

JSON PROCESSING IN THE DATABASE

GETTING, PROCESSING, AND RESHAPING
JSON DATA USING POSTGRESQL 9.4



RYAN B. HARVEY

JANUARY 7, 2015



JSON DATA

- JAVASCRIPT OBJECT NOTATION (JSON) DATA INTERCHANGE FORMAT: RFC 7159
- DATA FORMATTED IN JAVASCRIPT SYNTAX
- BASIC ELEMENTS:
 - OBJECT {"key": "value", "key2": "value"}
 - LIST ["string", 5, 6.24, true, "other string"]
 - ATOMICS "string", 5, 2.64, true, false, null

JSON DATA: EXAMPLE

```
[
  {
    "id": "usajobs:377106500",
    "url": "https://www.usajobs.gov/GetJob/ViewDetails/377106500",
    "maximum": 254848,
    "minimum": 98000,
    "end_date": "2015-07-29",
    "locations": ["San Francisco, CA", "Washington, DC"],
    "start_date": "2014-07-31",
    ...
  },
  {
    "id": "usajobs:382608000",
    "url": "https://www.usajobs.gov/GetJob/ViewDetails/382608000",
    ...
  },
  ...
]
```

BUT... ANALYSIS DATA

- AS A DATA ANALYST/SCIENTIST/HACKER/..., WE OFTEN WANT TABULAR ANALYSIS DATA
- JSON FORMAT IS CONVENIENT FOR API RESPONSES, BUT NOT USUALLY FOR ANALYSIS
- SOMEHOW, WE NEED TO MUNGE THAT DATA INTO A CLEAN TABLE (OFTEN LOTS OF WORK)
- POSTGRESQL ADDS JSON PROCESSING TO THE ALREADY AVAILABLE RELATIONAL MODEL (UNLIKE OTHER RELATIONAL DBMS WITH JSON)

NOW, TO THE DATABASE!

- FOR THIS, WE'LL BE LOOKING AT POSTGRESQL VERSION 9.4 (RELEASED DEC 18, 2014)
- POSTGRESQL ADDS EXTENSIONS TO SQL FOR JSON MANIPULATION
- JSON IS SUPPORTED VIA COLUMN TYPES
 - json (TEXT, FOR FAST INSERTS; 9.2, IMPROVED IN 9.3)
 - jsonb (BINARY, INDEXABLE, FOR FAST READS; 9.4)
- OTHER DATABASES DO THIS DIFFERENTLY!

REVIEW: RELATIONAL DATA

- RELATIONAL DATA IS ORGANIZED IN TABLES CONSISTING OF COLUMNS AND ROWS
- FIELDS (COLUMNS) CONSIST OF A COLUMN NAME AND DATA TYPE CONSTRAINT
- RECORDS (ROWS) IN A TABLE HAVE A COMMON FIELD (COLUMN) STRUCTURE AND ORDER
- RECORDS (ROWS) ARE LINKED ACROSS TABLES BY KEY FIELDS

SQL: WORKING WITH OBJECTS

- DATA DEFINITION LANGUAGE (DB OBJECTS)
 - **CREATE** (TABLE, INDEX, VIEW, FUNCTION, ...)
 - **ALTER** (TABLE, INDEX, VIEW, FUNCTION, ...)
 - **DROP** (TABLE, INDEX, VIEW, FUNCTION, ...)

SQL: WORKING WITH ROWS

- QUERY LANGUAGE (RECORDS)
 - SELECT ... FROM ...
 - INSERT INTO ...
 - UPDATE ... SET ...
 - DELETE FROM ...

JSON IN POSTGRESQL

```
CREATE TABLE my_raw_api_data (  
    response      JSON, /* or JSONB */  
    captured_at   TIMESTAMP WITH TIME ZONE  
                NOT NULL  
                DEFAULT CURRENT_TIMESTAMP,  
    captured_by   CHARACTER VARYING  
                NOT NULL  
                DEFAULT CURRENT_USER  
);
```

JSON IN POSTGRESQL

```
/* Note specified column list: default  
   values used for everything else.      */
```

```
COPY my_raw_api_data (response)  
FROM PROGRAM  
   'curl "http://api.usa.gov/jobs/..."';
```

```
SELECT response, captured_at, captured_by  
FROM my_raw_api_data;
```

JSON OPERATORS

```
/* Field 'a' of object j */  
SELECT j->'a' FROM my_table; /* json return */  
SELECT j->>'a' FROM my_table; /* text return */
```

```
/* Second element of list j (zero indexed) */  
SELECT j->1 FROM my_table; /* json return */  
SELECT j->>1 FROM my_table; /* text return */
```

```
/* Walk object j along specified path */  
SELECT j#>'{a,1,b}' FROM my_table; /* json */  
SELECT j#>>'{a,1,b}' FROM my_table; /* text */
```

