

ЗАЩИЩЕННАЯ СИСТЕМА УПРАВЛЕНИЯ
БАЗАМИ ДАННЫХ «ЈАТОВА»

Руководство по настройке. Часть 25.
Сжатие данных на уровне страниц.
Компонент «ja_Compression»

643.72410666.00067-07 98 01-25

Листов 43

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

АННОТАЦИЯ

В документе приведены сведения, необходимые для установки и эксплуатации компонента «ja_Compression» (далее по тексту – «компонент» или ja_Compression), предназначенного для сжатия данных СУБД «Jatoba».

Настоящее руководство предназначено для администраторов СУБД.



Все примеры в данном документе приведены для СУБД «Jatoba» версии ядра 6.x.

Например, СУБД «Jatoba» версии 6.x по умолчанию устанавливается в директорию 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.....	5
1.1. Условия применения.....	5
1.2. Ограничения по эксплуатации.....	5
1.2.1. Двойная компрессия TOAST в табличном пространстве.....	5
1.2.2. Фрагментация.....	6
1.2.3. RUM индекс.....	6
1.2.4. Особого внимания требуют табличные пространства (ТП).....	7
1.2.5. Изменение алгоритма компрессии.....	7
1.2.6. Перемещение таблицы между ТП с разным алгоритмом компрессии.....	8
1.2.7. Инкрементальные копии компрессированных таблиц в СУБД «Jatoba» 18.....	9
2. Установка и настройка.....	11
3. Функциональные возможности компонента.....	12
3.1. Создание табличного пространства с сжатием данных.....	14
3.2. Перенос БД в новое табличное пространство с сжатием данных.....	15
4. SQL операции над компрессированными таблицами и индексами.....	17
4.1. Компрессия таблиц.....	17
4.2. Компрессия индексов таблиц.....	18
4.3. Компрессия других видов индексов.....	19
4.3.1. BRIN.....	19
4.3.2. GIN.....	20
4.3.3. HASH.....	20
4.3.4. GIST.....	21
4.3.5. SP GIST.....	21
4.3.6. Btree.....	22
4.3.7. RUM.....	22
4.4. Смена алгоритма компрессии (alter table).....	23
4.5. Смена алгоритма компрессии индекса (alter index).....	24
4.6. Компрессия журналируемых таблиц (logged tables) и нежурналируемых таблиц (unlogged tables), временных таблиц (temporary tables) и их индексов.....	24
4.7. Компрессия TOAST файлов.....	25
4.8. Компрессия партиций.....	25
4.9. Компрессия распределенных таблиц.....	26
5. Резервирование и восстановление.....	29
5.1. Утилита pg_dump.....	29
5.2. Утилита pg_dumpall.....	29
5.3. Утилита pg_basebackup.....	29
6. Репликация.....	30

7. Удаление компонента	31
8. Настройка 1С:КИП с БД с компрессией	32
8.1. Предварительные условия	32
8.2. Настройка БД с компрессией.....	32
8.3. Запуск тестов в 1С КИП (Тест-центр)	37
Термины и определения	40
Перечень сокращений.....	42

1. НАЗНАЧЕНИЕ КОМПОНЕНТА

Компонент «ja_Compression» предназначен для экономии имеющегося дискового пространства физического сервера СУБД.

При использовании компонента «ja_Compression» достигаемая экономия дискового пространства составляет от двух до семи раз в зависимости от сжимаемости данных. При этом степень компрессии зависит от сжимаемости данных.

Имеют малый коэффициент сжатия:

- GUID - статистически уникальный 128-битный идентификатор;
- бинарные данные и случайные данные, например, иллюстрации, сохраненные в базе данных;

Плохо сжимаются индексы, так как содержат бинарные данные. Совсем не сжимаются таблицы, созданные компонентом JCS, так как данные зашифрованы и хаотичны. Не сжимаются уже сжатые данные, см ниже про TOAST (тост-таблицах). Мало эффективно сжатие маленьких таблиц до 100 кб данных.

1.1. Условия применения

Компонент «ja_Compression» может использоваться с СУБД «Jatoba» версий 6.x и выше, под управлением GNU/Linux.

1.2. Ограничения по эксплуатации

Для поддержки компрессии в ОС должен быть установлен zstd версии 1.4 и выше.

Следует избегать применения функций, указанных в следующих подразделах.

1.2.1. Двойная компрессия TOAST в табличном пространстве

Если в таблице присутствует длинный тип данных TOAST (LOB) и таблица помещена в ТП, то возможно двойное сжатие. То есть сначала применяется компрессия TOAST, потом поверх неё применяется компрессия на уровне страниц. Двойная компрессия может дать всего порядка 10% выигрыша по сравнению с одиночной.

Чтобы избежать двойного сжатия следует:

- Вариант № 1

Таблицы с TOAST не помещать сжатое ТП, а использовать индивидуальный признак компрессии. В этом случае TOAST будет сжат своим алгоритмом (по умолчанию используется алгоритм pglz).

– Вариант № 2

Возможно снять компрессию TOAST путем изменения режима хранения колонок на EXTERNAL:

```
ALTER TABLE mytable alter column mycol SET storage external;
```

В этом случае будет использована только компрессия на уровне страниц.

Дополнительным вариантом является снятие компрессии TOAST путем изменения режима хранения колонок на PLAIN:

```
ALTER TABLE mytable alter column mycol SET storage PLAIN;
```

1.2.2. Фрагментация

После множества операций UPDATE/DELETE возникает фрагментация и хаотичное расположение блоков данных. Это приводит к тому, что головка магнитного диска будет хаотично перемещаться, что уменьшает скорость доступа к фрагментированным данным. Так же возрастает объем файлов и появляются неиспользованные промежутки между блоками данных.

Чтобы дефрагментировать таблицу надо выполнить операции:

```
VACUUM full mytable;  
REINDEX table mytable;
```

1.2.3. RUM индекс

Объекты такие как индекс RUM таблицы имеют названия алгоритма в пути до конкретного файла этого индекса, если оно находится в компрессированном табличном пространстве.

1.2.4. Особого внимания требуют табличные пространства (ТП)

После создания ТП, его компрессию можно изменять только пока оно не содержит данных. Как только появляется хоть один объект (таблица, индекс и так далее), то изменение компрессии уже невозможно.

```
CREATE TABLESPACE tablespace_c LOCATION '/dir/tablespace_c';  
ALTER CREATE TABLESPACE tablespace_c SET (compression=zstd);  
OK  
CREATE TABLESPACE TABLESPACE_c LOCATION '/dir/tablespace_c';  
CREATE TABLE a(a int) TABLESPACE tablespace_c;  
ALTER CREATE TABLESPACE tablespace_c SET (compression=zstd);  
ОШИБКА смена алгоритма компрессии запрещена
```

После переноса базы данных в компрессированное ТП, - компрессия сразу не применяется. Чтобы применилась компрессия к объектам БД, - надо выполнить запросы `vacuum full; reindex database;`

```
CREATE TABLESPACE tablespace_c LOCATION  
'/var/lib/jatoba/6/tablespace_c' with(compression=zstd);  
ALTER DATABASE mydb SET TABLESPACE tablespace_c;  
\c mydb  
vacuum full;  
reindex database;  
checkpoint;
```

После команд `vacuum full, reindex database`, компрессируются системные объекты типа `pg_class`. Так как эти объекты небольшие, то применение компрессии к ним будет малоэффективным. Но такого поведения можно избежать, если применять команду `vacuum full` только к отдельным таблицам.

