

ЗАЩИЩЕННАЯ СИСТЕМА УПРАВЛЕНИЯ
БАЗАМИ ДАННЫХ «ЈАТОВА»

Руководство по настройке. Часть 5.
Планирование заданий СУБД
Компонент «pg_Task»

643.72410666.00067-07 98 01-05

Листов 17

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

АННОТАЦИЯ

В документе приведены сведения, описывающие работу планировщика заданий системы управления базами данных (СУБД), компонента pg_Task. Настоящее руководство предназначено для администратора СУБД «Jatoba».

Администратор СУБД «Jatoba» должен иметь навыки по работе с СУБД PostgreSQL или защищенной СУБД «Jatoba» (ООО «Газинформсервис»).



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 Windows – «C:\Program Files\GIS\Jatoba\5\bin»;
- ОС Linux – «/usr/jatoba-6/bin».

Для СУБД «Jatoba» версии ядра 4 используется версия компонента — 2.0.1

Для СУБД «Jatoba» версии ядра 5/6 используется версия компонента — 2.0.39



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. Функциональные возможности	4
1.2. Условия применения.....	4
2. Установка и настройка.....	5
2.1. Установка компонента на ОС Windows.....	5
2.2. Установка компонента ОС GNU/Linux	6
2.3. Установка компонента в СУБД	8
3. Создание задач для планировщика.....	11
4. Удаление задачи.....	14
5. Удаление компонента	15
Перечень сокращений.....	16

1. НАЗНАЧЕНИЕ КОМПОНЕНТА

Компонент pg_Task является планировщиком асинхронных задач.

1.1. Функциональные возможности

Компонент позволяет организовать выполнение фоновых заданий на языке SQL в заданное время в заданной базе данных от заданного пользователя.

Для своей работы в каждой базе данных, созданных в рамках сервера СУБД, компонент создает собственную таблицу для управления фоновыми заданиями.

Результаты выполнения заданий сохраняются в текстовом виде в специальном поле таблицы заданий.

1.2. Условия применения

Компонент pg_Task может использоваться совместно с СУБД «Jatoba» версий 1.x – 4.x и выше.

2. УСТАНОВКА И НАСТРОЙКА

2.1. Установка компонента на ОС Windows

Компонент устанавливается в составе СУБД «Jatoba» под управлением ОС Windows при первичной установке.

а) в окне «Выбор типа установки» следует выбрать тип установки «Выборочная» (см. рис. 2.1);

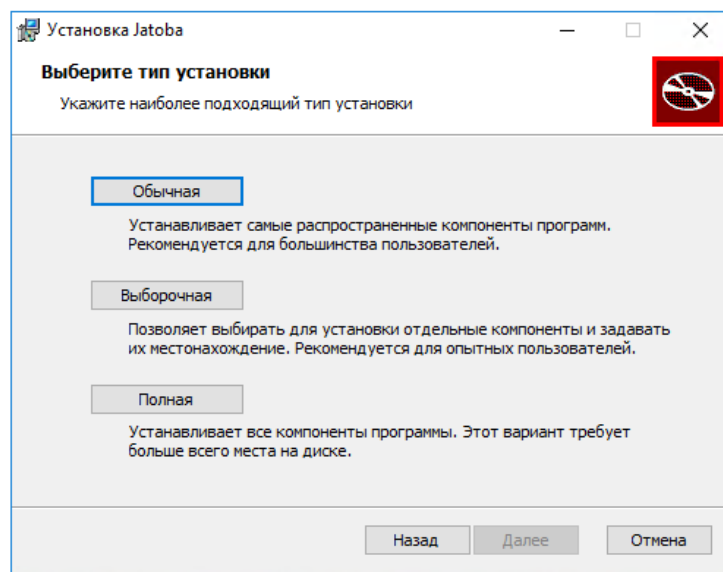


Рисунок 2.1 – Окно выбора типа установки

б) в диалоговом окне (см. рис. 2.2) выбрать «Планирование заданий СУБД»;

