

SkillBuilders
Oracle Webinar Series:
Introduction to JSON in Oracle

Webinar Series Objectives:

- Learn and use Oracle's new and advanced SQL features

Oracle Webinar Series: Introduction to JSON

Moderator: Dave Anderson

**Presenters: Geoff Wiland for JSON
Garry Lawton for APEX**

www.skillbuilders.com

+1-401-783-6172

About Your Presenters



- Geoff Wiland
 - Senior Oracle Instructor and Developer
 - Certifications: Oracle Certified Associate (OCA), Certified Technical Trainer CTT+
- Garry Lawton
 - Oracle APEX Expert and DBA
 - Author of SBODM
 - SkillBuilders Oracle Database Manager
 - SkillBuilders Developer and Teacher
- Dave Anderson
 - Founder, President and Oracle DBA
 - www.skillbuilders.com
 - 1.401.783.6172 x12

Introduction to JSON

▣ Objectives

- What is JSON?
- Storing JSON in the database
- Updating JSON in the database
- Consuming JSON data
- Converting JSON to relational data
- Converting relational data to JSON data

What is JSON? ...

- A standard markup language
 - Used to store and exchange data in a simple universal format as character data
- Previously done with XML (eXtensible Markup Language)
 - JSON has replaced XML in most applications
 - Less storage than XML so faster transmission
 - Less complex than XML so faster parsing with less memory overhead
- JSON is now a de facto standard format for document exchange
- JSON data can be accessed by most computer languages

... What is JSON?

- JSON documents are text
- Oracle's implementation of JSON is a work-in-progress.
 - Introduced in 12c
 - Enhanced significantly in all higher versions
 - We will focus on 12c and discuss later significant enhancements
- Oracle 21c includes a native JSON datatype
- We are providing an overview of JSON
- For more details see Oracle's *JSON Developer's Guide*:

- [12.2 version](#)

© SkillBuilders, Inc.

- [21c version](#)

JSON Document Data Format ...

- Stores data as Objects and Arrays
- Object = collection of name/value pairs in curly braces

```
{  
  "empid": 2257,  
  "firstName" : "Robert",  
  "lastname" : "Walters"  
}
```

- Array = multiple values in square brackets

```
[  
  "Walters", "Peterson", "Singh"  
]
```

- JSON Arrays are 0 based!

... JSON Document Data Format ...

□ Object with Array Data

```
{  
  "empid": 2257,  
  "firstName" : "Robert",  
  "lastname" : "Walters",  
  "dependents": ["William",  
"Roberta", "Teresa"]  
}
```


... JSON Document Data Format

- Objects and arrays can be embedded to any depth required.

- Array containing objects:

```
[ { color: "red", value: "#f00" }, { color: "green", value: "#0f0" }, { color: "blue", value: "#00f" }, { color: "cyan", value: "#0ff" }, { color: "magenta", value: "#f0f" }, { color: "Yellow", value: "#ff0" }, { color: "black", value: "#000" } ]
```

- Object containing objects and array:

```
{ "firstName": "Rick", "lastName": "Jackson", "gender": "male", "age": 24, "address": { "streetAddress": "126 Main Street", "city": "San Jose", "state": "CA", "postalCode": "394221" }, "phoneNumbers": [ { "type": "home", "number": "555-458-4921" }, { "type": "cell", "number": "555-894-2783" } ] }
```

Storing JSON in the Database

...

- Recall JSON documents are just text
 - Store as VARCHAR2 if small
 - Up to 4k generally, or 32k with extended data types
 - Store larger JSON documents as CLOB or BLOB
 - Oracle recommends BLOB instead of CLOB
 - BLOB saves space
 - But often more difficult to deal with BLOB due to required character conversion
 - We will use VARCHAR2 or CLOB in our examples for simplicity
 - When storing as a database column, use the new `is json` check constraint to assure it is a valid JSON document
 - Oracle 21c introduced a JSON datatype
 - Use it for all JSON data when your database is 21c or higher
 - Much more efficient for DML and query processing
 - When using JSON datatype, the `is json` check constraint is not needed

... Storing JSON in the Database

- Let's see how to:
 - Create a table with a JSON CLOB column with an “is json” constraint
 - What happens when we try to add good and bad JSON data
 - See how to display the data in SQL Developer
 - See script json1.sql

Updating JSON in the Database...