1.2.5. Изменение алгоритма компрессии

Нельзя изменять компрессию у объекта внутри компрессированного ТП.

```
CREATE TABLESPACE tablespace_c LOCATION '/dir/tablespace_c'  
with(compression=zstd);
```

```
CREATE TABLE a(a int) TABLESPACE tablespace_c;  
ALTER TABLE a SET (compression=lz4);  
ОШИБКА смена алгоритма компрессии запрещена
```

1.2.6. Перемещение таблицы между ТП с разным алгоритмом компрессии

Попытка переместить таблицу из ТП одним алгоритмом сжатия в ТП с другим алгоритмом, при последующем выполнении vacuum full приведет к возникновению ошибки:

```
ERROR: table and tablespace option conflict
```

Для того, чтобы безопасно переместить таблицу между табличными пространствами и исключить конфликт между алгоритмами сжатия необходимо:

- 1) Переместить таблицу в табличное пространство по умолчанию pg_default:

```
alter table [table_name] set tablespace pg_default;
```

- 2) Отключить компрессию данных таблицы:

```
alter table [table_name] reset(compression);
```

- 3) Переместить таблицу в табличное пространство с компрессией:

```
alter table [table_name] set tablespace  
[tablespace_compression];
```

- 4) Выполнить сжатие данных табличного пространства:

```
vacuum full;  
reindex database;  
checkpoint;
```

- 5) После этого перемещенная таблица будет сжата по алгоритму, примененному к текущему табличному пространству.

После перемещения таблицы возможно переместить БД при помощи выполнения запроса:

```
alter database [db_name] set tablespace  
[tablespace_compression];
```


1.2.7. Инкрементальные копии компрессированных таблиц в СУБД «Jatoba» 18

Для корректного создания при помощи pg_basebackup резервных копий компрессированных таблиц, сжатых при помощи компонента ja_compress, в СУБД «Jatoba» 18 необходимо выполнить следующие процедуры:

1) Открыть конфигурационный файл /var/lib/jatoba/18/data/postgresql.conf для активации параметров:

```
summarize_wal = on  
wal_summary_keep_time = '10d'
```

Активировать параметр summarize_wal возможно также через утилиту psql:

```
postgres=# show summarize_wal;  
  
summarize_wal  
-----  
off  
  
postgres=# alter system set summarize_wal to on;  
postgres=# select pg_reload_conf();  
postgres=# show summarize_wal;  
  
summarize_wal  
-----  
on
```

После активации указанных параметров будут созданы специальные файлы, в которых сохраняются номера с измененными блоками данных. После этого утилита pg_basebackup использует эти файлы для создания инкрементальных резервных копий.

2) Создать полную резервную копию компрессированных таблиц (здесь и далее в качестве директории используется /var/lib/jatoba/backup/full).

```
./pg_basebackup -D /var/lib/jatoba/backup/full -Ft -P
```

3) Создать новую инкрементальную копию с компрессированными таблицами. Для этого необходимо использовать манифест после создания предыдущих резервных копий, например:

```
pg_basebackup --  
incremental=/var/lib/jatoba/backup/full/backup_manifest -D  
/var/lib/jatoba/backup/incr_backup1/
```



Сжатые компоненты ja_compress таблицы записываются в каждую инкрементальную резервную копию полностью.

2. УСТАНОВКА И НАСТРОЙКА

Компонент «ja_Compression» выполнен в виде патча ядра СУБД «Jatoba» и не имеет отдельного пакета.

Установка компонента «ja_Compression» производится от имени пользователя, обладающего административными привилегиями в системе при базовой установке СУБД «Jatoba» в ОС GNU/Linux (см. документ «Защищенная система управления базами данных «Jatoba». Руководство по установке 643.72410666.00067-07 97 01).

3. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ КОМПОНЕНТА

Для каждой таблицы СУБД «Jatoba», соответствуют определённые файлы на диске. В файлы данные поступают в следующих обстоятельствах: после операций commit, checkpoint и вытеснения буфера из оперативной памяти на диск.

Компрессия данных на уровне страниц происходит в момент поступления данных в компрессированные файлы. При этом ни буферы в памяти, ни сетевой трафик не сжимаются

Возможны следующие варианты использования компрессии:

- табличные пространства с компрессией;
- таблицы с компрессией;
- индексы с компрессией.

Все варианты использования компонента для компрессии данных приведены в таблице 3.1.

Таблица 3.1 – Варианты использования компрессии данных

Объект	Индивидуальный признак компрессии	Сжатие с использованием табличных пространств
Таблица	Да	Да
Встроенный индекс btree,hash, gin,gist,spgist,brin	Да	Да
Партиция	Да	Да
TOAST	Да	Да
RUM индекс	Нет	Да
База данных	Нет	Да
Объект системного каталога	Нет	Да

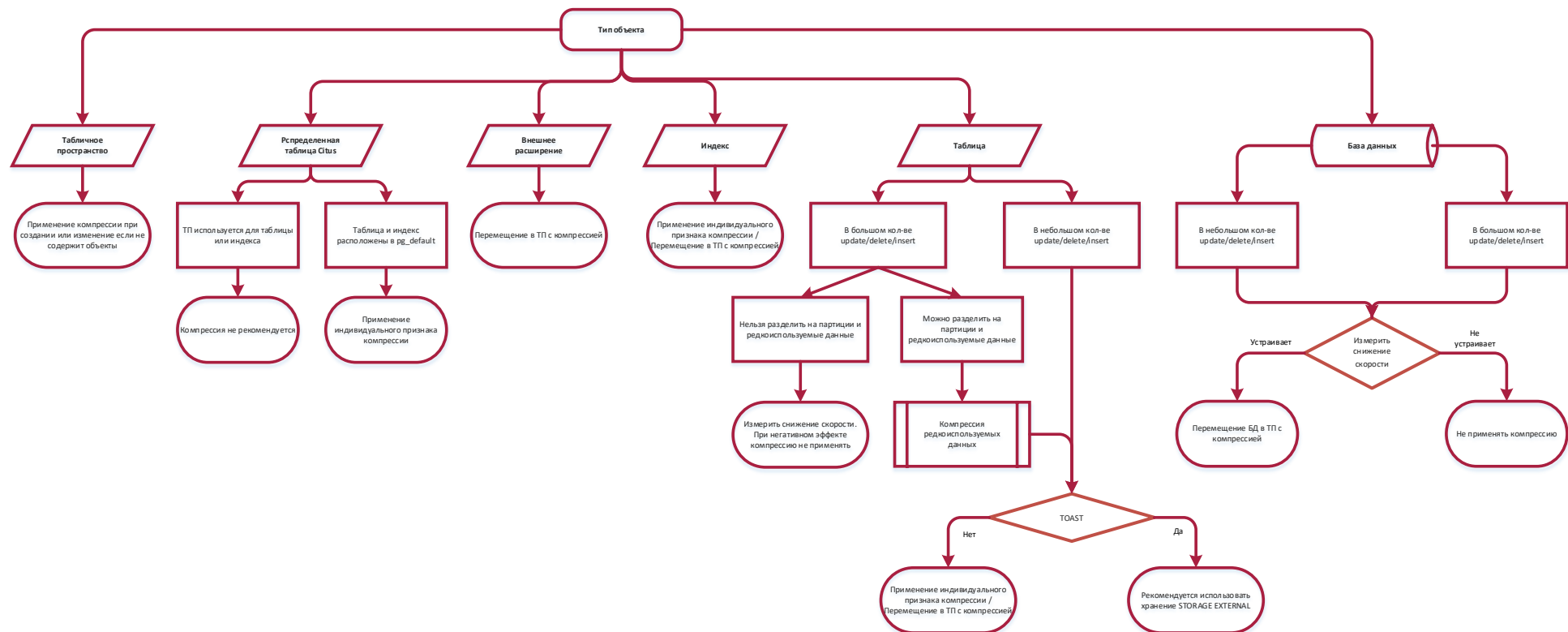


Рисунок 3.1 – Схема применения сжатия данных к различным объектам

Компонент «ja_Compression» совместим с БД 1С и может использоваться с любыми конфигурациями (см. раздел 8).