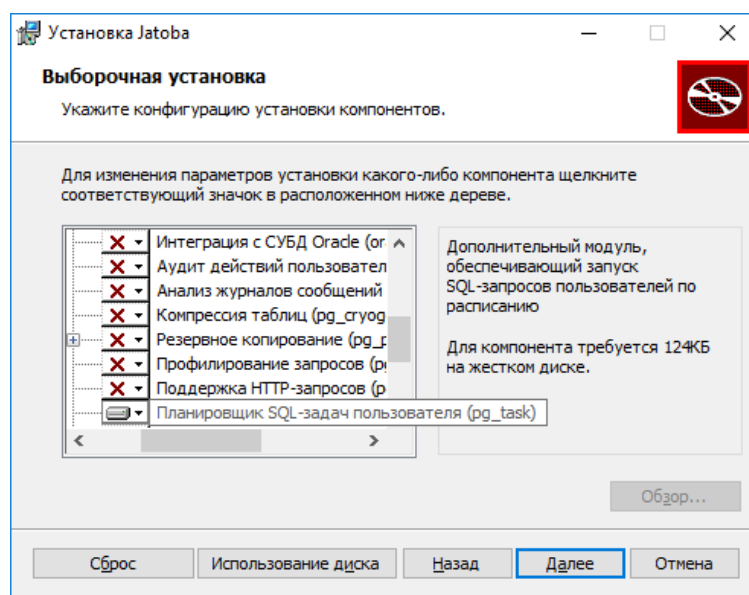


Рисунок 2.2 – Выбор устанавливаемых компонент

в) в открывшемся окне «Все готово к установке Jatoba» запустить процесс установки, нажав кнопку «Установить» (см. рис. 2.3);

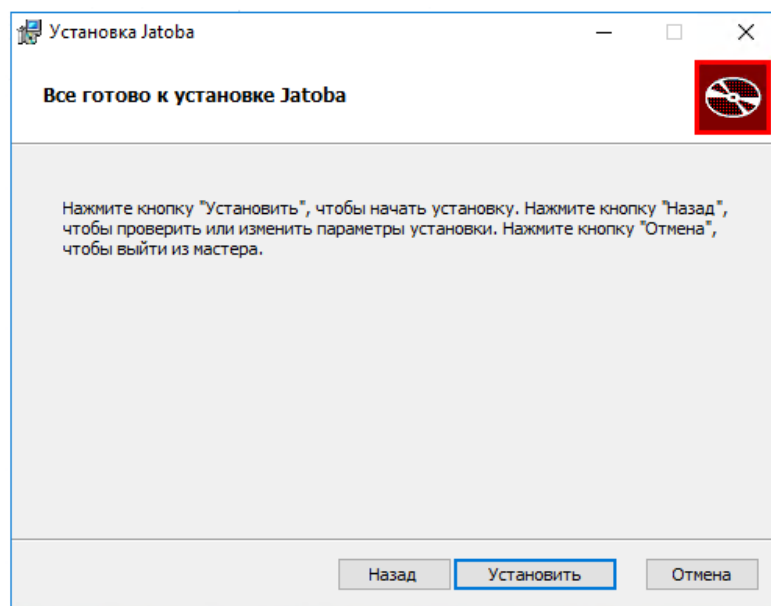


Рисунок 2.3 – Окно «Все готово к установке Jatoba»

2.2. 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 jatoba4-pg-task
```

– 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:

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

```
yum install jatoba4-pg-task
```

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 jatoba4-pg-task
```

– openSUSE также распространяется в виде rpm-пакетов, но использует собственный пакетный менеджер zypper, для нее команда установки выглядит следующим образом:

```
zypper install jatoba4-pg-task
```

Установка компонента в составе других версий СУБД «Jatoba» осуществляется аналогично. Отличие будет только в номере версии СУБД, в составе которой он распространяется. Например, jatoba1-pg-task 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.

