

ЗАЩИЩЕННАЯ СИСТЕМА УПРАВЛЕНИЯ
БАЗАМИ ДАННЫХ «ЈАТОВА»

Руководство по настройке. Часть 19.
Формирование HTTP/HTTPS запросов из СУБД
Компонент «pgSQL-HTTP»

643.72410666.00067-07 98 01-19

Листов 28

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

АННОТАЦИЯ

В документе приведены сведения, необходимые для установки и эксплуатации компонента «Формирование HTTP/HTTPS запросов из СУБД "pgSQL-HTTP" (далее по тексту – «компонент»).

Настоящее руководство предназначено для администраторов СУБД.



Все примеры в данном документе приведены для СУБД «Jatoba» версии ядра 4.x, для других версий все шаги выполняются аналогично, разница состоит в именах директорий.

Например, СУБД «Jatoba» версии 6.x по умолчанию устанавливается в директорию ОС Linux – «/usr/jatoba-6/bin».

Для СУБД «Jatoba» версии ядра 4 используется версия компонента — 1.5.

Для СУБД «Jatoba» версии ядра 5/6/18 используется версия компонента — 1.6.



Важная информация

Для сертифицированной версии СУБД «Jatoba» поддерживается работа только на ОС, указанных в формуляре на поставку!

Степени важности примечаний, применяемые в документе:



Важная информация – указания, требующие особого внимания



Дополнительная информация – указания, позволяющие упростить работу с изделием

СОДЕРЖАНИЕ

1. Назначение компонента.....	4
1.1. Условия применения.....	4
2. Установка.....	5
2.1. Установка компонента ОС GNU/Linux	5
2.2. Установка расширения компонента	6
3. Функциональные возможности	8
3.1. Функция «http_header».....	8
3.2. Функция «http».....	9
3.3. Функция «http_get»	10
3.3.1. IP-адрес веб-ресурса.....	11
3.3.2. Тип контента.....	12
3.3.3. Формат изображений.....	12
3.3.4. HTTP заголовок в табличном формате	14
3.4. Функция «http_post».....	15
3.5. Функция «http_put»	17
3.6. Функция «http_patch».....	18
3.7. Функция «http_delete».....	19
3.8. Функция «http_head».....	20
3.9. Функция «curl_opt»	21
3.9.1. Функция «http_set_curl_opt»	22
3.9.2. Функция «http_reset_curl_opt»	23
3.9.3. Функция «http_list_curl_opt».....	23
3.10. Функция «url_encode».....	24
3.10.1. Кодирование строки	24
3.10.2. Кодирование ассоциативного массива JSON	25
4. Удаление компонента	26
4.1. Удаление компонента при отсутствии зависимых от него объектов	26
4.2. Удаление компонента при наличии зависимых от него объектов.....	26
4.3. Удаление пакета	26
Перечень сокращений.....	27

1. НАЗНАЧЕНИЕ КОМПОНЕНТА

Компонент "pgSQL-HTTP" предназначен для выполнения запросов по протоколам HTTP и HTTPS с веб-ресурсов.



Компонент "pgSQL-HTTP" не реализует функции безопасности СУБД согласно нормативной документации ФСТЭК России

1.1. Условия применения

Компонент "pgSQL-HTTP" может использоваться совместно с СУБД «Jatoba» версий 4.x и выше под управлением ОС GNU/Linux.



В текущей реализации компонента не поддерживается управление через компонент пользовательского веб-интерфейса для администраторов «Jatoba data safe».

Ограничений по совместимости с другими компонентами нет.

2. УСТАНОВКА

Компонент функционирует под управлением ОС семейства GNU Linux. Установка компонента должна производиться от имени пользователя, обладающего административными привилегиями в системе.

2.1. Установка компонента ОС GNU/Linux

Компонент устанавливается в составе СУБД «Jatoba». Его возможно установить при первичной установке либо доустановить.

Установку компонента возможно провести двумя способами:

- 1) установка из локального репозитория (CDROM) – производится из файлов, записанных на компакт-диск или скопированных с него;
- 2) установка непосредственно из deb/rpm-файлов – производится опционально, по усмотрению пользователя.

Компонент выполнен в виде отдельного deb или rpm-пакета. Установка компонента осуществляется средствами пакетного менеджера ОС. Для разных типов пакетных менеджеров команда установки немного отличается. Ниже приведены основные типы:

– для систем на основе пакетного менеджера APT (к таким системам относятся все ОС семейства Debian, использующие deb-пакеты) команда установки следующая:

```
apt-get install jatoba18-pgsql-http
```

– для систем на основе пакетных менеджеров YUM/DNF (к таким системам относятся все ОС семейства RedHat и вышедшие из нее, использующие rpm-пакеты) команда установки следующая:

```
yum install jatoba18-pgsql-http
```

Отдельного уточнения требуют операционные системы ALT Linux и openSUSE.

– ALT Linux использует пакетный менеджер APT, но распространяется в виде rpm-пакетов и для нее команда установки выглядит аналогично Debian:

```
apt-get install jatoba18-pgsql-http
```

Установка компонента в составе других версий СУБД «Jatoba» осуществляется аналогично. Отличие будет только в номере версии СУБД, в составе которой он распространяется. Например, jatoba18-pgsql-http и т.п.

Удаление модуля также осуществляется средствами пакетного менеджера ОС. Вместо команды `install` нужно использовать соответствующую данному пакетному менеджеру команду удаления (`remove`, `purge`, `erase` и т.п.).

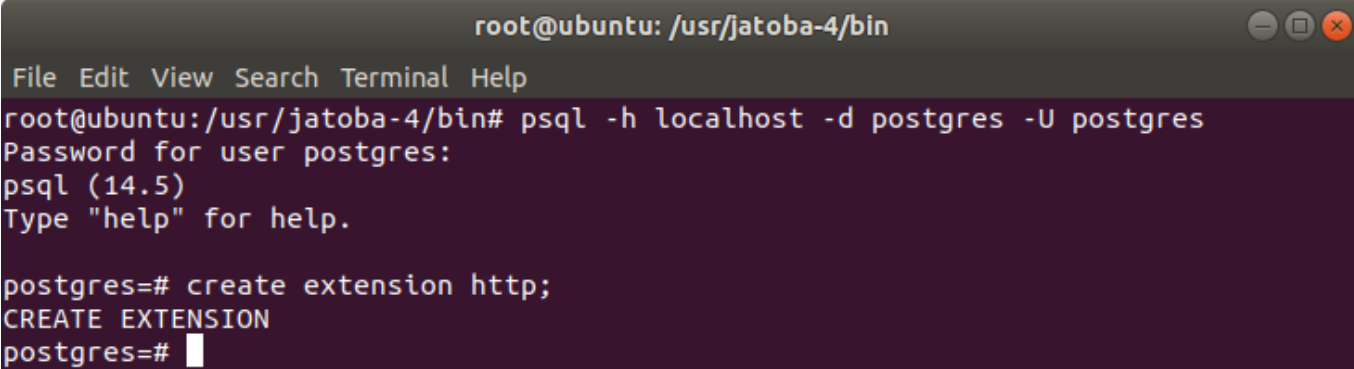
Для получения детальной информации по пакетному менеджеру рекомендуется обратиться к документации по ОС.