Компонент «ja_Compression» позволяет компрессировать данные с использованием следующих алгоритмов сжатия без потерь:

- zstd;
- lz4.

Далее по тексту в синтаксисе запросов `compres_type` может принимать значения `zstd` или `lz4` в зависимости от выбранного алгоритма сжатия данных

3.1. Создание табличного пространства с сжатием данных

При создании табличного пространства с сжатием данных применяется следующий синтаксис команды:

```
CREATE TABLESPACE [name_ts] location 'path to the tablespace'  
with (compression=[zstd|lz4]);
```

Реализовать это можно при помощи следующих шагов.

Создать каталог и назначить права в терминале ОС:

```
# mkdir -p /data/dbs  
# chown postgres:postgres /data/dbs
```

Создать табличное пространство в отдельном каталоге, например с использованием компрессии `zstd`:

```
CREATE TABLESPACE tablespace_compressed LOCATION '/data/dbs'  
with (compression=zstd);
```

Для того чтобы увидеть основные и дополнительные параметры созданного табличного пространства необходимо выполнить команду:

```
\db+
```

```
root@node1: /home/admin1
postgres=# CREATE TABLESPACE tablespace_compressed LOCATION '/data/dbs' with(compression=zstd);
CREATE TABLESPACE
postgres=# \db+

```

Name	Owner	Location	Access privileges	Options	Size
pg_default	postgres				29 MB
pg_global	postgres				589 kB
tablespace_compressed	postgres	/data/dbs		{compression=zstd}	4096 bytes

```
(3 rows)
postgres=#
```

Рисунок 3.2 – Создание табличного пространства с сжатием данных

3.2. Перенос БД в новое табличное пространство с сжатием данных

В существующее табличное пространство возможно переместить БД. Применяется следующий синтаксис команды:

```
ALTER DATABASE [name_db] SET TABLESPACE [name_ts];
```

Например:

```
ALTER DATABASE test_db1 SET TABLESPACE tablespace_compressed;
```

```
root@ubuntu: /home
postgres=# ALTER DATABASE test_db1 SET TABLESPACE tablespace_compressed;
ALTER DATABASE
postgres=#
```

Рисунок 3.3 – Перемещение БД в табличное пространство

После переноса переключиться на БД:

```
\c test_db1
```

После переноса БД в компрессированное ТП компрессия к данным сразу не применяется. Чтобы применилась компрессия к объектам БД следует выполнить полный вакуум (vacuum full), реиндексацию БД (reindex database) и создать контрольные точки (checkpoint).

В силу указанных причин следует выполнить SQL-команды:

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

```
vacuum full;  
reindex database;  
checkpoint;
```


4. SQL ОПЕРАЦИИ НАД КОМПРЕССИРОВАННЫМИ ТАБЛИЦАМИ И ИНДЕКСАМИ

Компрессия индексов и таблиц разнесена в отдельные операции.

Для задания компрессии на момент создания таблицы или индекса используется конструкция `with(compression=[compression _type])`, где `compression _type` может соответствовать одному из поддерживаемых компонентом алгоритмов сжатия данных без потерь:

- `zstd`;
- `lz4`.

4.1. Компрессия таблиц

Таблицы могут создаваться с признаком компрессии данных в БД или в табличном пространстве.

SQL-команда создания таблицы с компрессией имеет следующий синтаксис:

```
CREATE TABLE [name_table] WITH (compression=[zstd|lz4]);
```

Например:

```
CREATE TABLE table_example(a int) WITH (compression=zstd);
```

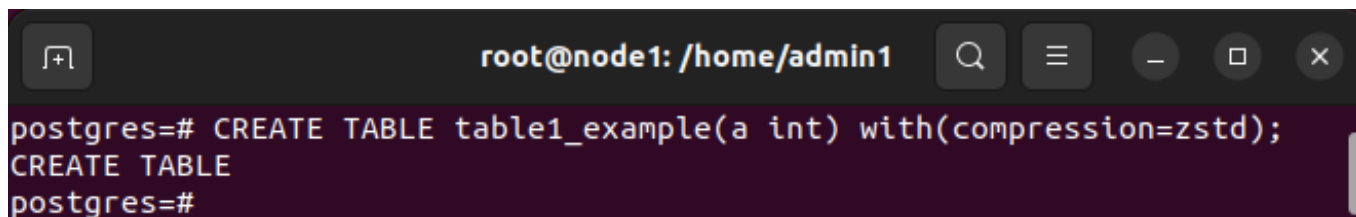


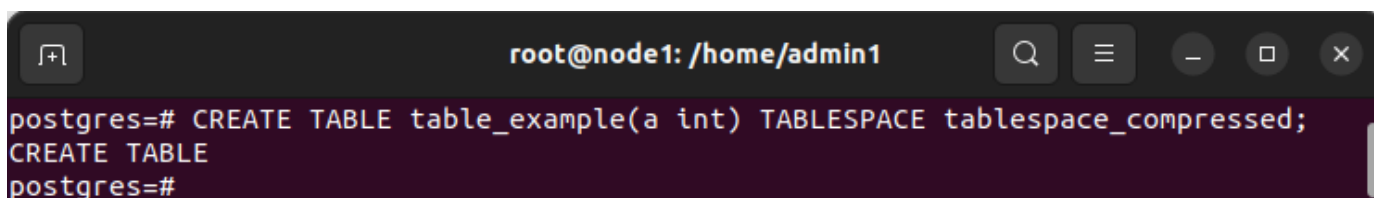
Рисунок 4.1 – Создание таблицы с компрессией zstd

SQL-команда для создания таблицы в табличном пространстве с компрессией имеет следующий синтаксис:

```
CREATE TABLE [name_table] TABLESPACE [name_ts];
```

Например:

```
CREATE TABLE table_example(a int) TABLESPACE  
tablespace_compressed;
```



```
root@node1: /home/admin1
postgres=# CREATE TABLE table_example(a int) TABLESPACE tablespace_compressed;
CREATE TABLE
postgres=#
```

Рисунок 4.2 - SQL-команда для создания таблицы в табличном пространстве с компрессией

4.2. Компрессия индексов таблиц

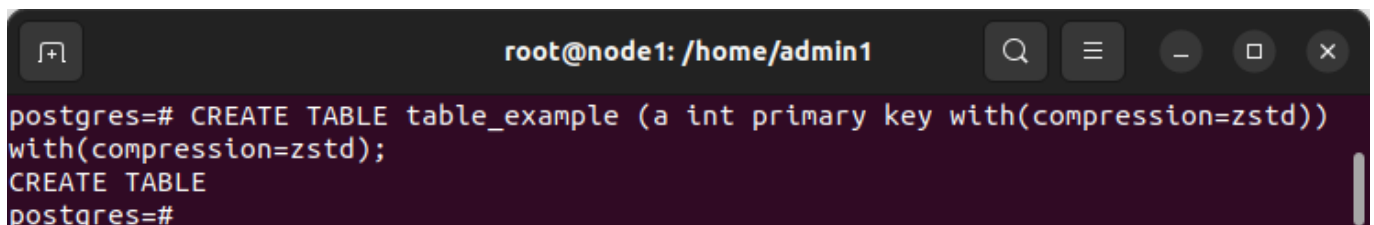
Компонент «ja_Compression» поддерживает компрессию индексов. Ниже приведены примеры сжатия индексов таблиц.