2.3. Установка компонента в СУБД

Установка производится администратором базы данных. Для установки планировщика не требуется загрузка расширения, достаточно прописать в postgresql.conf:

```
shared_preload_libraries = 'pg_task'
```

В этом же файле можно по желанию добавить значение для пустой строки (параметр null в таблице 2.1):

```
pg_task.default_null = '\\N'
```

Можно также добавить список баз данных и пользователей, на которых должны выполняться задачи, в формате json-строки. Если БД и/или роли не существуют на текущий момент, то они будут созданы.

```
pg_task.json = ' [{"data":"postgres"}] '
```

Примеры:

```
pg_task.json= ' [{"data":"db1", "user":"user1"}, {"data":"db2",  
"user":"user2"}] '
```

```
pg_task.json =  
' [{"data":"database1"}, {"data":"database2", "user":"username2"},  
{"data":"database3", "schema":"schema3"}, {"data":"database4", "ta  
ble":"table4"}, {"data":"database5", "timeout":100}] '
```

Если эти данные не указать, по умолчанию планировщик создаст таблицы task во всех базах данных от имени пользователей баз данных и в схемах по умолчанию для этих пользователей.



Пользователь по умолчанию – postgres.

База данных по умолчанию – postgres.

Схема – public.

Использование компонента доступно для пользователей, имеющих доступ к таблице task.

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

После старта сервера планировщик создает, если еще не создана, таблицу задач со следующими столбцами (см. таблицу 2.1):

Таблица 2.1 – Таблица задач

Параметр	Тип	Обязательный (ненулевой)	Значение по умолчанию	Описание
id	BIGSERIAL	+	autoincrement	первичный ключ
parent	BIGINT	-		внешний ключ к родительской задаче (при необходимости)
plan	TIMESTAMP	+	CURRENT_TIME STAMP	запланированное время начала (по умолчанию – как можно быстрее)
start	TIMESTAMP	-		фактическое время начала
stop	TIMESTAMP	-		фактическое время остановки
active	INTERVAL	+	1 hour	период, в который задача исполняется
live	INTERVAL	+	0 sec	максимальное время жизни текущего рабочего процесса
repeat	INTERVAL	+	0 sec	интервал повторения задачи
timeout	INTERVAL	+	0 sec	позволяет ограничивать время выполнения задачи
count	INTEGER	+	0	количество одновременно выполняемых задач в группе
hash	INTEGER	+		хэш-сумма групповой задачи
max	INT	+	0	максимальное количество одновременно выполняемых задач в группе
pid	INT	-		идентификатор процесса, выполняющего задачу
state	STATE	+	PLAN	статус задачи: PLAN, TAKE, WORK, DONE, FAIL, STOP
delete	BOOLEAN	+	true	удалять задачу автоматически после выполнения, если нет результата (вывод равен нулю)
drift	BOOLEAN	+	false	предотвращать ли отсчет времени при повторе задачи
header	BOOLEAN	+	true	флаг присоединения заголовка с названием колонок, если задача возвращает таблицу
string	BOOLEAN	+	true	флаг, отображающий, является ли результат вывода строковым типом

Параметр	Тип	Обязательный (ненулевой)	Значение по умолчанию	Описание
delimiter	CHAR	+	\t	символ-разделитель, используемый для оформления строки результата (параметр output)
escape	CHAR	+		escape-символ, используемый для оформления строки результата
quote	CHAR	+		символ кавычек, используемый для оформления строки результата
data	TEXT	-		пользовательская информация
error	TEXT	-		текст ошибки
group	TEXT	+	group	групповая задача
input	TEXT	+		непосредственно сама задача для выполнения на языке SQL
null	TEXT	+	\N	текст «пустого» значения, используется для оформления строки результата
output	TEXT	-		полученный результат выполнения задачи
remote	TEXT	-		подключение к удаленной базе данных (при необходимости)

3. СОЗДАНИЕ ЗАДАЧ ДЛЯ ПЛАНИРОВЩИКА

Планировщик запускает несколько фоновых рабочих процессов:

- один для отслеживания изменений в конфигурационном файле и запуске/остановке при необходимости остальных фоновых процессов планировщика;
- по одному фоновому рабочему процессу для каждой базы для проверки запланированных задач в каждой базе и запуске при необходимости выполнения задач.

