

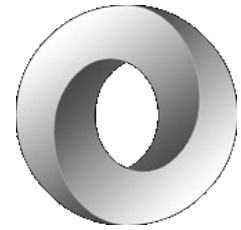
PostgreSQL JSON Features - Heute und in der Zukunft

PGDay Austria
06.09.2019
Stefanie Janine Stölting

mail@stefanie-stoelting.de



JSON



JavaScript Object Notation

Um das Encoding muss man sich nicht kümmern, es ist immer Unicode, die meisten Implementationen benutzen UTF8

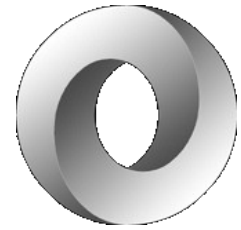
Benutzt für den Datenaustausch in Web Applikationen

Momentan gibt es zwei Standards [RFC 7159](#) von Douglas Crockford und [ECMA-404](#)

PostgreSQL Implementation ist RFC 7159



ISO SQL/JSON Standard

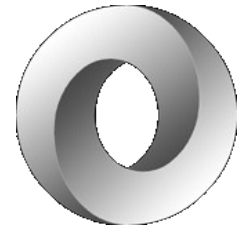


Im März 2017 wurde ein Standard
veröffentlicht: [ISO/IEC TR 19075-6:2017](#)

Zusätzlich frei verfügbar: [ISO/IEC TR 19075-6](#)



JSON Datatypes



JSON

Verfügbar seit Version 9.2

BSON

Als Erweiterung verfügbar auf GitHub seit 2013

JSONB

Verfügbar seit Version 9.4

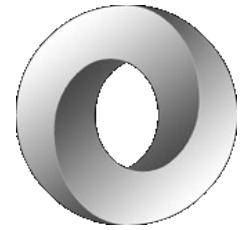
Komprimiertes JSON

Voll Transaktional

Bis zu 1 GB (benutzt **TOAST**)



JSON Funktionen



`row_to_json({row})`

Gibt eine Zeile als JSON zurück

`array_to_json({array})`

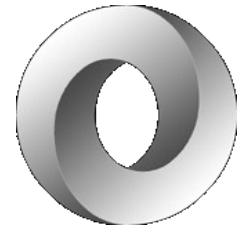
Gibt ein Array als JSON zurück

`jsonb_to_recordset`

Gibt ein Recordset aus JSONB zurück



JSON Operatoren



Array element

->{int}

Array element by name

->{text}

Object element

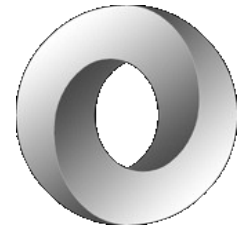
->> {text}

Value at path

#> {text}



Index auf JSON



Indexierung von JSONB Content für schnelleren Zugriff

GIN Index

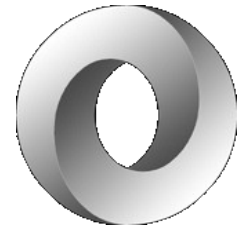
```
CREATE INDEX idx_1 ON jsonb.actor USING  
GIN (jsondata);
```

Sogar Unique **B-Tree** Indizes sind möglich

```
CREATE UNIQUE INDEX actor_id_2 ON  
jsonb.actor((CAST(jsondata->>'actor_id' AS  
INTEGER)));
```



Neue JSON Funktionen



PostgreSQL 9.5 Funktionen:

jsonb_pretty: Formatiert JSONB human readable

jsonb_set: Ändern oder Hinzufügen von Werten

PostgreSQL 9.5 JSONB Operatoren:

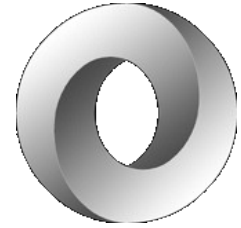
||: Verbindet zwei JSONB

-: Löscht Schlüssel

Als Extension verfügbar für 9.4 über PGXN: [jsonbx](#)



Neue JSON Funktionen



PostgreSQL 9.6 JSONB Funktion:

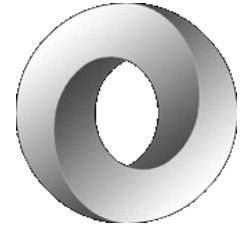
`jsonb_insert`:

Fügt einen neuen Wert in ein JSONB im Pfad ein
und gibt das geänderte JSONB zurück

Details hierzu siehe 9.6 JSONB [Dokumentation](#)



Neue JSON Funktionen



PostgreSQL 10 JSON/JSONB Funktionen:

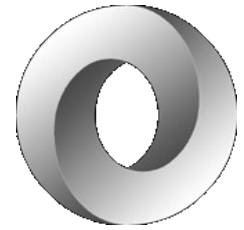
Text Suche:

Suche in JSON/JSONB mit tsvector.