Создание таблицы и индекса таблицы с компрессией:

```
CREATE TABLE [name_table] ([key_name] primary key WITH
(compression=[zstd|lz4])) WITH (compression=[zstd|lz4]);
```

Например:

```
CREATE TABLE table_example (a int primary key
WITH(compression=zstd)) WITH (compression=zstd);
```



```
root@node1: /home/admin1
postgres=# CREATE TABLE table_example (a int primary key with(compression=zstd))
with(compression=zstd);
CREATE TABLE
postgres=#
```

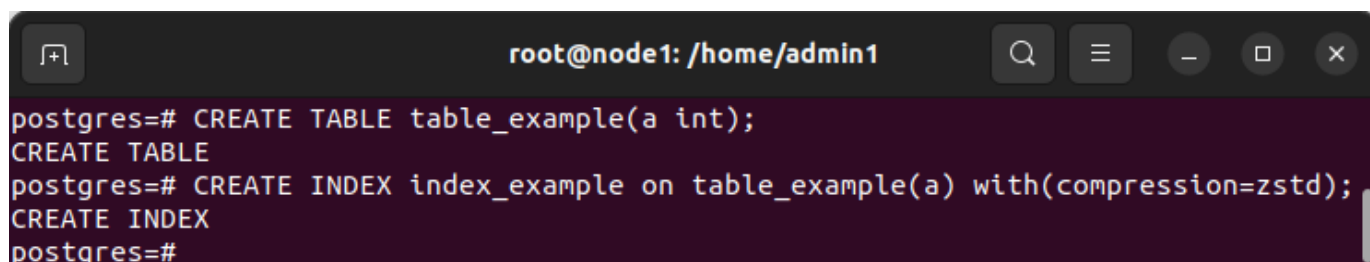
Рисунок 4.3 - Создание таблицы и индекса таблицы с компрессией zstd

Компрессия создаваемого индекса таблицы:

```
CREATE INDEX [index_name] ON [table_name]([column_name]) WITH
(compression=[zstd|lz4]);
```

Например:

```
CREATE INDEX index_example ON table_example(a)
with(compression=zstd);
```



```
root@node1: /home/admin1
postgres=# CREATE TABLE table_example(a int);
CREATE TABLE
postgres=# CREATE INDEX index_example on table_example(a) with(compression=zstd);
CREATE INDEX
postgres=#
```

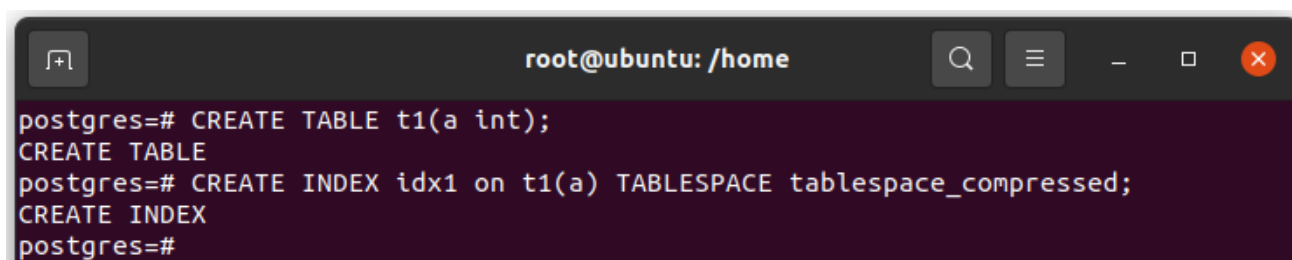
Рисунок 4.4 - Компрессия создаваемого индекса таблицы с использованием алгоритма zstd

Компрессия создаваемого индекса таблицы в табличном пространстве:

```
CREATE INDEX [index_name] ON [table_name] ([column_name])
TABLESPACE tablespace_compressed;
```

Например:

```
CREATE INDEX index_example ON table_example(a) TABLESPACE
tablespace_compressed;
```



```
root@ubuntu: /home
postgres=# CREATE TABLE t1(a int);
CREATE TABLE
postgres=# CREATE INDEX idx1 on t1(a) TABLESPACE tablespace_compressed;
CREATE INDEX
postgres=#
```

Рисунок 4.5 - Компрессия создаваемого индекса таблицы в табличном пространстве

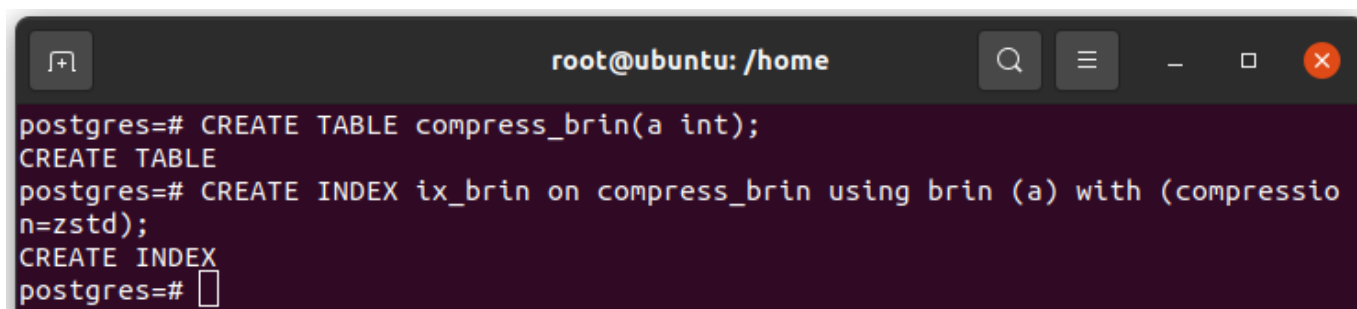
4.3. Компрессия других видов индексов

4.3.1. BRIN

Компонент «ja_Compression» поддерживает компрессию индексов типа BRIN.