2.2. Установка расширения компонента

Предварительной настройки конфигурационного файла `postgresql.conf` не требуется.

Расширение устанавливается на при помощи SQL-команды (рисунок 2.1).

```
CREATE EXTENSION http;
```



```
root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
root@ubuntu:/usr/jatoba-4/bin# psql -h localhost -d postgres -U postgres
Password for user postgres:
psql (14.5)
Type "help" for help.

postgres=# create extension http;
CREATE EXTENSION
postgres=#
```

Рисунок 2.1 – Команда установки расширения

В результате выполнения SQL-команды будет создано расширение (EXTENSION) «http».

Убедиться в установке расширения возможно при помощи SQL-команды:

```
\dx
```

```
root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# \dx
                                List of installed extensions
  Name      | Version | Schema  | Description
-----+-----+-----+-----
 http       | 1.5     | public  | HTTP client for PostgreSQL, allows web page re
trieval inside the database.
 plpgsql    | 1.0     | pg_catalog | PL/pgSQL procedural language
(2 rows)

postgres=#
```

Рисунок 2.2 – Список установленных расширений

3. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ

3.1. Функция «http_header»

Функция «http_header» используется для составления заголовка запроса.

Функция используется с синтаксисом SQL-запросов:

```
http_header(field VARCHAR, value VARCHAR)
```

Применяется с параметрами, приведенными в таблице 3.1.

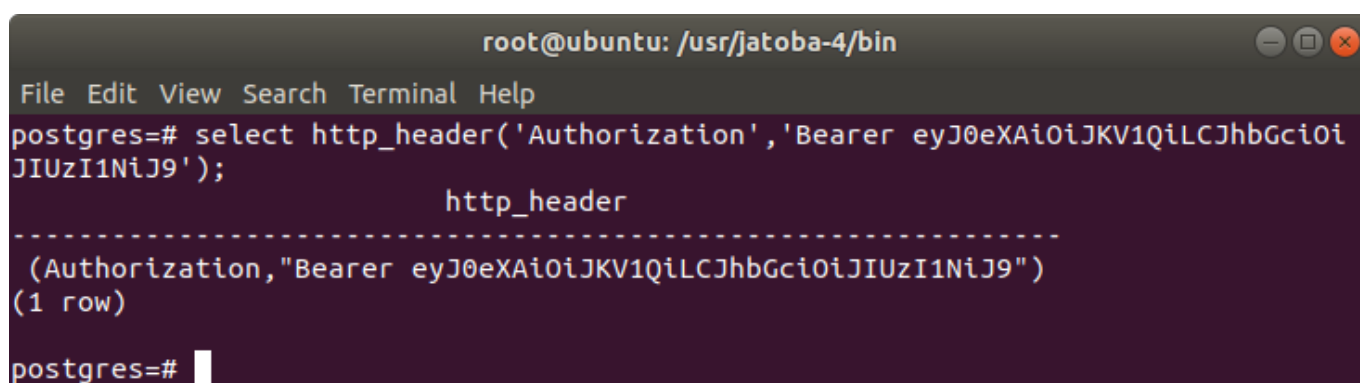
Таблица 3.1 – Параметры функции «http_header»

Параметр	Тип данных	Обозначение
http_header(field VARCHAR, value VARCHAR)		
#1 field	varchar	Имя поля заголовка
#2 value	varchar	Значение поля заголовка

Пример

```
select http_header('Authorization','Bearer eyJ0eXAI0iJKV1QiLCJhbGciOiJIUzI1NiJ9');
```

В представленном примере функция составляет заголовок для авторизации с использованием имени поля «Authorization» и значением поля (рисунок 3.1)



```
root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# select http_header('Authorization','Bearer eyJ0eXAI0iJKV1QiLCJhbGciOiJIUzI1NiJ9');
               http_header
-----
 (Authorization,"Bearer eyJ0eXAI0iJKV1QiLCJhbGciOiJIUzI1NiJ9")
(1 row)
postgres=#
```

Рисунок 3.1 – SQL-запрос http_header

3.2. Функция «http»

Функция «http» используется для создания и выполнения http запроса.

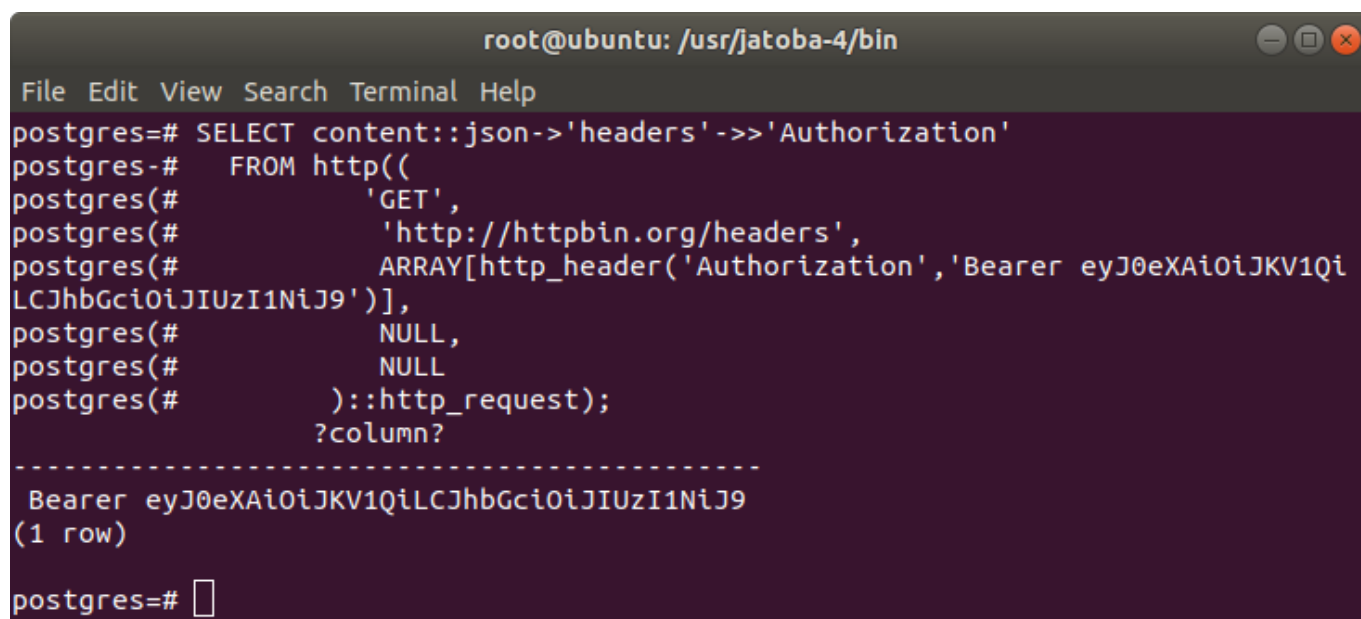
Функция используется с синтаксисом SQL-запросов:

```
http(request http_request)
```

