

ЗАЩИЩЕННАЯ СИСТЕМА УПРАВЛЕНИЯ
БАЗАМИ ДАННЫХ «ЈАТОВА»

Руководство по настройке. Часть 22.
Полнотекстовый поиск и определение похожих текстов.
Компонент «ja_Similar»

643.72410666.00067-07 98 01-22

Листов 16

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

АННОТАЦИЯ

В документе приведены сведения, необходимые для установки и эксплуатации компонента «ja_Similar» (далее по тексту – «компонент» или ja_Similar), предназначенного для выполнения полнотекстового поиска и определения похожих текстов.

Настоящее руководство предназначено для администраторов СУБД.



Все примеры в данном документе приведены для СУБД «Jatoba» версии ядра 6.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 Windows – «C:\Program Files\GIS\Jatoba\6\bin»;
- ОС Linux – «/usr/jatoba-6/bin».

Версия компонента — 1.0

Степени важности примечаний, применяемые в документе:



Важная информация – указания, требующие особого внимания



Дополнительная информация – указания, позволяющие упростить работу с изделием



Важная информация

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

СОДЕРЖАНИЕ

1. Назначение компонента.....	4
1.1. Условия применения.....	4
2. Установка и настройка.....	5
2.1. Установка пакета «ja_Similar».....	5
2.2. Установка расширения «ja_Similar».....	6
3. Функциональные возможности компонента.....	7
3.1. Функции и операторы	7
3.1.1. Функции расширения.....	7
3.1.2. Операторы расширения.....	7
3.2. Параметры GUC	7
3.3. Класс операторов для индексов.....	8
3.4. Пример использования.....	9
4. Удаление компонента	14
Перечень сокращений.....	15

1. НАЗНАЧЕНИЕ КОМПОНЕНТА

Компонент «ja_Similar» предназначен для предоставления функции и операторы для определения схожести текстов на основе техники хеширования, чувствительного к близким значениям (Locally-Sensitive Hashing, или LSH).

Решение такого типа задачи ранее не было доступно в PostgreSQL. С помощью этого расширения решаются типы задач:

- поиск плагиата;
- сравнение содержания статей;
- дедупликация документов;
- поиск аномалий в типовых документах;
- сравнение с эталоном на схожесть.

1.1. Условия применения

Компонент «ja_Similar» может использоваться с СУБД «Jatoba» версий 5.x и выше, под управлением операционных систем Windows и GNU/Linux.



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

Ограничений по совместимости с другими компонентами нет.

2. УСТАНОВКА И НАСТРОЙКА

Установка компонента должна производиться от имени пользователя, обладающего административными привилегиями в системе.

2.1. Установка пакета «ja_Similar»

Пакет компонента устанавливается после установки базовых пакетов СУБД «Jatoba»:

- jatoba<ver>-client – клиентская часть СУБД;
- jatoba<ver>-contrib – вспомогательный набор модулей (расширений) СУБД;
- jatoba<ver>-libs – основные библиотеки для клиентской и серверной части СУБД;
- jatoba<ver>-server – серверная часть СУБД.

В зависимости от типа ОС GNU/Linux пакет компонента устанавливается командой в терминале:

- ОС GNU/Linux Debian и производные от нее:

```
apt-get install jatoba<ver>-ja-similar
```

- ОС GNU/Linux Red Hat и производные от нее:

```
yum install jatoba<ver>-ja-similar
```

- ОС ALTLinux:

```
apt-get install jatoba<ver>-ja-similar
```

- ОС openSUSE:

```
zypper install jatoba<ver>-ja-similar
```

В ОС семейства Windows пакет компонента устанавливается при первичной установке в режиме «Выборочная установка».

2.2. Установка расширения «ja_Similar»

В СУБД «Jatoba» расширение устанавливается от имени и с правами привилегированного пользователя SQL-командой:

```
CREATE EXTENSION ja_similar;
```

3. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ КОМПОНЕНТА

Расширение «ja_similar» предоставляет функции и операторы для определения схожести текстов на основе техники хеширования, чувствительного к близким значениям (Locally-Sensitive Hashing, или LSH).