Например:

```
CREATE TABLE compress_brin(a int);
CREATE INDEX ix_brin on compress_brin using brin (a) with
(compression=[zstd|lz4]);
```



```
root@ubuntu: /home
postgres=# CREATE TABLE compress_brin(a int);
CREATE TABLE
postgres=# CREATE INDEX ix_brin on compress_brin using brin (a) with (compression=zstd);
CREATE INDEX
postgres=#
```

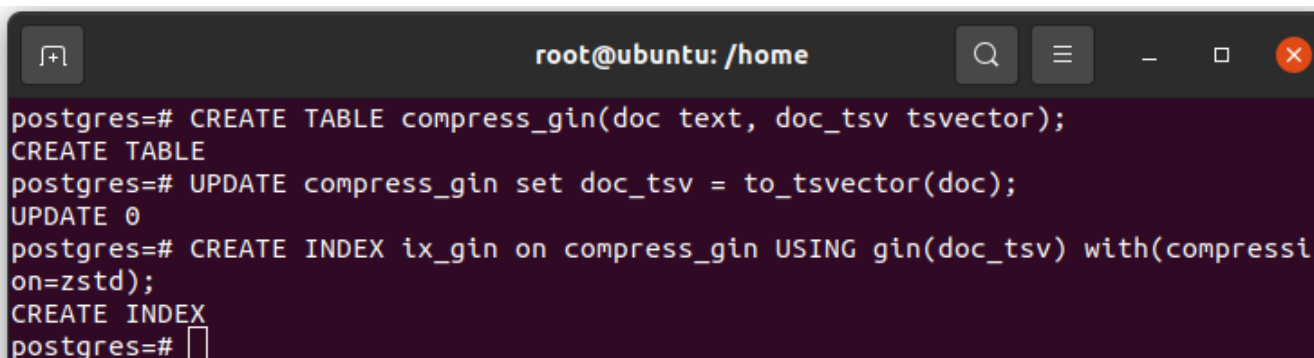
Рисунок 4.6 – Компрессия индекса типа BRIN с использованием алгоритма zstd

4.3.2. GIN

Компонент «ja_Compression» поддерживает компрессию индексов типа GIN.

Например:

```
CREATE TABLE compress_gin(doc text, doc_tsv tsvector);
UPDATE compress_gin set doc_tsv = to_tsvector(doc);
CREATE INDEX ix_gin on compress_gin USING gin(doc_tsv)
with(compression=[zstd|lz4]);
```



```
root@ubuntu: /home
postgres=# CREATE TABLE compress_gin(doc text, doc_tsv tsvector);
CREATE TABLE
postgres=# UPDATE compress_gin set doc_tsv = to_tsvector(doc);
UPDATE 0
postgres=# CREATE INDEX ix_gin on compress_gin USING gin(doc_tsv) with(compression=zstd);
CREATE INDEX
postgres=#
```

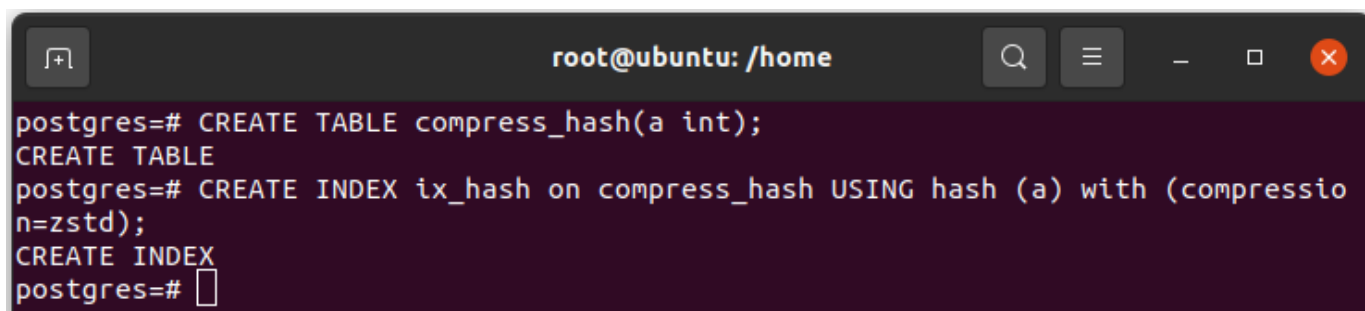
Рисунок 4.7 – Компрессия индекса типа GIN с использованием алгоритма zstd

4.3.3. HASH

Компонент «ja_Compression» поддерживает компрессию индексов типа HASH.

Например:

```
CREATE TABLE compress_hash(a int);
CREATE INDEX ix_hash on compress_hash USING hash (a) with
(compression=[zstd|lz4]);
```



```
root@ubuntu: /home
postgres=# CREATE TABLE compress_hash(a int);
CREATE TABLE
postgres=# CREATE INDEX ix_hash on compress_hash USING hash (a) with (compression=zstd);
CREATE INDEX
postgres=#
```

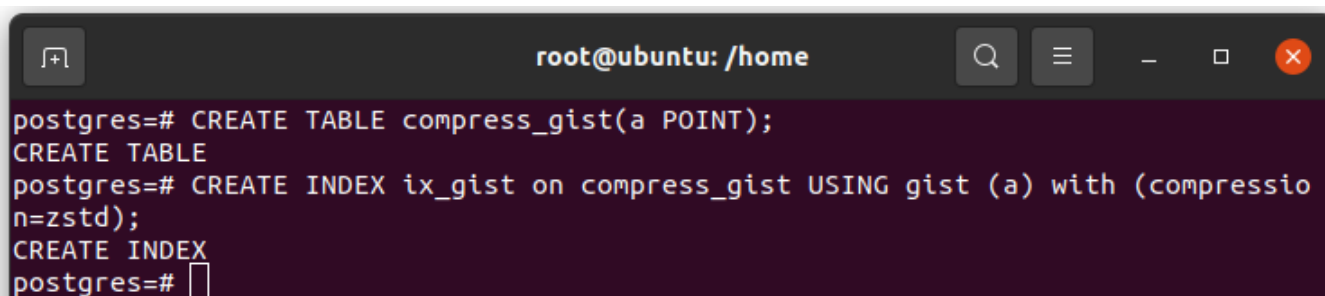
Рисунок 4.8 – Компрессия индекса типа HASH с использованием алгоритма zstd

4.3.4. GIST

Компонент «ja_Compression» поддерживает компрессию индексов типа GIST.

Например:

```
CREATE TABLE compress_gist(a POINT);
CREATE INDEX ix_gist on compress_gist USING gist (a) with
(compression=[zstd|lz4]);
```



```
root@ubuntu: /home
postgres=# CREATE TABLE compress_gist(a POINT);
CREATE TABLE
postgres=# CREATE INDEX ix_gist on compress_gist USING gist (a) with (compression=zstd);
CREATE INDEX
postgres=#
```

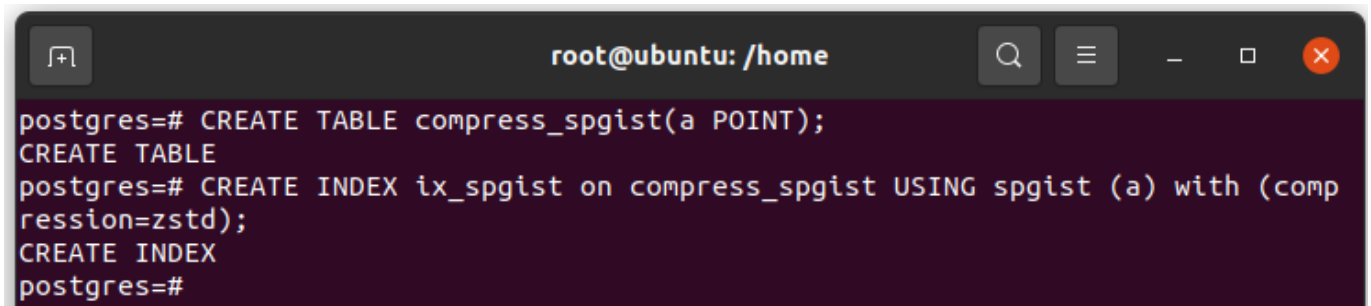
Рисунок 4.9 – Компрессия индекса типа GIST с использованием алгоритма zstd

4.3.5. SP GIST

Компонент «ja_Compression» поддерживает компрессию индексов типа SP GIST.