Пример

```
SELECT content::json->'headers'->>'Authorization'
FROM http((
    'GET',
    'http://httpbin.org/headers',
    ARRAY[http_header('Authorization','Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9')],
    NULL,
    NULL
)::http_request);
```

В представленном запросе отсылается запрос на авторизацию пользователя на указанном веб-ресурсе и разбирается ответ в формате JSON.



```
root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# SELECT content::json->'headers'->>'Authorization'
postgres=# FROM http((
postgres=# 'GET',
postgres=# 'http://httpbin.org/headers',
postgres=# ARRAY[http_header('Authorization','Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9')],
postgres=# NULL,
postgres=# NULL
postgres=# )::http_request);
?column?
-----
Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
(1 row)
postgres=#
```

Рисунок 3.2 – SQL-запрос http

3.3. Функция «http_get»

Функция «http_get» используется для получения информации о веб-ресурсе.

Функция используется с синтаксисом SQL-запросов:

```
http_get(uri VARCHAR)
http_get(uri VARCHAR, data JSONB)
```

Функция позволяет получать данные:

- IP-адрес веб-ресурса (п. 3.3.1);
- Тип контента (п. 3.3.2);
- Формат изображений (п. 3.3.3);
- HTTP заголовок в табличном формате (п. 3.3.4).

Применяется с параметрами, приведенными в таблице 3.2.

Таблица 3.2 – Параметры функции «http_get»

Параметр	Тип данных	Обозначение
http_get(uri VARCHAR)		
#1	varchar	Адрес веб-ресурса
http_get(uri VARCHAR, data JSONB)		
#1	varchar	Адрес веб-ресурса
#2	JSONB	Аргумент данных

Пример

В качестве ярлыка для отправки данных в запрос GET передан аргумент данных JSONB.

```
SELECT status, content::json->'args' AS args
FROM http_get('http://httpbin.org/get',
              jsonb_build_object('myvar', 'myval', 'foo', 'bar'));
```

```

root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# SELECT status, content::json->'args' AS args
postgres-# FROM http_get('http://httpbin.org/get',
postgres-# jsonb_build_object('myvar','myval','foo','bar'));
 status |
-----+-----
      200 | {
          |   "foo": "bar",
          |   "myvar": "myval"+
          | }
(1 row)

postgres=#

```

Рисунок 3.3 – SQL-запрос GET с передачей аргумента данных JSONB

3.3.1. IP-адрес веб-ресурса

Функция «http_get» может использоваться для определения IP-адреса веб-ресурса с применением аргумента – «IP».

```
http_get(uri VARCHAR/ip)
```

Пример

Для выяснения IP-адреса ресурса «http://httpbin.org» формируется следующая SQL-команда:

```

SELECT content
FROM http_get('http://httpbin.org/ip');

```

```

root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# SELECT content
postgres-# FROM http_get('http://httpbin.org/ip');
 content
-----+-----
 {
  "origin": "185.61.252.7"+
 }
(1 row)

postgres=#

```

Рисунок 3.4 – Пример запроса IP-адреса

3.3.2. Тип контента

Функция «http_get» может использоваться для определения типа контента веб-ресурса. В этом случае составляется SQL-запрос:

```
SELECT status, content_type  
FROM http_get('http://httpbin.org/');
```

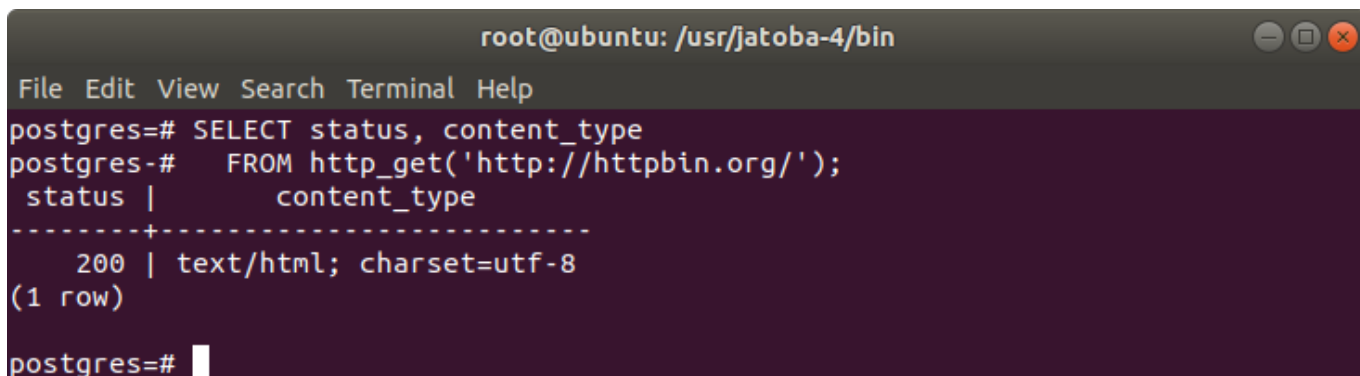


Рисунок 3.5 – Вывод типа контента веб-ресурса

В результате будет выведена информация о типе текста и кодировке.