3.1. Функции и операторы

3.1.1. Функции расширения

`minhash_similarity` - это функция расширения, определяющая **расчетную схожесть текстов**. Выводимый расчет определяет схожесть текстов в диапазоне результатов:

- от нуля - значение указывает, что два текста полностью различны;
- до одного - значение указывает, что два текста идентичны.

Описание функции приведено в таблице 3.1.

Таблица 3.1 – Описание функции «`minhash_similarity`»

Оператор	Возвращает	Описание
<code>minhash_similarity(text, text)</code>	float8	Возвращает расчетную схожесть двух текстов.

3.1.2. Операторы расширения

Расширение имеет оператор приведенный в таблице 3.2.

Таблица 3.2 – Операторы расширения «ja_similar»

Оператор	Возвращает	Описание
<code>text <~> text</code>	boolean	Возвращает true, если схожесть аргументов выше текущего порога, заданного параметром <code>ja_lsh_fts.similarity_threshold</code> .

3.2. Параметры GUC

Расширение «ja_similar» использует параметры:

- `similarity_threshold` - **порог схожести**;

```
ja_similar.similarity_threshold (float8)
```

Параметр задаёт текущий порог схожести, который используют оператор `<~>` и функция «`minhash_similarity`». Это значение должно быть в диапазоне от 0 до 1.

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

Значение по умолчанию – 0.1.

– consistent_threshold - **отсекающий порог** (уровень);

```
ja_similar.consistent_threshold (float8)
```

Параметр задает отсекающий порог (уровень), который используется внутри индекса GIN при проверке значений на соответствие запросу, т.е. это предвыборка, которая отсекает неподходящий текст до поиска.

Это значение должно быть в диапазоне от 0 до 1.

Значение по умолчанию – 0.2.

Если указать высокое значение отсекающего порога (consistent_threshold), то при поиске будут отбираться тексты, совпадающие по наполнению примерно на 100%. То есть для расчета схожести будут отобраны тексты, совпадающие с искомым почти полностью.

При низком значении текущего порога схожести (consistent_threshold) для расчета схожести отбирается больше текстов, в том числе не обязательно схожих.

Если указать высокое значение отсекающего порога (consistent_threshold), то при расчете схожести (minhash_similarity) в результат попадут только те из подтекстов, которые совпадают в соотношении similarity_threshold * 100%.

Тогда расчетная схожесть текстов (minhash_similarity) будет стремиться к «1», если искомый текст содержится в найденном примерно полностью.

При низком пороге схожести (similarity_threshold) значение расчетной схожести тестов (minhash_similarity) отражает более общую схожесть текстов, т.е. им не обязательно совпадать.

Эти параметры были вынесены в GUC, для оптимизации конкретного поискового запроса, если при стандартных параметрах он долго выполняется или плохо считает схожесть.

3.3. Класс операторов для индексов

Компонент «ja_Similar» предоставляет класс операторов индекса GIN, позволяющий создавать индекс по текстовым столбцам для очень быстрого поиска по критерию схожести

подтекста с исходным текстом большой длины. Этот тип индекса поддерживает вышеописанный оператор схожести.

Пример:

```
# CREATE TABLE test_sim (id bigserial, content text);  
# CREATE INDEX sim_idx ON test_sim USING GIN (content  
gin_minhash_ops);
```

На данном этапе создан индекс по столбцу «content», используя который можно осуществлять поиск по схожести.

Пример типичного запроса:

```
SELECT t.id, minhash_similarity(t.content, :'text') AS  
similarity  
FROM test_sim AS t  
WHERE t.content <~> :'text'  
ORDER BY similarity DESC;
```

Расчет схожести текстов основан на подходе к созданию сигнатур, который по своей сути вероятностный, из-за чего могут возникать ситуации, когда:

- Подтекст полностью содержится в тексте, но схожесть меньше 1;
- Подтекст не полностью содержится в тексте, но итоговая схожесть равна 1;

Данные ситуации нормальны и не являются поводом для беспокойства. Также стоит выбирать для запросов тексты длиной не менее порядка 200-300 символов, так как на коротких текстах точность поиска довольно низкая.