Например:

```
CREATE TABLE compress_spgist(a POINT);
CREATE INDEX ix_spgist on compress_spgist USING spgist (a) with
(compression=[zstd|lz4]);
```



```
root@ubuntu: /home
postgres=# CREATE TABLE compress_spgist(a POINT);
CREATE TABLE
postgres=# CREATE INDEX ix_spgist on compress_spgist USING spgist (a) with (compression=zstd);
CREATE INDEX
postgres=#
```

Рисунок 4.10 – Компрессия индекса типа SP GIST с использованием алгоритма zstd



Индекс SP GIST очень медленный индекс, лучше создавать его в конце после заполнения таблицы данными.

4.3.6. Btree

Компонент «ja_Compression» поддерживает компрессию индексов типа Btree.

Например:

```
CREATE TABLE compress_btree(a int);
CREATE INDEX ix_brin on compress_btree using Btree (a) with
(compression=[zstd|lz4]);
```

4.3.7. RUM

Компонент RUM является внешним расширением.

Установка компонента описана в документе «Руководство по установке» 643.72410666.00067-07 97 01.

Функциональные возможности компонента описаны в документе «Руководство администратора» 643.72410666.00067-07 95 01.

Компрессию RUM индекса можно включить, если использовать компрессированное табличное пространство.

Например:

```
CREATE EXTENSION rum;
CREATE TABLESPACE tablespace_compr 'path/to/file'
with(compression=[zstd|lz4]);
CREATE table test_rum (a tsvector);
CREATE INDEX rumidx ON test_rum USING rum (a rum_tsvector_ops)
tablespace tablespace_compr;
```

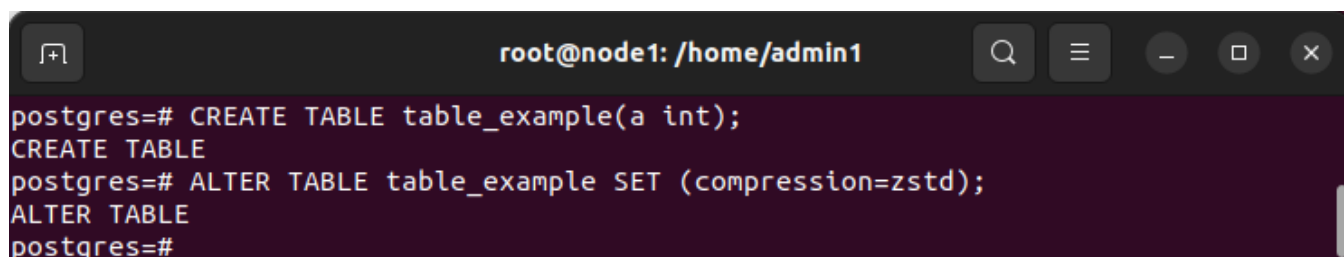
4.4. Смена алгоритма компрессии (alter table)

Для существующей таблицы возможно установить тип компрессии используя оператор SET.

```
ALTER TABLE [table_name] SET (compression=[zstd|lz4]);
```

Например:

```
ALTER TABLE table_example SET (compression=zstd);
```



```
root@node1: /home/admin1
postgres=# CREATE TABLE table_example(a int);
CREATE TABLE
postgres=# ALTER TABLE table_example SET (compression=zstd);
ALTER TABLE
postgres=#
```

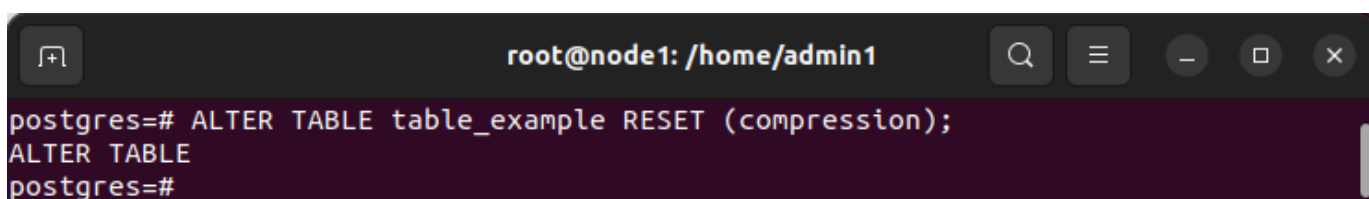
Рисунок 4.11 – Установка типа компрессии

Снять компрессию с таблицы возможно оператором RESET.

```
ALTER TABLE [table_name] RESET (compression);
```

Например:

```
ALTER TABLE table_example RESET (compression);
```



```
root@node1: /home/admin1
postgres=# ALTER TABLE table_example RESET (compression);
ALTER TABLE
postgres=#
```

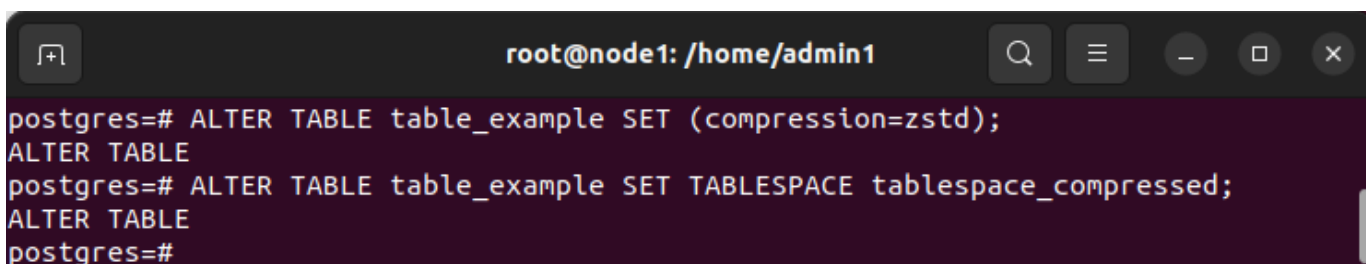
Рисунок 4.12 – Снятие компрессии с таблицы

Перенос таблицы в ТП с компрессией возможно с использованием оператора SET.

```
ALTER TABLE [table_name] SET TABLESPACE [name_ts];
```

Например:

```
ALTER TABLE table_example SET TABLESPACE tablespace_compressed;
```



```
root@node1: /home/admin1
postgres=# ALTER TABLE table_example SET (compression=zstd);
ALTER TABLE
postgres=# ALTER TABLE table_example SET TABLESPACE tablespace_compressed;
ALTER TABLE
postgres=#
```

Рисунок 4.13 - Перенос таблицы в табличное пространство

4.5. Смена алгоритма компрессии индекса (alter index)

Смена индивидуального алгоритма компрессии у индексов не работает. Вместо этого необходимо пересоздать индексы или переместить в табличное пространство.