PARSING JSON

```
/* If response is a list. [ ... ] */
```

```
SELECT json_array_length(response)  
FROM my_raw_api_data;
```

```
SELECT json_array_elements(response)  
FROM my_raw_api_data;
```

PARSING JSON

```
/* If response is an object. { ... } */  
SELECT json_object_keys(response)  
FROM my_raw_api_data;
```

```
SELECT response->'a', response#>'{a,b}'  
FROM my_raw_api_data;
```

```
SELECT response->>'id' AS object_id,  
       (jsonb_each(response)).key AS field_name,  
       (jsonb_each(response)).value AS field_value  
FROM my_raw_api_data;
```

POPULATING TABLES

```
/* If response is an object. { ... } */  
CREATE TABLE my_table (...field list...);  
  
SELECT  
    json_populate_record(NULL::my_table, response)  
FROM my_raw_api_data;  
  
INSERT INTO my_table (...columns...)  
SELECT ...manipulate results of above...  
FROM (...above query...);
```

JSONB KEY EXISTENCE

```
/* Does key 'a' exist in response? */  
SELECT response ? 'a'  
FROM my_raw_api_data;
```

```
/* Do any of the keys listed exist in response? */  
SELECT response ?| array['a','b','c']  
FROM my_raw_api_data;
```

```
/* Do all of the keys listed exist in response? */  
SELECT response ?& array['a','b','c']  
FROM my_raw_api_data;
```

JSONB CONTAINMENT

```
/* Is the specified JSON contained in response? */
```

```
SELECT response @> '{"a":1, "b":2}'::jsonb  
FROM my_raw_api_data;
```

```
SELECT '{"a":1, "b":2}'::jsonb <@ response  
FROM my_raw_api_data;
```


JSONB INDEXING

- FOR THE jsonb DATA TYPE, CONTAINMENT AND EXISTENCE TESTS CAN BE SPED UP CONSIDERABLY VIA THE GIN INDEX TYPES.
- json_ops (DEFAULT): INDEXES EVERY KEY AND VALUE; SUPPORTS @>, ?, ?&, ?| OPS
- json_path_ops: INDEXES HASH OF VALUE AND KEYS LEADING TO IT; ONLY SUPPORTS @>

CAVEATS: USE UTF8

- IF YOU'RE PROCESSING JSON, IT'S BEST TO USE THE UTF8 CHARACTER SET.
- POSTGRESQL (AND RFC 7159) ALLOW UNICODE ESCAPE SEQUENCES OF THE FORM `\uXXXX` (WHERE X IS A HEX DIGIT).
- THE `jsonb` TYPE RESTRICTS UNICODE ESCAPES FOR NON-ASCII CHARACTERS (ABOVE `\u007F`) UNLESS THE DATABASE IS UTF8.

CAVEATS: NULLS

- THE JSON null VALUE DOES NOT HAVE ITS OWN TYPE IN POSTGRESQL (IT'S A TEXT VALUE), AND IS NOT THE SQL NULL TYPE.
- TO SEE THE DIFFERENCE:
 - /* JSON null value */
SELECT json_typeof('null'::json);
 - /* SQL NULL value */
SELECT json_typeof(NULL::json);

RELEVANT DOCS

- IETF RFC 7159
<http://rfc7159.net/rfc7159>
- POSTGRESQL 9.4 DOCUMENTATION
- JSON TYPES
<http://www.postgresql.org/docs/9.4/interactive/datatype-json.html>
- JSON FUNCTIONS AND OPERATORS
<http://www.postgresql.org/docs/9.4/interactive/functions-json.html>

EXAMPLE CODE & DATA

- ALL CODE FOR EXAMPLES IS ON GITHUB AT:
<https://github.com/nihonjinrxs/dwdc-january2015>
- EXAMPLES USE DATA FROM TWO PUBLIC APIS:
 - OPEN WEATHER MAP
<http://openweathermap.org/api>
 - DIGITALGOV JOBS API
<http://search.digitalgov.gov/developer/jobs.html>



RYAN B. HARVEY

[HTTP://DATASCIENTIST.GURU](http://datascientist.guru)
RYAN.B.HARVEY@GMAIL.COM
[@NIHONJINRXS](#)
[+RYAN.B.HARVEY](#)

EMPLOYMENT & AFFILIATIONS*

IT PROJECT MANAGER

OFFICE OF MANAGEMENT AND BUDGET

EXECUTIVE OFFICE OF THE PRESIDENT

DATA SCIENTIST & SOFTWARE ARCHITECT

KITCHOLOGY INC.

RESEARCH AFFILIATE

NORBERT WIENER CENTER FOR HARMONIC ANALYSIS & APPLICATIONS

COLLEGE OF COMPUTER, MATHEMATICAL & NATURAL SCIENCES

UNIVERSITY OF MARYLAND AT COLLEGE PARK

Thank you!
Questions?

* MY REMARKS, PRESENTATION AND PREPARED MATERIALS ARE MY OWN, AND DO NOT REPRESENT THE VIEWS OF MY EMPLOYERS.