3.4. Пример использования

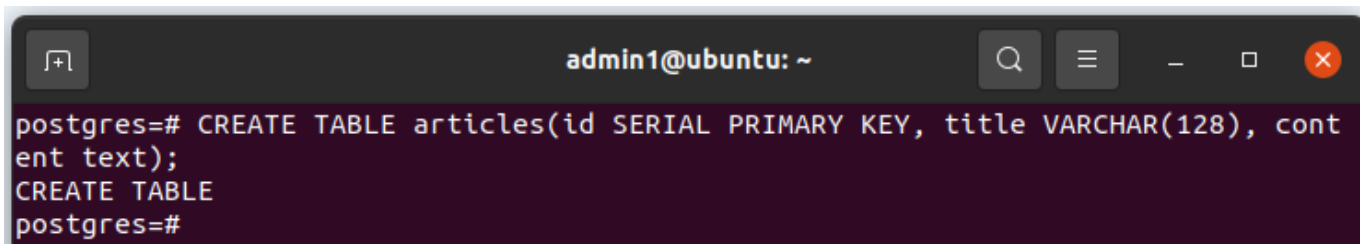
На примере, показанном ниже, демонстрируется, как выглядит работа с расширением на реальных данных. Для демонстрации взят дамп из 10000 статей Википедии сохраненный в каталог /wiki.

Требуется выполнить следующие шаги:

- Создать таблицу:

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

```
CREATE TABLE articles(id SERIAL PRIMARY KEY, title  
VARCHAR(128), content text);
```

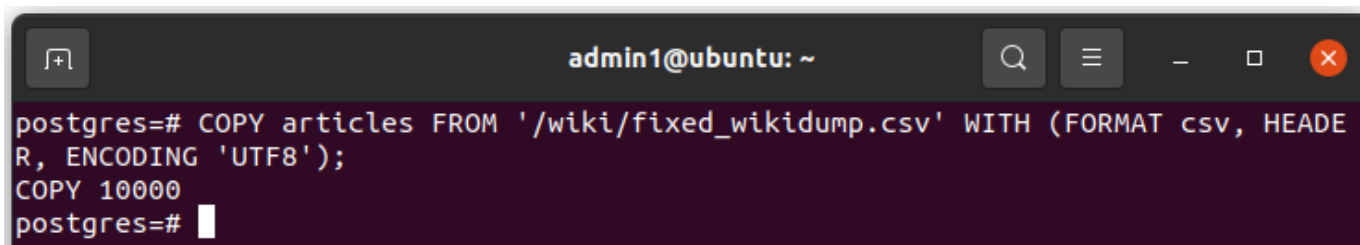


The screenshot shows a terminal window with the title 'admin1@ubuntu: ~'. The command 'CREATE TABLE articles(id SERIAL PRIMARY KEY, title VARCHAR(128), content text);' has been entered and executed successfully, as indicated by the prompt 'postgres=#' appearing twice.

Рисунок 3.1 – Создание таблицы

В таблицу загружаются данные из дампа Википедии fixed_wikidump.csv:

```
FROM '/wiki/fixed_wikidump.csv' WITH (FORMAT csv, HEADER,  
ENCODING 'UTF8');
```



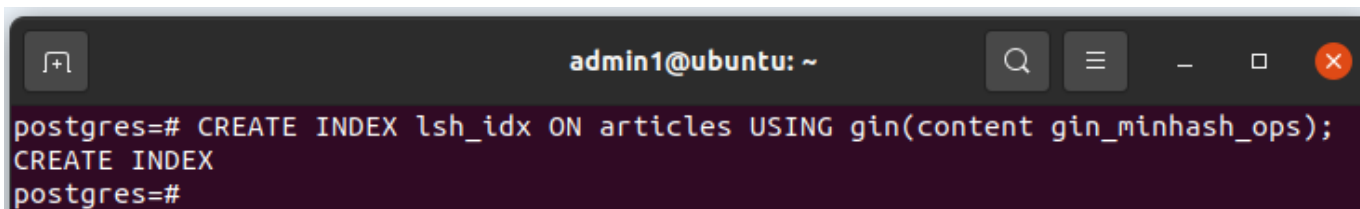
The screenshot shows a terminal window with the title 'admin1@ubuntu: ~'. The command 'COPY articles FROM '/wiki/fixed_wikidump.csv' WITH (FORMAT csv, HEADER, ENCODING 'UTF8');' has been entered and executed successfully, as indicated by the prompt 'postgres=#' appearing twice.