Так же в целях уплотнения данных можно выполнить переиндексацию во время операций, подразумевающих переиндексацию, типа vacuum full, cluster, reindex.

```
ALTER INDEX [index_name] set tablespace [name_ts];
```

Например:

```
ALTER INDEX index_example set tablespace tablespace_compressed;
```

4.6. Компрессия журналируемых таблиц (logged tables) и нежурналируемых таблиц (unlogged tables), временных таблиц (temporary_tables) и их индексов

Компонент «ja_Compression» поддерживает компрессию журналируемых таблиц (logged tables), нежурналируемых таблиц (unlogged tables), временных таблиц (temporary_tables) и их индексов.

```
CREATE [unlogged|temporary|temp] table [table_name] ([key_name])
with (compression=[zstd|lz4]);
ALTER TABLE [table_name] SET unlogged;
```

Например:

```
CREATE unlogged TABLE table_example(a int)
with (compression=zstd);
ALTER TABLE table_example set unlogged;
```

- работает для постоянных (нежурналируемых) таблиц.

```
ALTER TABLE table_example SET logged;
```


- работает для постоянных (журналируемых) таблиц.

4.7. Компрессия TOAST файлов

При наличии столбцов с длинным типом (LOB) помимо основного файла - может создаваться TOAST файл, который привязан к таблице.

Концепция TOAST файлов предназначена для записи в таблицы длинных строк (так называемых LOB. TOAST файл создается, если в таблице объявлены колонки типа text или любого другого типа с большой длиной.

TOAST не имеет своих опций и наследует их от таблицы. При перемещении таблицы между ТП, файлы TOAST также автоматически перемещаются другое ТП. Применение сжатия к таблице так же автоматически распространяется на TOAST.

СУБД «Jatoba» предоставляет команды для управления TOAST:

```
CREATE TABLE [table_name] ([column_name]TEXT STORAGE  
[PLAIN|EXTERNAL|EXTENDED|MAIN] );
```

или

```
ALTER TABLE [table_name] ALTER COLUMN [column_name] SET STORAGE  
[PLAIN|EXTERNAL|EXTENDED|MAIN] ;
```

При этом некоторые режимы хранения подразумевают свой механизм сжатия данных. на уровне строк. Чтобы избежать двойного сжатия, рекомендуется использовать режим хранения EXTERNAL. В таком случае будет использоваться только сжатие на уровне страниц.

4.8. Компрессия партиций

В данном подразделе приведен сценарий, позволяющий сжимать партиции, содержащие данные, которые редко используются.

Создаем партицированную таблицу при помощи следующей команды:

```
CREATE TABLE compressed_partitioned_table(a int) PARTITION BY  
range(a) ;
```

Создаем компрессированную партицию при помощи команды:

```
CREATE TABLE compressed_partition_1 PARTITION OF  
compressed_partitioned_table FOR VALUES FROM (0) TO (500000)  
WITH (compression=[zstd|lz4]);
```

Создаем вторую некомпрессированную партицию при помощи команды:

```
CREATE TABLE compressed_partition_2 PARTITION OF  
compressed_partitioned_table FOR VALUES FROM (500000) TO  
(1000000);
```

Компенсируем вторую партицию с использованием одного из двух алгоритмов сжатия данных при помощи команды:

```
ALTER TABLE compressed_partition_2 SET (compression=[zstd|lz4]);
```

Либо выполняем перемещение второй компрессированной таблицы в другое ТП при помощи команды:

```
ALTER TABLE compressed_partition_2 SET TABLESPACE tablespace_  
compressed;
```

Табличные пространства могут располагаться на более дешевых и более медленных дисках, поэтому следует следить за частотой обращения к партициям и вовремя предпринимать меры по оптимизации скорости доступа и степени сжатия.



Алгоритм сжатия данных партиции и ТП при выполнении перемещения должны совпадать. Если для компрессированной партиции выбран алгоритм zstd, то перемещать такую партицию разрешено только в компрессированное ТП с алгоритмом сжатия данных zstd.

4.9. Компрессия распределенных таблиц

При использовании компонента ja_Hipe_Cluster для реализации горизонтального масштабирования может применяться компрессия распределенных таблиц (шардов).

В таком случае на координаторе кластера необходимо создать таблицу с использованием одного из алгоритмов сжатия:

```
CREATE TABLE [table_name] (f1 int) WITH (compression=[zstd|lz4]);
```

Или применить на узле координатор кластера алгоритм сжатия к существующей таблице при помощи запроса:

```
ALTER TABLE [table_name] SET (compression=[zstd|lz4]);
```

После этого необходимо создать распределенную таблицу на узлах кластера при помощи запроса:

```
SELECT create_distributed_table('[table_name]', 'f1',  
shard_count:=2);
```

Количество шардов в предыдущем запросе указывается в параметре `shard_count`.

После этого скомпрессированная таблица будет распределена на узлы кластера и к ней будет применяться выбранный ранее алгоритм сжатия.

В связи с особенностями шардирования размер сжатой распределенной таблицы может отличаться на разных узлах кластера.

Для того, чтобы отключить механизм сжатия данных для распределенных таблиц необходимо на узле координатор выбрать таблицу и выполнить следующий запрос

```
ALTER TABLE [table_name] reset (compression);
```

Ограничения по применению сжатия распределенных таблиц.

Можно применять компрессию, если таблица и её индексы принадлежат табличному пространству `pg_default`. Данный сценарий является предпочтительным.

Если же таблица или её индексы на координаторе принадлежит другому табличному пространству, и данное ТП не создано на узлах, то следующие запросы выполняются с ошибками:

```
CREATE TABLE a1(a text) tablespace tsp;  
SELECT create_distributed_table('a1', 'a');  
ERROR: tablespace "tsp" does not exist  
CONTEXT: while executing command on [IP-адрес]:5432
```

Для решения такой ситуации необходимо создать таблицу `a1` в ТП по умолчанию `pg_default`, или создать ТП `tsp` на распределенных узлах.

Создание индексов в ТП, отличном от `pg_default`, имеет определенные ограничения:

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

```
CREATE TABLE a2(a int);  
CREATE INDEX ix2 ON a2(a) tablespace tsp;  
SELECT create_distributed_table('a2', 'a');  
ERROR:  tablespace "tsp" does not exist  
CONTEXT:  while executing command on [IP-адрес]:5432
```

Для решения такой ситуации необходимо создать индекс ix2 в ТП по умолчанию pg_default, или создать ТП tsp на распределенных узлах.

Если распределяемая таблица или её индекс создается в ТП, для которого уже используется сжатие данных:

```
CREATE TABLE [table_name](a int) tablespace [tablespace_zstd];
```

То на распределенных узлах кластера эта же таблица будет располагаться в не сжатом ТП по умолчанию pg_default и не будет сжата.

5. РЕЗЕРВИРОВАНИЕ И ВОССТАНОВЛЕНИЕ

5.1. Утилита pg_dump

Утилита pg_dump сохраняет атрибут компрессии у отдельных таблиц. После завершения восстановления таблиц данных компрессия также восстанавливается.

Для ТП признак компрессии не сохраняется.

5.2. Утилита pg_dumpall

Утилита pg_dumpall сохраняет атрибут компрессии для табличных пространств и для индивидуальных объектов и табличных пространств.