3.3.3. Формат изображений

Функция «http_get» может применяться для получения данных об изображениях на веб-ресурсе. И используется синтаксис SQL-команд:

```
http_get(uri VARCHAR/image/[type])
```

функция использует операторы:

- /image

Запрос возвращает данные о полях заголовков HTTP

```
http_get(uri VARCHAR/image)
```

- /image/jpeg

Запрос возвращает данные о изображениях в формате JPEG

```
http_get(uri VARCHAR/image/jpeg)
```

- /image/png

Запрос возвращает данные о изображениях в формате PNG

```
http_get(uri VARCHAR/image/png)
```

- /image/svg

Запрос возвращает данные о изображениях в формате SVG

```
http_get(uri VARCHAR/image/svg)
```

- /image/webp

Запрос возвращает данные об изображениях, сжатых в формате WEBP

```
http_get(uri VARCHAR/image/webp)
```

Пример

Ниже рассмотрен запрос сведений об изображениях в формате «SVG».

SQL-запрос будет следующим:

```
WITH
  http AS (
    SELECT * FROM http_get('http://httpbin.org/image/svg')
  ),
  headers AS (
    SELECT (unnest(headers)).* FROM http
  )
SELECT
  http.content_type,
  length(textsend(http.content)) AS length_binary,
  headers.value AS length_headers
FROM http, headers
WHERE field = 'Content-Length';
```

```

root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# WITH
postgres=#   http AS (
postgres=#     SELECT * FROM http_get('http://httpbin.org/image/svg')
postgres=#   ),
postgres=#   headers AS (
postgres=#     SELECT (unnest(headers)).* FROM http
postgres=#   )
postgres=# SELECT
postgres=#   http.content_type,
postgres=#   length(textsend(http.content)) AS length_binary,
postgres=#   headers.value AS length_headers
postgres=# FROM http, headers
postgres=# WHERE field = 'Content-Length';
  content_type | length_binary | length_headers
-----+-----+-----
image/svg+xml |          8984 |          8984
(1 row)

postgres=#

```

Рисунок 3.6 – Пример SQL-запроса «http_get» с аргументом «svg»

3.3.4. HTTP заголовок в табличном формате

Показывает все заголовки HTTP (http_header) в ответе (http_response), используя табличный формат.

Пример

```

SELECT (unnest(headers)).*
FROM http_get('http://httpbin.org/');

```

```

root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# SELECT (unnest(headers)).*
postgres=# FROM http_get('http://httpbin.org/');
  field | value
-----+-----
Date   | Tue, 21 Feb 2023 05:43:24 GMT
Content-Type | text/html; charset=utf-8
Content-Length | 9593
Connection | close
Server | unicorn/19.9.0
Access-Control-Allow-Origin | *
Access-Control-Allow-Credentials | true
(7 rows)

postgres=#

```

Рисунок 3.7 – SQL-запрос получения заголовка в табличном формате

3.4. Функция «http_post»

Функция «http_post» используется для отправки на веб-ресурс данных.

Функция используется с синтаксисом SQL-запросов:

```
http_post(uri VARCHAR, content VARCHAR, content_type VARCHAR)
http_post(uri VARCHAR, data JSONB)
```

Применяется с параметрами, приведенными в таблице 3.3.

Таблица 3.3 – Параметры функции «http_post»

Параметр	Тип данных	Обозначение
http_post(uri VARCHAR, content VARCHAR, content_type VARCHAR)		
#1 uri	varchar	Адрес веб-ресурса
#2 content	varchar	Отправляемый контент
#3 content_type	varchar	Аргумент данных
http_post(uri VARCHAR, data JSONB)		
#1 uri	varchar	Адрес веб-ресурса
#2 data	JSONB	Аргумент данных

Пример

Позволяет выполнить стандартный запрос «post» стандарта HTTP, используя синтаксис SQL:

```
http_post(uri VARCHAR, content VARCHAR, content_type VARCHAR)
```

SQL-запрос может быть следующим:

```
select * from http_post('https://httpbin.org/post',
'param1=value1&param2=value2', 'application/json');
```

В представленном примере отправлен запрос с функцией «http_post»:

- на адрес (url) 'https://httpbin.org/post';
- контент в виде строки, которая может быть любой;
- получили ответ формате JSON.

```

root@ubuntu: /usr/jatoba-4/bin
postgres=# select * from http_post('https://httpbin.org/post', 'param1=value1&param2=value2', 'application/json');
 status | content_type | headers
-----+-----+-----
      200 | application/json | [{"Date":"Tue, 21 Feb 2023 06:36:27 GMT"},"(Content-Type,application/json)","(Content-Length,546)","(Connection,close)","(Server,gunicorn/19.9.0)","(Access-Control-Allow-Origin,*)","(Access-Control-Allow-Credentials,true)"}] [{"args": {}, "data": "param1=value1&param2=value2", "files": {}, "form": {}, "headers": {"Accept": "*/*", "Accept-Encoding": "deflate, gzip",

```

Рисунок 3.8 – SQL-запрос http_post

Пример

Чтобы выполнить POST для URL-адреса с использованием полезных данных вместо параметров, встроенных в URL-адрес, закодируйте данные в JSONB как полезные данные.

```
http_post(uri VARCHAR, data JSONB)
```

SQL-запрос может быть следующим:

```

SELECT status, content::json->'form' AS form
FROM http_post('http://httpbin.org/post',
jsonb_build_object('myvar','myval','foo','bar'));

```

```

root@ubuntu: /usr/jatoba-4/bin
postgres=# SELECT status, content::json->'form' AS form
postgres=# FROM http_post('http://httpbin.org/post',
postgres=# jsonb_build_object('myvar','myval','foo','bar'));
 status | form
-----+-----
      200 | {"foo": "bar", "myvar": "myval"}
(1 row)

postgres=#

```

Рисунок 3.9 – SQL-запрос http_post с кодировкой данных в формате JSONB

3.5. Функция «http_put»

Функция «http_put» используется для отправки простого документа на веб-сервер.

Функция используется с синтаксисом SQL-запросов:

```
http_put(uri VARCHAR, content VARCHAR, content_type VARCHAR)
```

И применяется с параметрами, приведенными в таблице 3.4.