Для быстрого выполнения задачи нужно выполнить SQL-команду:

```
INSERT INTO task (input) VALUES ('SELECT now()');
```

- input – задача, которую необходимо выполнить.

Результат выполнения задачи планировщик записывает в столбец результата (output) в текстовом виде. Если в результате выполнения задачи будет несколько колонок, то pg_task создаст текстовую таблицу в колонке output.

В результате выполнения задачи может быть несколько строк, все они добавятся в колонку результата.

Для того, чтобы создать задачу и выполнить ее не сразу, а в определенное время, достаточно добавить в таблицу task следующую команду:

```
INSERT INTO task (plan, input) VALUES (now() + '5 min'::INTERVAL, 'SELECT now()');
```

- plan – планируемое время запуска задачи в стандартных форматах (например: "now() + '5 min'::INTERVAL" или, указав время, "2021-09-01 00:00:00");
- input – задача, которую необходимо выполнить.

Для того, чтобы поставить задачу на периодическое исполнение, нужно записать в таблицу task следующую команду:

```
INSERT INTO task (repeat, input) VALUES ('5 min', 'SELECT now()');
```

- repeat – периодичность запуска задачи;

- input – задача которую необходимо выполнить.



Если время исполнения задачи превышает период запуска задачи, то возможна ситуация, когда новая задача запустится до того, как закончится первая и задачи будут выполняться параллельно, что может вызвать некоторые проблемы.

Для недопущения подобной ситуации, нужно правильно оценить время выполнения задачи, либо использовать опцию drift, которая будет отсчитывать время запуска последующей задачи от момента завершения первой задачи:

```
INSERT INTO task (repeat, input, drift) VALUES ('5 min',  
'SELECT now()', false);
```

- repeat – периодичность запуска задачи;
- input – задача которую необходимо выполнить;
- drift – флаг, который указывает, отсчитывать ли время последующего запуска задачи от старта первой задачи (true) или от окончания первой задачи (false, по умолчанию).

При возникновении ошибки при выполнении задачи, она перехватывается и в текстовом виде записывается в колонку error, а задаче присваивается соответствующее состояние в колонке state.

Например, выполнение команды:

```
INSERT INTO task (input) VALUES ('SELECT 1/0');
```

создаст запись в колонке error:

```
....  
sqlerror\33816706  
message\division by zero  
....
```

При необходимости параллельного выполнения нескольких задач – можно создать группы:

```
INSERT INTO task ("group", max, input) VALUES ('group_1', 2,  
'SELECT 1');  
INSERT INTO task ("group", max, input) VALUES ('group_1', 2,  
'SELECT 2');
```

- group – уникальное имя группы;
- max – количество параллельно исполняемых задач в группе;
- input – задача, которую необходимо выполнить.

Также можно увеличить количество параллельных задач для определенной группы:

```
INSERT INTO task ("group", max, input) VALUES ('group_1', 3,  
'SELECT 3');
```

- группа group_1 имеет 3 параллельных задачи.

Запуск задачи на удаленной базе данных:

```
INSERT INTO task (input, remote) VALUES ('SELECT now()',  
'user=user host=host');
```

- input – задача, которую необходимо выполнить;
- remote – данные для подключения к удаленной базе данных, входные параметры – user, password, host, port, dbname.

Все вышеуказанные примеры использования компонента можно комбинировать под конкретные цели.

4. УДАЛЕНИЕ ЗАДАЧИ

Для удаления задачи необходимо удалить строку (строки) конкретной задачи из таблицы task.



Проще всего идентифицировать строки по полям input, id, или parent.

```
DELETE FROM task WHERE id = 77;  
DELETE FROM task WHERE parent = 77;  
DELETE FROM task WHERE input = 'SELECT now()';
```

5. УДАЛЕНИЕ КОМПОНЕНТА

Для удаления компонента достаточно закомментировать в конфигурационном файле «postgresql.conf» строку:

```
#shared_preload_libraries = 'pg_task'
```

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

SQL	–	Structured Query Language – язык структурированных запросов
БД	–	База данных
СУБД	–	Система управления базами данных

Лист регистрации изменений

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