Siehe *Text Search Functions and Operators*
[*Dokumentation*](#) für Details.



Neue JSON Funktionen

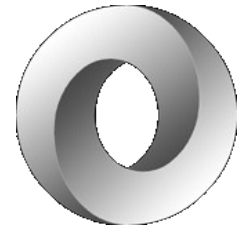


PostgreSQL 11 JSON/JSONB Funktionen:

Erweiterung von tsvector, siehe [Dokumentation](#) für Details.



Neue JSON Funktionen



PostgreSQL 12 JSON/JSONB Funktionen:

Einführung von SQL/JSON Path Ausdrücke, die Bestandteil des JSON Standards unter JSONPath sind.

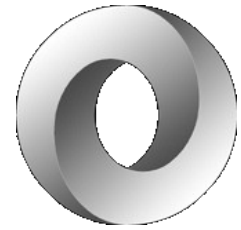
Die PostgreSQL Implementation ist `jsonb_path_query`. Sie ist nur für Felder vom Typ JSONB verfügbar.

Die Implementation von JSONPath ist noch nicht vollständig!

Siehe [Dokumentation](#) und den Blog von Hubert Lubaczewski, [depesz](#), für Details.



Extension JQuery

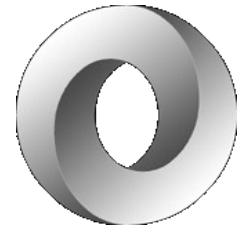


JsQuery ist eine JSON Abfragesprache mit GIN Index Unterstützung.

Sollte mit PostgreSQL 12 nicht mehr genutzt werden, da es nun die **SQL/JSON Path Ausdrücke** unterstützt werden und nicht weiterentwickelt wird (**Talk von Oleg Bartunov, ab Minute 49 über JSON** , PGDay Israel 2019).



Datenquellen

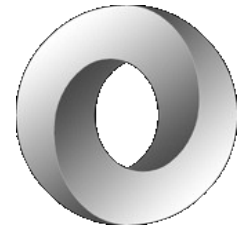


Die Chinook Datenbank ist auf [GitHub](#) verfügbar.

Amazon book reviews von 1998 sind von [Citrus](#) verfügbar.



Chinook Tabellen



	T tablename
1	Artist
2	Invoice
3	Employee
4	Customer
5	Playlist
6	InvoiceLine
7	Album
8	Genre
9	PlaylistTrack
10	MediaType
11	Track

	T table_name	T column_name	T data_type
1	Artist	ArtistId	integer
2	Artist	Name	character varying (120)

	T table_name	T column_name	T data_type
1	Album	AlbumId	integer
2	Album	Title	character varying (160)
3	Album	ArtistId	integer

	T table_name	T column_name	T data_type
1	Track	TrackId	integer
2	Track	Name	character varying (200)
3	Track	AlbumId	integer
4	Track	MediaTypeId	integer
5	Track	GenreId	integer
6	Track	Composer	character varying (220)
7	Track	Milliseconds	integer
8	Track	Bytes	integer
9	Track	UnitPrice	numeric



CTE

Common Table Expressions werden in den Beispielen benutzt

- Beispiel:

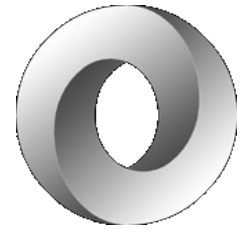
```
WITH RECURSIVE t(n) AS (  
    VALUES (1)  
    UNION ALL  
    SELECT n+1 FROM t WHERE n < 100  
)  
SELECT sum(n), min(n), max(n) FROM t;
```

- Ergebnis:

	sum bigint	min integer	max integer
1	5050	1	100



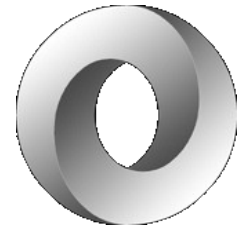
Live Beispiele



Let's see, how it does work.



JSON by example



This document by [Stefanie Janine Stölting](#) is covered by the [Creative Commons Attribution 4.0 International](#)

Ihr findet die Slides und den Quellcode zum Vortrag auf GitLab:

<https://gitlab.com/sjstoelting/talks/tree/master/PostgreSQL-JSON-Features-Current-And-Future-PGDay-Austria-2019>