Рисунок 3.2 – Загрузка дампа в таблицу

- Создать индекс LSH;

Создается индекс для поиска похожих текстов над колонкой «content» с содержимым статей SQL-командой:

```
CREATE INDEX lsh_idx ON articles USING gin(content  
gin_minhash_ops);
```



The screenshot shows a terminal window with the title 'admin1@ubuntu: ~'. The command 'CREATE INDEX lsh_idx ON articles USING gin(content gin_minhash_ops);' has been entered and executed successfully, as indicated by the prompt 'postgres=#' appearing twice.

Рисунок 3.3 – Создание индекса LSH

- Отключить последовательного сканирования SeqScan;

Для нормальной работы индекса GIN на больших данных выключить последовательное сканирование SeqScan, которое подразумевает последовательный перебор всех строк БД в поисках требуемого значения SQL-командой:

```
SET enable_seqscan = OFF;
```

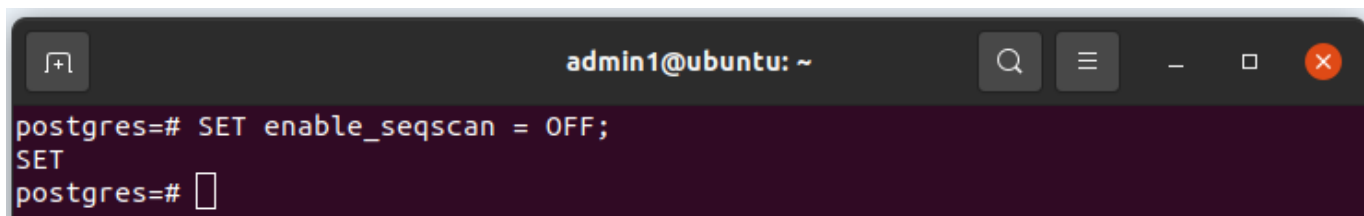


Рисунок 3.4 – Отключение сканирования SeqScan

- Задать произвольный текст;

