse* static method to create the in-memory representation of your JSON data.
- *Serialization* does the opposite: converts an object representation of JSON data into a textual representation.
  - The STRINGIFY and TO_* methods
- Use TREAT to *cast* an instance of JSON_ELEMENT_T to a subtype.
  - Most of your code will work with objects and arrays.
- *Introspection* methods return information about your data.
  - Is it an array, is it a string? What is its size? etc.

Introspection Methods

- JSON_ELEMENT_T (the most general type) offers a set of methods to tell you what specific subtype you are working with.
  - IS_OBJECT, IS_ARRAY, IS_SCALAR, IS_NULL, etc.
- The return value of GET_SIZE depends on what it is "sizing":
  - For scalar, returns 1.
  - For object, returns the number of top-level keys
  - For array, returns the number of items

LiveSQL: search for "introspection"

Error Handling and JSON Object Types

- The default behavior of JSON object type methods is to return NULL if anything goes wrong.
  - Consistent with behavior of other JSON APIs already loose in the world.
- But that can lead to problems.
  - Can "escalate" error handling to force the raising of exceptions.
- On a per-object type instance basis, call the ON_ERROR method and pass it a value of 0 through 4.
  - 0 = Return NULL (default), 1= Raise all errors ...

LiveSQL: search for "on_error"

Working with JSON Objects: JSON_OBJECT_T

- JSON object: unordered set of name-value pairs
  - The value could be an array, or another object...
- STRINGIFY: return a string representation of an object
- PUT: change value of existing key or add new one
- PUT_NULL: replace value of key with NULL (or add new)
- REMOVE: remove name-value pair from object
- RENAME_KEY: renames the key in the name-value pair

LiveSQL: search for "JSON_OBJECT_T"

Working with JSON Arrays

- If you see [], you've got an array.
  - Arrays can nested. They can contain scalars or objects.
- STRINGIFY: return a string representation of an array
- PUT: add a new element at the specified position
- PUT_NULL: add a new element with value NULL
- REMOVE: remove specified element from array
- APPEND: append new element on end of array

LiveSQL: search for "JSON_ARRAY_T"

There's No \Escaping JSON!


```
select json_object (
          'department' value d.department_name,
          'employees' value json_arrayagg (
             json_object (
               'name' value first_name || ',' || last_name,
               'job' value job_title )))
from hr.departments d, hr.employees e, hr.jobs j
where d.department_id = e.department_id
and e.job_id = j.job_id
group by d.department_name;
```

- It will be the dominant data exchange format for years to come.
  - And compared to SQL it's *easy*.
- Oracle Database gives you all the tools you need to combine the best of both worlds: relational AND document.
- Use your expertise in SQL, PL/SQL *and* JSON to become an invaluable partner with your UI developers.
  - Help them be successful, and *you* will be successful.


ORACLE®