Таблица 3.4 – Параметры функции «http_put»

Параметр	Тип данных	Обозначение
http_put(uri VARCHAR, content VARCHAR, content_type VARCHAR)		
#1 uri	varchar	Адрес веб-ресурса
#2 content	varchar	Отправляемый контент
#3 content_type	varchar	Аргумент данных

Пример

Для отправки простого документа на веб-сервер формируется SQL-команда:

```
SELECT status, content_type, content::json->>'data' AS data
FROM http_put('http://httpbin.org/put', 'some text',
'text/plain');
```

```

root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# SELECT status, content_type, content::json->>'data' AS data
postgres=# FROM http_put('http://httpbin.org/put', 'some text', 'text/plain');
 status | content_type | data
-----+-----+-----
    200 | application/json | some text
(1 row)

postgres=#
postgres=#

```

Рисунок 3.10 – Отправка простого документа

3.6. Функция «http_patch»

Функция «http_patch» используется для отправки простого документа JSON на веб-сервер.

Функция используется с синтаксисом SQL-запросов:

```
http_patch(uri VARCHAR, content VARCHAR, content_type VARCHAR)
```

И применяется с параметрами, приведенными в таблице 3.5.

Таблица 3.5 – Параметры функции «http_patch»

Параметр	Тип данных	Обозначение
http_patch(uri VARCHAR, content VARCHAR, content_type VARCHAR)		
#1 uri	varchar	Адрес веб-ресурса
#2 content	varchar	Отправляемый контент
#3 content_type	varchar	Аргумент данных

Пример

Для отправки простого документа JSON на веб-сервер формируется SQL-команда:

```
SELECT status, content_type, content::json->>'data' AS data
FROM http_patch('http://httpbin.org/patch',
'{"this":"that"}', 'application/json');
```

```

root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# SELECT status, content_type, content::json->>'data' AS data
postgres=# FROM http_patch('http://httpbin.org/patch', '{"this":"that"}', 'app
lication/json');
 status | content_type | data
-----+-----+-----
    200 | application/json | {"this":"that"}
(1 row)

postgres=#

```

Рисунок 3.11 – Отправка простого документа JSON

3.7. Функция «http_delete»

Функция «http_delete» используется для запроса удаления ресурса на HTTP-сервере.

Функция используется с синтаксисом SQL-запросов:

```
http_delete(uri VARCHAR, content VARCHAR, content_type VARCHAR)
```

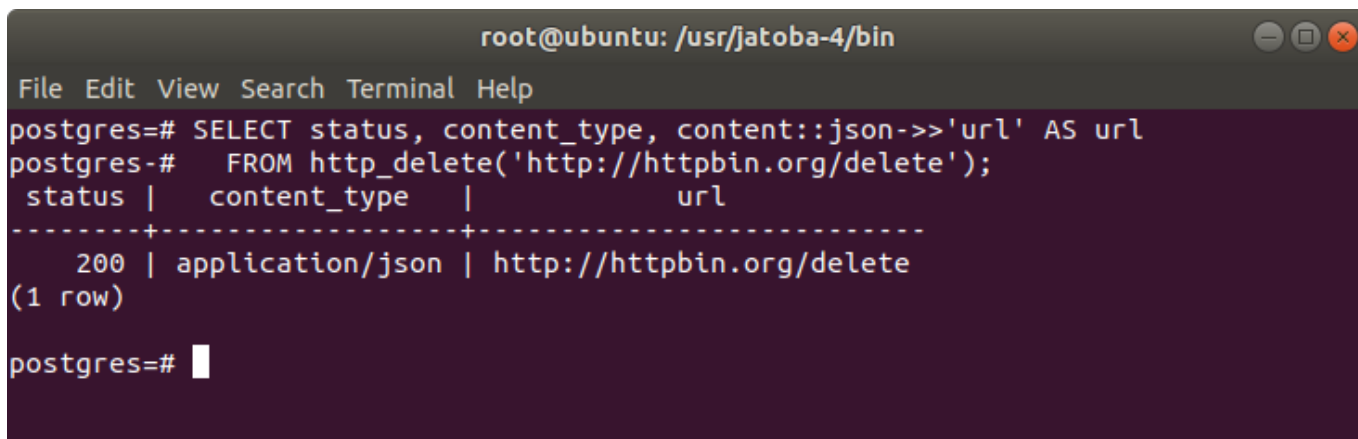
Применяется с параметрами, приведенными в таблице 3.6.

Таблица 3.6 – Параметры функции «http_delete»

Параметр	Тип данных	Обозначение
http_delete(uri VARCHAR, content VARCHAR, content_type VARCHAR)		
#1 uri	varchar	Адрес веб-ресурса
#2 content	varchar	Отправляемый контент
#3 content_type	varchar	Аргумент данных

Пример

```
SELECT status, content_type, content::json->>'url' AS url  
FROM http_delete('http://httpbin.org/delete');
```



```
root@ubuntu: /usr/jatoba-4/bin  
File Edit View Search Terminal Help  
postgres=# SELECT status, content_type, content::json->>'url' AS url  
postgres=# FROM http_delete('http://httpbin.org/delete');  
status | content_type | url  
-----+-----+-----  
200 | application/json | http://httpbin.org/delete  
(1 row)  
postgres=#
```

Рисунок 3.12 – Запрос удаления ресурса

3.8. Функция «http_head»

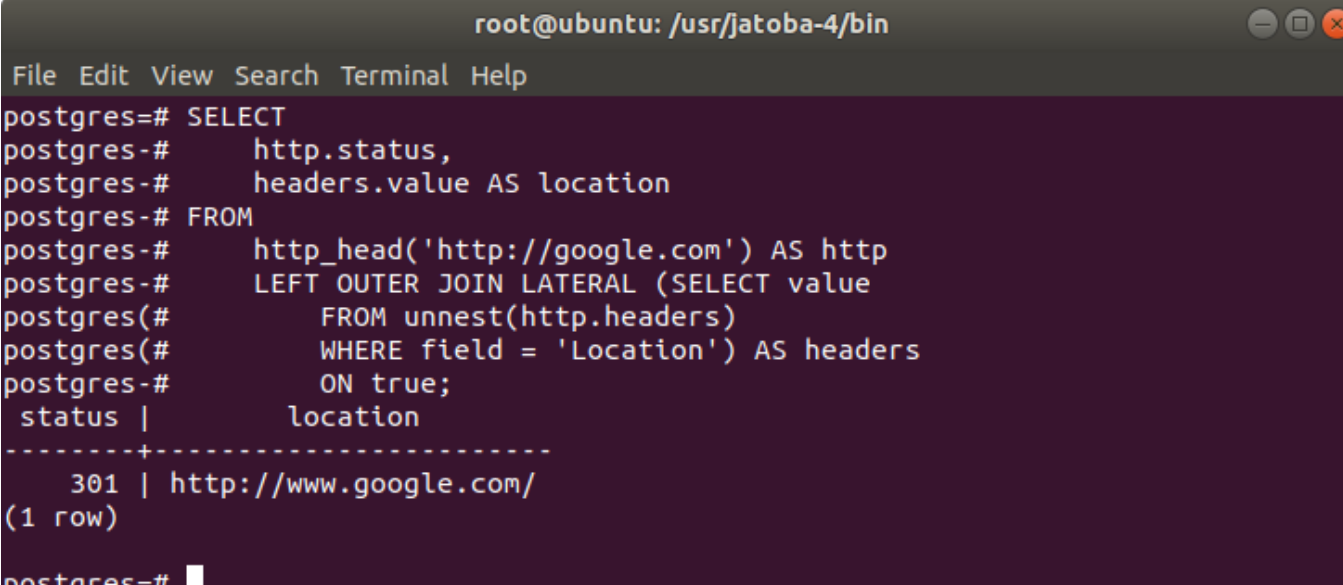
Функция «http_head» используется для получения заголовка запроса и принятия решения о дальнейших действиях.

