1. Przetwarzanie danych typ JSON w bazie PostgreSQL.

Typ JSON i JSONB przechowuje dane w formacie JSON zgodnie ze specyfikacją RFC 7159. Dane przechowywane są jak tekst ale muszą spełniać warunki tego formatu. Postgresql proponuje dwa typy danych json i jsonb. Pierwszy przechowuje dane bez ingerencji, drugi przetwarza dane wejściowe do postaci bardziej optymalnej dla zapytań typu Select. Typ ten dostarcza także pokaźnego zbioru operatorów i funkcji.

W celu zaprezentowania możliwości tego typu prześledźmy działanie przykładów.

```
DROP TABLE IF EXISTS person_sample_table;

CREATE TABLE person_sample_table (
fname varchar,
lname varchar,
city varchar,
street VARCHAR,
address VARCHAR,
email varchar,
phone varchar,
id varchar ) ;

INSERT INTO person_sample_table VALUES
( 'Adam', 'Abacki', 'Krakow', 'Mickiewicza', '87m3', 'abacki@o2.pl',
'123 400 212', '10001'),
( 'Marta', 'Babacka', 'Konin', 'Lipowa', '19', 'babacki@onet.pl',
'411 909 123', '11011'),
( 'Bogdan', 'Cabacki', 'Tarnow', 'Lubelska', '34m4','cabacki@wp.pl',
'222 190 100', '20010'),
( 'Marek', 'Dadacki', 'Gdynia', 'Portowa', '146',
'dadacki@google.com', '431 222 100', '10022'),
( 'Edward', 'Kabacki', 'Szczecin', 'Kasztanowa', '8a',
'kabacki@o2.pl', '780 159 100', '21145') ;

SELECT * FROM person_sample_table;

DROP TABLE IF EXISTS json_table;
CREATE TABLE json_table ( id varchar, person JSONB ) ;

INSERT INTO json_table
WITH flat2json ( person ) AS
 ( SELECT row_to_json ( row, true )
   FROM ( SELECT * FROM person_sample_table ) row )
SELECT row.person::json->>'id' AS id, row_to_json ( row, true)
```

```
FROM ( SELECT * FROM flat2json ) row ;

SELECT * FROM json_table;

SELECT id, jsonb_pretty(person) FROM json_table ;
```

Operatory.

```
SELECT id, person->'person'->'email' AS mail FROM json_table ;
SELECT id, person->'person'->>'email' AS mail FROM json_table ;

SELECT id, (person->'person'->'email') || '-' ||  (person->'person'-
>'phone') AS contact FROM json_table ;
SELECT id, (person->'person'->>'email') || '-' || (person->'person'-
>>'phone') AS contact FROM json_table ;

SELECT id, person::json#>('{person,email}') AS mail FROM json_table ;
SELECT id, person::json#>>('{person,street}') AS ulica FROM json_table ;

SELECT id, jsonb_pretty( person #- '{person,phone}') FROM json_table LIMIT
1;

SELECT id, person->'person'->>'email' FROM json_table
WHERE person->'person'->>'id' = '10022';

SELECT id, person->'person'->>'fname' AS mail FROM json_table
WHERE person->'person' @> '{"city":"Gdynia"}'::jsonb;

SELECT id, person->'person'->>'fname' AS mail FROM json_table
WHERE '{"city":"Gdynia"}'::jsonb <@ (person->'person');
```

Operacje UPDATE i INSERT.

```
UPDATE json_table
       SET person = jsonb_set ( person, '{person,midname}', '"Karol"',
true)
       WHERE  id = '10022' ;
SELECT id, jsonb_pretty(person) FROM json_table  WHERE  id = '10022';

UPDATE json_table
       SET person = jsonb_set ( person, '{person,midname}', '"Jan"', true)
       WHERE  id = '10022' ;
SELECT id, jsonb_pretty(person) FROM json_table  WHERE  id = '10022';

UPDATE json_table
       SET person = jsonb_insert ( person, '{person,midname}', '"Michal"',
false)
       WHERE  id = '10001' ;
SELECT id, jsonb_pretty(person) FROM json_table  WHERE  id = '10001';
```

Funkcje przetwarzające dane między strukturami relacyjną i JSON.

```
CREATE TYPE personrecord AS ( "id" varchar, "fname" varchar, "lname"
varchar, "email" varchar, "city" VARCHAR, "street" varchar);

SELECT ( jsonb_populate_record(null::personrecord, person->'person')).*
FROM json_table;
SELECT id, jsonb_pretty(person) FROM json_table;
```

Przekształcenie danych z tabeli relacyjnej do postaci obiektu JSON zapewnia funkcja jsonb_build_object() i jsonb_build_array().

```
SELECT jsonb_pretty( jsonb_build_object( 'person_record',
json_build_object('personid',id,'firstname',fname,'lastnamename',lname)))
FROM person_sample_table WHERE id = '10001' ;


SELECT jsonb_pretty( jsonb_build_object( 'person_record',
json_build_object('pid',id,'firstname',fname,'lastname',lname,'email',
                                        json_build_array(email))
                                        ))
FROM person_sample_table WHERE  id = '10022';

SELECT jsonb_pretty( jsonb_build_object( 'person_record',

json_build_object('pid',id,'firstname',fname,'lastname',lname,
                     'contact', json_build_object('email',email,
'phone',phone), 'personaddress', json_build_object('street',street,
'address',address)) ))
FROM person_sample_table WHERE  id = '11011';
```

Do kontroli typu danych w obiektch JSON można wykorzystać funkcję jsonb_typeof().

```
SELECT jsonb_typeof(person) AS persondata,
  jsonb_typeof(person->'person') AS person,
  jsonb_typeof(person->'person'->'id') AS id,
  jsonb_typeof(person->'person'->'lname') AS lastname
FROM json_table;
```

Odnośniki.

9.15. JSON Functions and Operators
PostgreSQL JSON
The unofficial guide to JSONB operators in Postgres - Part 1