- Update JSON data before 19c
 - Must update entire JSON column
 - See script json2.sql

... Updating JSON in the Database

- Use `JSON_MERGEPATCH` for limited update capability
 - Created for 21c and backported to 19c
 - Cannot patch JSON arrays
 - Must replace the entire array
 - Cannot use specific NULL values
 - See [JSON MergePatch](#) for details
- In 21c much more sophisticated updates can be done with `JSON_TRANSFORM`
 - Syntax more complex
 - But supports updating, inserting, and deleting any part of JSON data
 - See [21c JSON TRANSFORM](#) for details

Consuming JSON data with Dot Notation ...

- Select entire JSON column
 - Parse it manually (using substr, instr, etc.)
 - Works with or without the `is json` column constraint
 - Too much work – error prone
- Select only the parts of the JSON column you need
 - Use dot notation to select attribute(s) you want
 - Must have the `is json` constraint
- Use indices to get array elements
 - Recall that JSON arrays are 0 based

More JSON to come, but first: Back to Dave for a brief commercial

- **skillbuilders.com/json-training**
 - Next JSON class is 4 half days: 7/18 – 7/21

- **skillbuilders.com/apex-application-development**

- **For more information:**
 - Call 401-783-6172
 - Or email **gary@skillbuilders.com**

... Consuming JSON data with Dot Notation

□ Limitations of dot notation

- Dot notation will NOT allow us to differentiate between a null value and a missing element!
- Must have the `is json` column constraint
 - Workaround if no `is json` column constraint:
 - Oracle 18c added a `TREAT` function to treat the column as JSON
- Using dot notation on data that is not proper JSON:
 - May return incorrect data
 - May return NULL when item is missing
 - Does not result in an error
- The takeaway:
 - BEWARE using dot notation with data that is not well-formed JSON.

Use the IS JSON constraint to enforce well-formed JSON data!

Better ways to Consume JSON data

- JSON Dot Notation is limited
- JSON Functions provide more flexibility:
 - JSON_VALUE selects a scalar value as a SQL value
 - JSON_QUERY selects one or more values as a SQL string
 - Typically used to select parts of a JSON object or array
 - JSON_TABLE generates a virtual table/inline view from JSON data

Consuming JSON data with JSON_VALUE ...

- We often need to retrieve single values
 - Can retrieve the entire JSON column and parse it
 - Too inefficient
 - Too much code to write
 - Can use Dot Notation
 - But Dot Notation has limitations
 - Solution: JSON_VALUE
- Retrieves a single scalar value
- Takes 2 arguments:
 - JSON column name
 - A valid JSON *Path Expression* which starts with a \$ character to identify a scalar value

... Consuming JSON data with JSON_VALUE

- JSON_VALUE returns a single scalar value
 - Raises an error if a non-scalar is returned
- JSON_VALUE returns a NULL by default if no value is found
 - Actually returns an error, but defaults to NULL ON ERROR, so a NULL is returned
 - So how do we differentiate between a real NULL and no value? Solution:
 - Use the ERROR ON ERROR clause
 - Default clause is NULL ON ERROR

□ See script json6.sql

© SkillBuilders, Inc.

Consuming JSON data with JSON_QUERY

- ❑ JSON_VALUE retrieves a single scalar value only
- ❑ But what if we need a fragment of JSON data that is not a single value (e.g., an array)?
 - Use the JSON_QUERY function
- ❑ JSON_QUERY takes the following arguments:
 - JSON column name
 - A valid JSON *Path Expression* which starts with a \$ character with a “returning” clause
 - The “returning” clause specifies a return datatype.
 - VARCHAR2 is the only valid return datatype in 12c.
 - CLOB and BLOB return datatypes are supported in 18c.
 - Returns ‘NULL’ for a null value
 - **Must add [PRETTY] WITH WRAPPER clause if returning more than one object**
 - As with JSON_VALUE, use the ERROR ON ERROR clause to detect errors.
- ❑ Returned string is in JSON format

Consuming JSON data with JSON_TABLE ...