Аналогично «http_get» функция «http_head» возвращает заголовок:

```
http_head(uri VARCHAR)
```

Пример

```
SELECT
    http.status,
    headers.value AS location
FROM
    http_head('http://google.com') AS http
LEFT OUTER JOIN LATERAL (SELECT value
    FROM unnest(http.headers)
    WHERE field = 'Location') AS headers
ON true;
```



The screenshot shows a terminal window titled 'root@ubuntu: /usr/jatoba-4/bin'. The terminal contains a PostgreSQL prompt 'postgres=#' followed by the same SQL query as in the example block. The query returns a single row with the status '301' and the location 'http://www.google.com/'. The output is displayed in a table format with headers 'status' and 'location'.

```
root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# SELECT
postgres=#     http.status,
postgres=#     headers.value AS location
postgres=# FROM
postgres=#     http_head('http://google.com') AS http
postgres=#     LEFT OUTER JOIN LATERAL (SELECT value
postgres=#         FROM unnest(http.headers)
postgres=#         WHERE field = 'Location') AS headers
postgres=#     ON true;
 status | location
-----+-----
    301 | http://www.google.com/
(1 row)

postgres=#
```

Рисунок 3.13 – SQL-запрос с функцией «http_head»

3.9. Функция «curl»

Функция «curl» использует SQL-синтаксис:

```
http_set_curl(curl VARCHAR, value varchar)
```

и может использоваться с параметрами:

- CURLOPT_DNS_SERVERS;
- CURLOPT_PROXY;
- CURLOPT_PRE_PROXY;
- CURLOPT_PROXYPORT;
- CURLOPT_PROXYUSERPWD;
- CURLOPT_PROXYUSERNAME;
- CURLOPT_PROXYPASSWORD;
- CURLOPT_PROXY_TLSAUTH_USERNAME;
- CURLOPT_PROXY_TLSAUTH_PASSWORD;
- CURLOPT_PROXY_TLSAUTH_TYPE;
- CURLOPT_TLSAUTH_USERNAME;
- CURLOPT_TLSAUTH_PASSWORD;
- CURLOPT_TLSAUTH_TYPE;
- CURLOPT_SSL_VERIFYHOST;
- CURLOPT_SSL_VERIFYPEER;
- CURLOPT_SSLCERT;
- CURLOPT_SSLKEY;
- CURLOPT_SSLCERTTYPE;
- CURLOPT_CAINFO;
- CURLOPT_TIMEOUT;
- CURLOPT_TIMEOUT_MS;
- CURLOPT_TCP_KEEPALIVE;
- CURLOPT_TCP_KEEPIDLE;
- CURLOPT_CONNECTTIMEOUT;
- CURLOPT_USERAGENT.

3.9.1. Функция «http_set_curlopt»

Функция «http_set_curlopt» может использоваться для установки параметров прокси-порта на время существования соединения с базой данных.

```
http_set_curlopt(curlopt VARCHAR, value varchar)
```

Пример

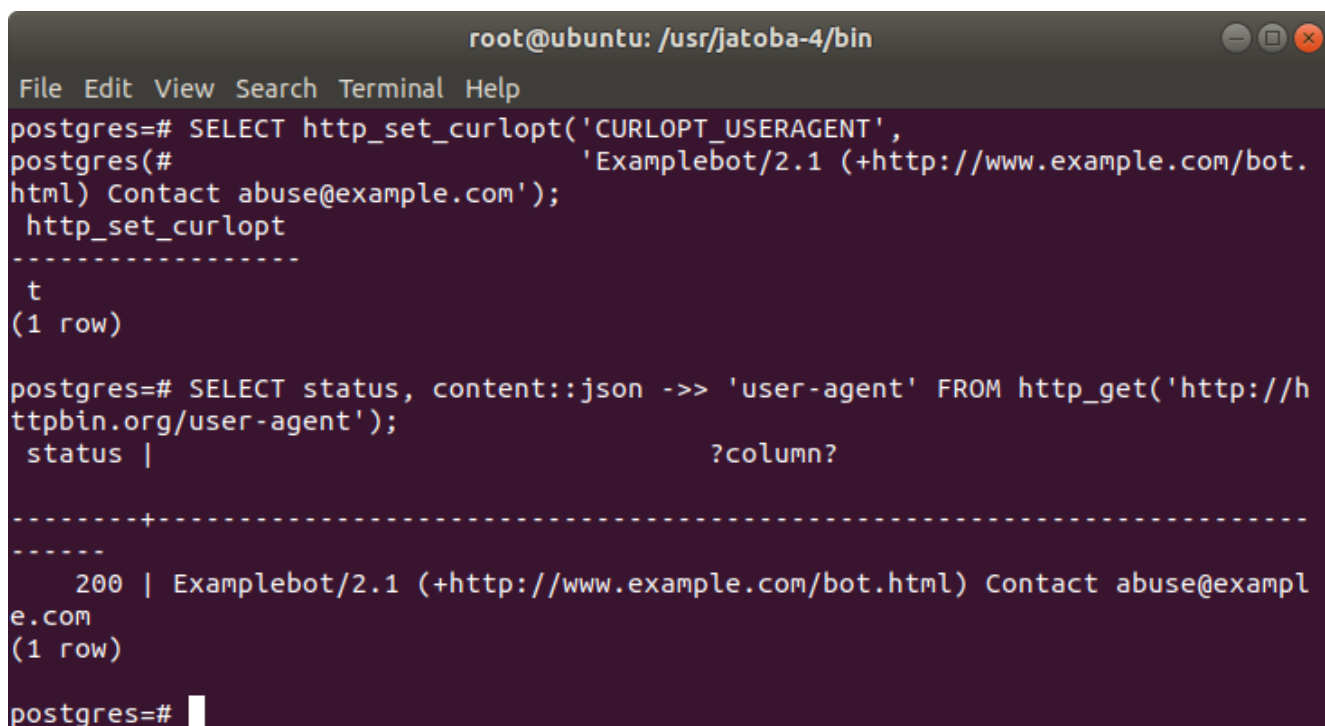
```
SELECT http_set_curlopt('CURLOPT_PROXYPORT', '12345');
```