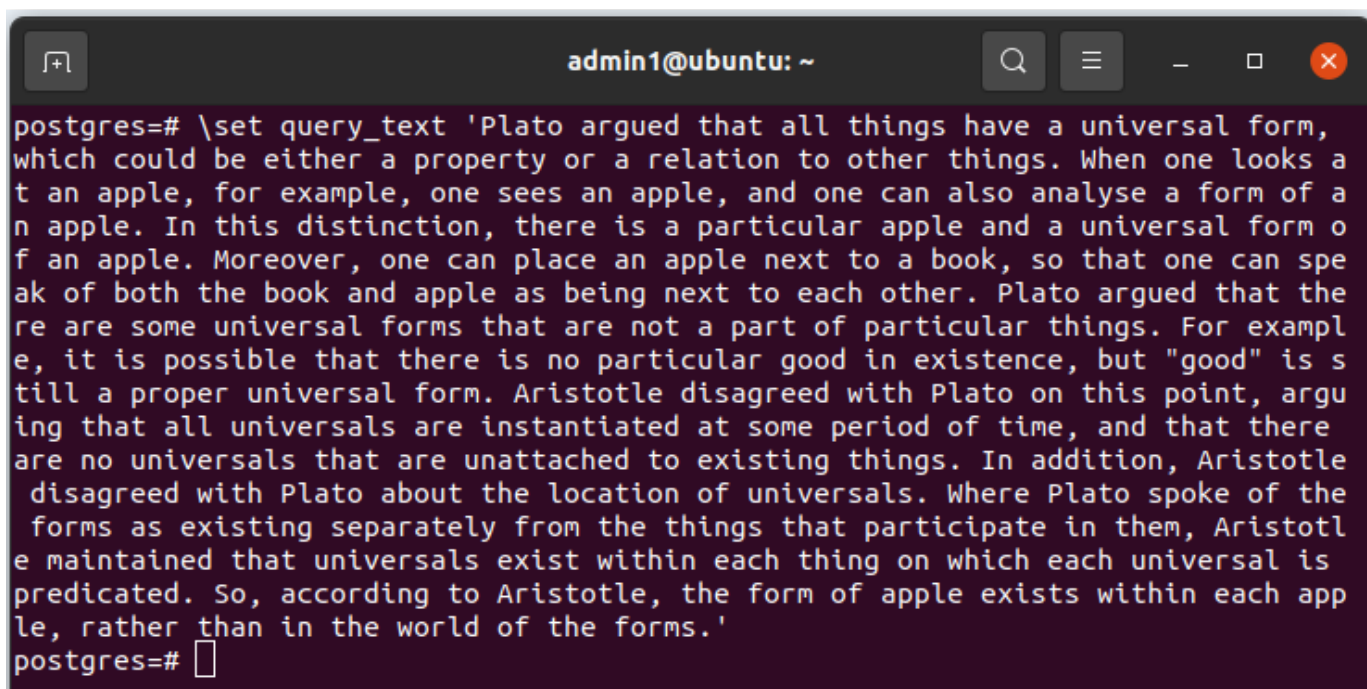
Задать произвольный текст для создания читаемого запроса, который будет искаться.

Для примера взят фрагмент из статьи про Аристотеля.



В СУБД «Jatoba» под управлением ОС Windows в SQL-команде используются двойные кавычки

```
\set query_text 'Plato argued that all things have a universal form, which could be either a property or a relation to other things. When one looks at an apple, for example, one sees an apple, and one can also analyse a form of an apple. In this distinction, there is a particular apple and a universal form of an apple. Moreover, one can place an apple next to a book, so that one can speak of both the book and apple as being next to each other. Plato argued that there are some universal forms that are not a part of particular things. For example, it is possible that there is no particular good in existence, but "good" is still a proper universal form. Aristotle disagreed with Plato on this point, arguing that all universals are instantiated at some period of time, and that there are no universals that are unattached to existing things. In addition, Aristotle disagreed with Plato about the location of universals. Where Plato spoke of the forms as existing separately from the things that participate in them, Aristotle maintained that universals exist within each thing on which each universal is predicated. So, according to Aristotle, the form of apple exists within each apple, rather than in the world of the forms.'
```