- ❑ JSON_VALUE returns a single scalar value
- ❑ JSON_QUERY returns fragments of JSON data in JSON format
- ❑ But what if we want to project JSON data as rows and columns to support:
 - Displaying data as standard columnar reports
 - Inserting into relational tables as rows and columns

Create relational VIEWS from JSON data

... Consuming JSON data with JSON_TABLE ...

- For a simple JSON document with no arrays:
 - We can use `JSON_VALUE` to retrieve a single scalar value
 - We can use `JSON_QUERY` to retrieve a JSON fragment as a character string
 - And we may be able to convert JSON to relational data
- But if the JSON document has arrays we have a problem
 - Each employee in the employees array must generate a row in the relational view

... Consuming JSON data with JSON_TABLE ...

- Concept – this is a little *tricky*:
 - Use JSON_TABLE to create a table that has one row for each value in a JSON array
 - List the table created by the JSON_TABLE function, together with the table containing the JSON column, in the FROM clause
 - Oracle does an **implicit JOIN** of the tables, resulting in one row for each element in the JSON array
 - Result is relational data
 - This is REALLY COOL stuff!

... Consuming JSON data with JSON_TABLE

- JSON_TABLE takes the following arguments:
 - The JSON column name
 - A list of high-level attributes to be displayed
 - A “nested” path for any arrays below a high-level attribute
 - A list of all the columns in the array to be displayed
 - Use standard JSON path expressions followed by an optional error clause
- See scripts json8.sql and json9.sql

Searching JSON data

- Use `JSON_VALUE` to search for specific values in a JSON column
- But JSON data can be huge, so how can we search large amounts of JSON data efficiently?
 - Create indexes!!
- Index types
 - JSON function-based indexes when you know what you are searching for
 - Oracle TEXT indexes if you might search on anything
- See [12c JSON indexing](#) for more details

Creating JSON data from relational data

- We learned how to create relational data from JSON data
- Now let's create JSON data from relational data
 - Use special JSON Functions built for this purpose
- Convert relational data to fully formatted JSON data
- JSON data will be returned as a SQL value
 - Character data
 - CLOB if too large to fit in a VARCHAR2 variable

JSON Functions to create JSON...

- 4 functions simplify converting relational data to JSON data
 - JSON_object
 - Returns one document for each row of input
 - Returns data as attribute/value pairs surrounded by { }
 - JSON_objectagg
 - Combines all rows into one document
 - Returns data as attribute/value pairs surrounded by { }
 - JSON_array
 - Returns one document for each row of input
 - Returns data as an array surrounded by []
 - JSON_arrayagg
 - Combines all rows into one document
 - Returns data as an array surrounded by []
 - See SQL/JSON Generation Functions for details

... JSON Functions to create JSON

- These JSON Functions are guaranteed to generate proper JSON
- Alternative is to build JSON programmatically
 - More code
 - May need PL/SQL to generate
 - Very error-prone
- JSON Functions, especially for embedded arrays, can get complex
 - But better than the alternative
- Best approach is to build JSON objects from the inside out

PL/SQL and APEX Support for JSON

- PL/SQL has also been enhanced to support JSON.
 - That capability is beyond our scope.
- A good introductory article can be found at [PL/SQL Support for JSON](#).
- For details of how to use PL/SQL to manipulate JSON data in Oracle 12.2 see [PL/SQL JSON Object Types](#)
- APEX 5 provides JSON support via the `APEX_JSON` package
 - Provides ability to generate and parse JSON documents
 - This package can also be used directly in PL/SQL without APEX
 - Beware: `APEX_JSON.GET_VARCHAR2` retrieves data from PL/SQL arrays that start with

Summary of JSON Support

- Oracle provided basic JSON support starting in 12.2
- JSON support has improved with each version since then
- 21c is the first version to support a JSON datatype
- New features provide:
 - Enhanced data manipulation
 - Better processing efficiency

What's Next?

- Coming soon for APEX Developers:
 - REST services and JSON in APEX

Thanks for Listening

- skillbuilders.com/json-training
 - Next JSON class is 4 half days: 7/18 – 7/21

- skillbuilders.com/apex-application-development

- **For more information:**
 - Call 401-783-6172
 - Or email gary@skillbuilders.com