Политики API могут требовать предоставления определенной контактной информации при каждом запросе. Другие могут запретить определенных агентов, которых они не распознают.

Для таких случаев можно установить CURLOPT_USERAGENT опцию.

Пример

```
SELECT http_set_curlopt('CURLOPT_USERAGENT',  
'Examplebot/2.1 (+http://www.example.com/bot.html) Contact  
abuse@example.com');  
  
SELECT status, content::json ->> 'user-agent' FROM  
http_get('http://httpbin.org/user-agent');
```



```
root@ubuntu: /usr/jatoba-4/bin  
File Edit View Search Terminal Help  
postgres=# SELECT http_set_curlopt('CURLOPT_USERAGENT',  
postgres(# 'Examplebot/2.1 (+http://www.example.com/bot.  
html) Contact abuse@example.com');  
http_set_curlopt  
-----  
t  
(1 row)  
  
postgres=# SELECT status, content::json ->> 'user-agent' FROM http_get('http://h  
ttpbin.org/user-agent');  
status |  
-----+-----  
200 | Examplebot/2.1 (+http://www.example.com/bot.html) Contact abuse@exampl  
e.com  
(1 row)  
  
postgres=#
```

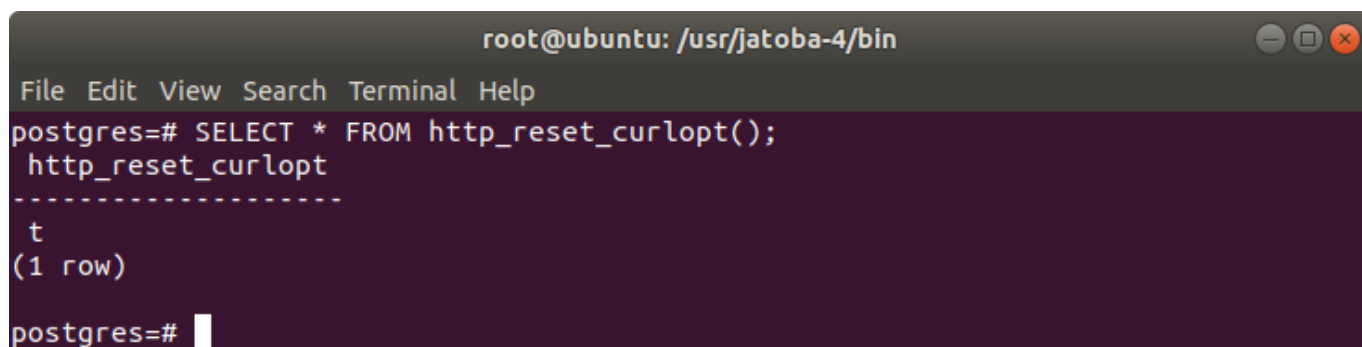
Рисунок 3.14 – Установка CURLOPT_USERAGENT

3.9.2. Функция «http_reset_curl_opt»

Функция «http_reset_curl_opt» может использоваться для сброса всех параметров CURL до значений по умолчанию.

SQL-запрос будет следующим:

```
SELECT * FROM http_reset_curl_opt();
```



```
root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# SELECT * FROM http_reset_curl_opt();
 http_reset_curl_opt
-----
 t
(1 row)
postgres=#
```

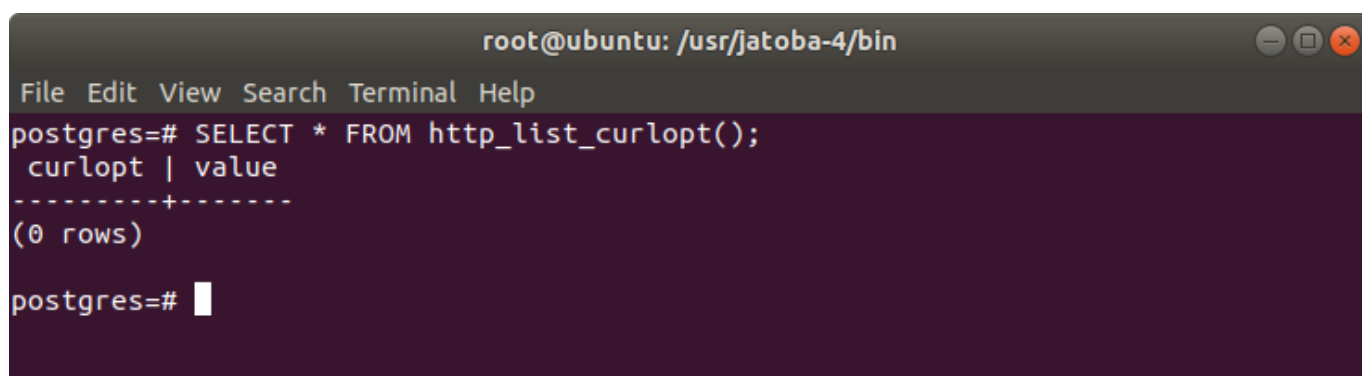
Рисунок 3.15 – SQL-запрос для сбора параметров прокси

3.9.3. Функция «http_list_curl_opt»

Функция «http_list_curl_opt» может использоваться для получения всех установленных параметров.

```
SELECT * FROM http_list_curl_opt();
```

Пример



```
root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# SELECT * FROM http_list_curl_opt();
 curl_opt | value
-----+-----
(0 rows)
postgres=#
```

Рисунок 3.16 – Вывод установленных параметров

3.10. Функция «urlencode»

Функция «urlencode» используется, для:

- кодирования строки (п. 3.10.1);
- кодирования ассоциативного массива JSON (п. 3.10.2).