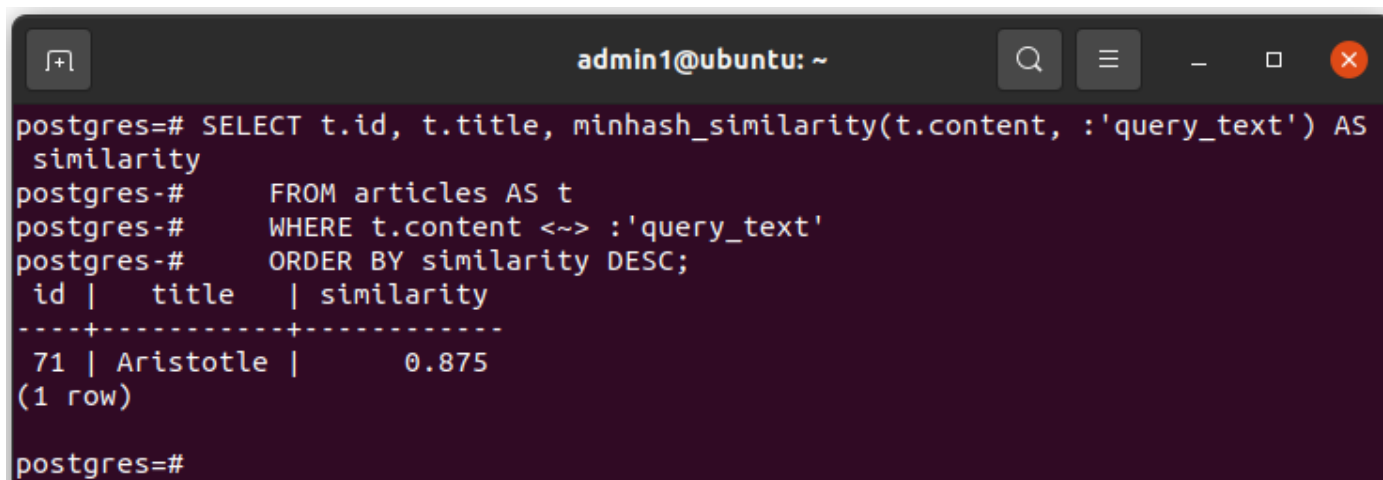
```
admin1@ubuntu: ~
postgres=# \set query_text 'Plato argued that all things have a universal form,
which could be either a property or a relation to other things. When one looks a
t an apple, for example, one sees an apple, and one can also analyse a form of a
n apple. In this distinction, there is a particular apple and a universal form o
f an apple. Moreover, one can place an apple next to a book, so that one can spe
ak of both the book and apple as being next to each other. Plato argued that the
re are some universal forms that are not a part of particular things. For exampl
e, it is possible that there is no particular good in existence, but "good" is s
till a proper universal form. Aristotle disagreed with Plato on this point, argu
ing that all universals are instantiated at some period of time, and that there
are no universals that are unattached to existing things. In addition, Aristotle
disagreed with Plato about the location of universals. Where Plato spoke of the
forms as existing separately from the things that participate in them, Aristotl
e maintained that universals exist within each thing on which each universal is
predicated. So, according to Aristotle, the form of apple exists within each app
le, rather than in the world of the forms.'
postgres=#
```

Рисунок 3.5 – Задание искомого текста

- Выполнить запрос на поиск похожих текстов:

В этом запросе мы получается список статей, содержимое которых имеет схожесть с нашим искомым куском текста, и эту самую расчетную схожесть.

```
SELECT t.id, t.title, minhash_similarity(t.content,
:'query_text') AS similarity
FROM articles AS t
WHERE t.content <~> :'query_text'
ORDER BY similarity DESC;
```



```
admin1@ubuntu: ~
postgres=# SELECT t.id, t.title, minhash_similarity(t.content, :'query_text') AS
similarity
postgres=# FROM articles AS t
postgres=# WHERE t.content <~> :'query_text'
postgres=# ORDER BY similarity DESC;
 id | title | similarity
----+-----+-----
 71 | Aristotle | 0.875
(1 row)

postgres=#
```

Рисунок 3.6 – Запрос и вывод схожести текстов

В выводе SQL-запроса возвращены:

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

- ID -71;
- название статьи – Aristotle;
- коэффициент схожести - 87.5%, близкий к 1 (87.5 процентов).

4. УДАЛЕНИЕ КОМПОНЕНТА

Удаление компонента проводится поэтапно.

Расширение удаляется SQL-командой:

```
DROP EXTENSION ja_similar CASCADE;
```

После чего проверяется отсутствие расширения в БД:

```
\dx
```

Проверяется отсутствие схемы данных в БД:

```
\dn
```

В зависимости от типа ОС GNU/Linux пакет компонента удаляется командой в терминале:

- ОС GNU/Linux Debian и производные от нее:

```
apt-get remove jatoba<ver>-ja-similar
```

- ОС GNU/Linux Red Hat и производные от нее:

```
yum remove jatoba<ver>-yum ja-similar
```

- ОС ALTLinux:

```
apt-get remove jatoba<ver>-ja-similar
```

- ОС openSUSE:

```
zypper remove jatoba<ver>-ja-similar
```

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

SQL	–	Structured Query Language
БД	–	База данных
ОС	–	Операционная система
СУБД	–	Система управления базами данных

[illegible]

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