Используется SQL-синтаксис:

```
urlencode(string VARCHAR)
urlencode(data JSONB)
```

3.10.1. Кодирование строки

Эта функция удобна, когда закодированная строка будет использоваться в запросе, как часть URL, также это удобный способ для передачи переменных другим страницам.

Возвращает строку, в которой все не цифробуквенные символы, кроме (– _.) должны быть заменены знаком процента (%), за которым следует два шестнадцатеричных числа, а пробелы кодируются как знак сложения (+). Строка кодируется тем же способом, что и POST данные WWW-формы, то есть по типу контента application/x-www-form-urlencoded. Это отличается от RFC 3986 кодирования тем, что по историческим соображениям, пробелы кодируются как знак "плюс" (+).

```
SELECT urlencode('my special string's & things?');
```

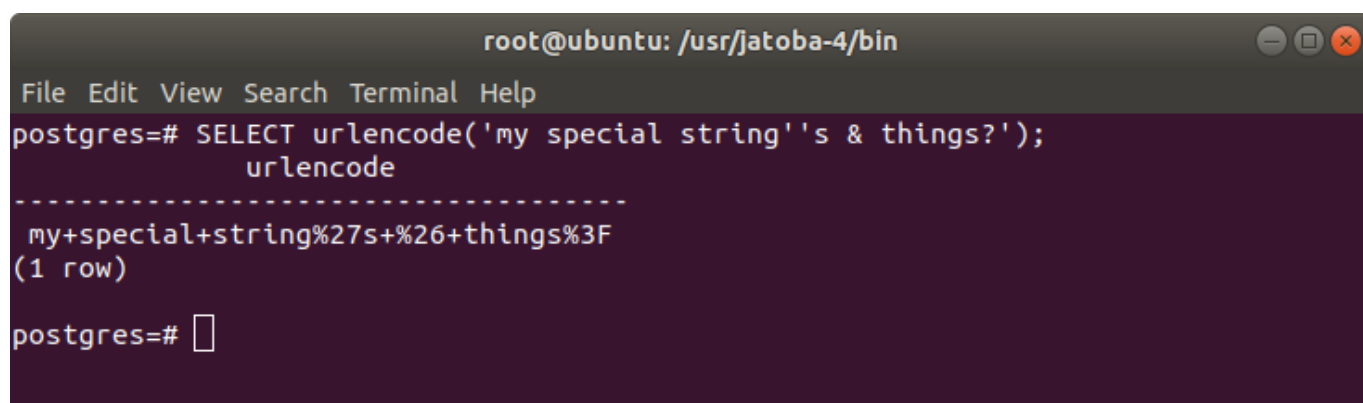


Рисунок 3.17 – SQL-запрос с кодированием строки

3.10.2. Кодирование ассоциативного массива JSON

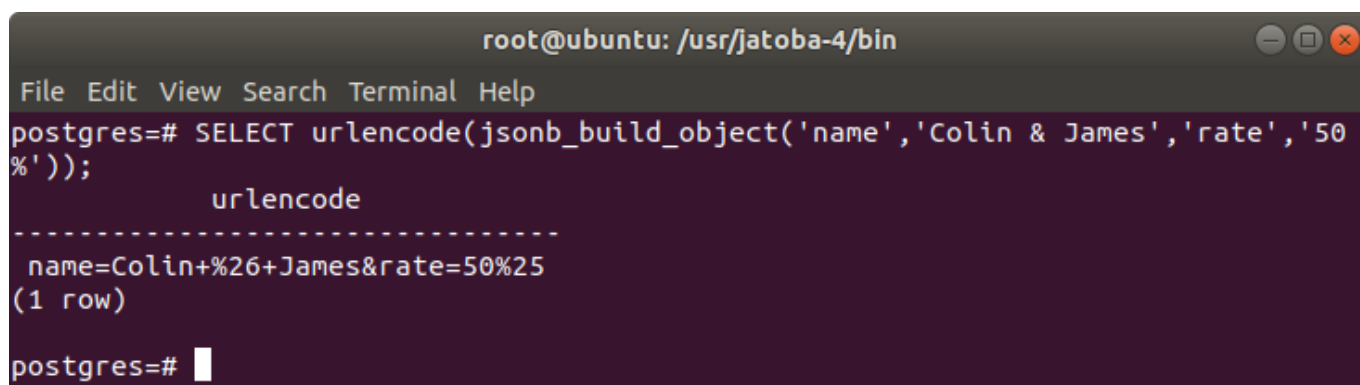
При использовании функции «urlencode» с использованием аргумента JSON

```
urlencode(data JSONB)
```

URL кодирует ассоциативный массив JSON.

Пример

```
SELECT urlencode(jsonb_build_object('name','Colin & James','rate','50%'));
```



The screenshot shows a terminal window titled 'root@ubuntu: /usr/jatoba-4/bin'. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The user has entered the PostgreSQL command: `postgres=# SELECT urlencode(jsonb_build_object('name','Colin & James','rate','50%'));`. The output shows the function name 'urlencode' followed by a dashed line separator and the result 'name=Colin+%26+James&rate=50%25'. Below the result, it says '(1 row)'. The prompt 'postgres=#' is visible at the bottom.

Рисунок 3.18 – SQL-запрос с кодированием массива JSON

4. УДАЛЕНИЕ КОМПОНЕНТА

4.1. Удаление компонента при отсутствии зависимых от него объектов

Для удаления компонента потребуется авторизоваться в СУБД и выполнить команду:

```
DROP extension http;
```

4.2. Удаление компонента при наличии зависимых от него объектов

Для удаления компонента вместе со всеми зависимыми от него объектами потребуется:

- авторизоваться в СУБД;
- выполнить команду:

```
DROP extension http cascade;
```

4.3. Удаление пакета

В ОС GNU/Linux выйти из psql и удалить пакет расширения, выполнив команду:

```
apt-get remove jatoba4-pgsql-http;
```

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

SQL	–	Structured Query Language – язык структурированных запросов
CLI	–	Command-line interface – интерфейс командной строки
БД	–	База данных
ОС	–	Операционная система
СУБД	–	Система управления базами данных
ФСТЭК России	–	Федеральная служба по техническому и экспортному контролю России

Лист регистрации изменений

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------