При восстановлении требуется создать пустые директории, соответствующие ТП.

5.3. Утилита pg_basebackup

Утилита pg_basebackup сохраняет атрибут компрессии. После восстановления данных компрессия тоже восстанавливается при условии, что версия СУБД поддерживает компрессию.

6. РЕПЛИКАЦИЯ

Следует заметить, что сетевой трафик не сжимается, а сжимается только дисковые данные.

Различаются два вида репликации данных – физическая и логическая.

Для логической репликации данных можно настроить отдельно сжатие таблицы на мастере и отдельно на реплике(ах). Можно, например, указать, что таблица будет компрессирована только на реплике, а на мастере не будет.

Для физической репликации данных различается два случая:

- Если таблица создана до настройки репликации. При этом признак компрессии передается на реплику через утилиту `pg_basebackup`
- Если таблица создана после настройки репликации. При этом признак компрессии данных передается посредством WAL-журналов в реплику.

На данный момент реализована передача признака компрессии для обоих приведенных случаев.

Компрессия данных поддерживается компонентом «jaDog».

7. УДАЛЕНИЕ КОМПОНЕНТА

Выполнить удаление компонента «ja_Compression» стандартными средствами управления пакетами ОС невозможно, т.к. он выполнен в виде патча ядра СУБД «Jatoba».

8. НАСТРОЙКА 1С:КИП С БД С КОМПРЕССИЕЙ

В данном разделе рассматривается пример настройки 1С для работы с БД под управлением СУБД «Jatoba» подвергнутой компрессии компонентом «ja_Compression».

1С:КИП является инструментом для тестирования совместимости СУБД с 1С. Функциональные возможности компонента «ja_Compression» позволяют подвергать компрессии БД в любых конфигурациях 1С.

8.1. Предварительные условия

- настройка производится на 2 серверах:
 - Windows Server 2019 (сервер 1С);
 - Astra Linux 1.7.5 (сервер СУБД, версия Linux-сервера неважна);
- На Windows-сервере установлена платформа 1С актуальной версии, к примеру, 8.3.25.1445 для совместимости с актуальными версиями СУБД, настроен сервер администрирования 1С («кластер 1С»);
- На сервере СУБД установлена Jatoba 6.4.1 с поддержкой компрессии;
- На сервере СУБД для супер-пользователя задан пароль;

Тестирование заключается в вычислении условного относительного коэффициента APDEX (значение от 0.0 до 1.0)

8.2. Настройка БД с компрессией

- На сервере СУБД установить пакет компонента «1С_support»:

```
apt install jatoba6-1csupport
```

- В конфигурационный файл postgresql.conf внести рекомендуемые параметры для работы с 1С, как описано в п.п. 3.2 и 3.3 документа «Поддержка платформы 1С» 643.72410666.00067-07 98 01-13.

- В конфигурационный файл pg_hba.conf внести параметры, позволяющие серверу 1С подключаться к СУБД, к примеру

```
# IPv4 local connections:
```



```
host kip_compress postgres <IP-адрес сервера 1С> password
```

- Перезапустить СУБД в терминале ОС:

```
systemctl restart jatoba-6.service
```

- На Windows-сервере создать пустую базу данных 1С. Для этого запустить клиент 1С, нажать «Добавить»

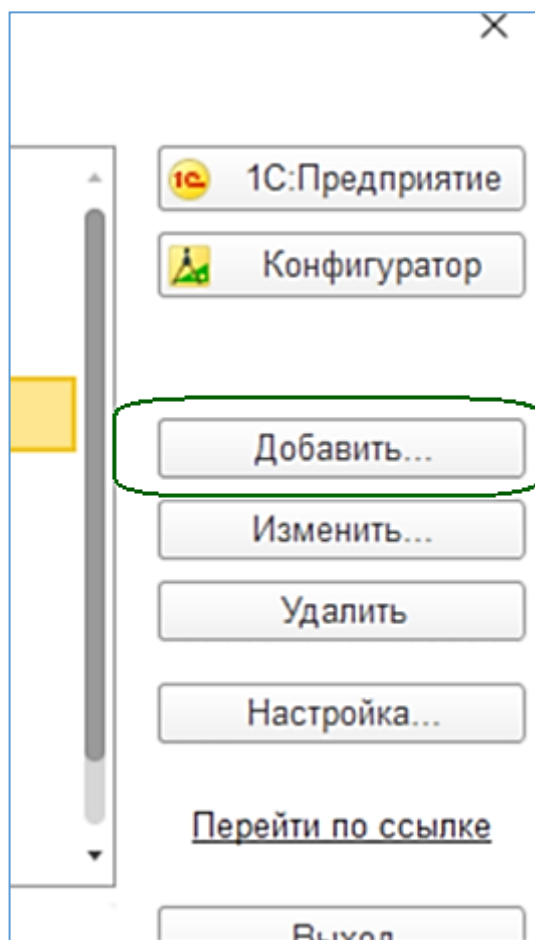


Рисунок 8.1 – Кнопка добавления БД

- Выбрать «Создание новой информационной базы» и нажать кнопку «Далее»;
- Выбрать «Создание информационной базы без конфигурации...» и нажать кнопку «Далее»;
- Задать название будущей базы данных, выбрать «На сервере 1С-предприятия» и нажать кнопку «Далее»;

Добавление информационной базы/группы

Укажите наименование информационной базы:

kip_compress

Выберите тип расположения информационной базы:

☐ На данном компьютере или на компьютере в локальной сети

☒ На сервере 1С:Предприятия

< Назад Далее > Отмена

Рисунок 8.2 – Окно «Добавление информационной базы»

- Задать настройки БД в 1С-кластере и для подключения к серверу СУБД:
 - кластер серверов: localhost;
 - имя базы в кластере: название БД в кластере 1С;
 - тип СУБД: PostgreSQL;
 - сервер баз данных: адрес сервера СУБД;
 - имя базы данных: с каким названием БД будет создана на сервере СУБД;
 - имя пользователя и пароль для доступа к серверу СУБД
- Дважды нажать кнопку «Далее»;

Добавление информационной базы/группы

Укажите параметры информационной базы:

Кластер серверов 1С:Предприятия: localhost

Имя информационной базы в кластере: kip_compress

Защищенное соединение: Выключено

Тип СУБД: PostgreSQL

Сервер баз данных: ip-адрес сервера

Имя базы данных: kip_compress

Пользователь базы данных: postgres

Пароль пользователя:

Смещение дат: 0

☒ Создать базу данных в случае ее отсутствия

Язык (Страна): русский (Россия)

☐ Установить блокировку регламентных заданий

< Назад Далее > Отмена

Рисунок 8.3 – Окно «Добавление информационной базы»

- В клиенте 1С выбрать созданную БД, нажать кнопку «Конфигуратор»;

Запуск 1С:Предприятия

Информационные базы

- 1С-К-ТС
- kamaz3
- kip_compress**
- kip_compress_pgpro
- kip_lw
- kip_LW3
- test_mvar
- Информационная база
- Информационная база #1

Srv="localhost";Ref="kip_compress";

1С:Предприятие

Конфигуратор

Добавить...

Изменить...

Удалить

Настройка...

[Перейти по ссылке](#)

Выход

Рисунок 8.4 – Окно клиента 1С

- В открывшемся Конфигураторе выбрать Администрирование и опцию «Загрузить информационную базу»;

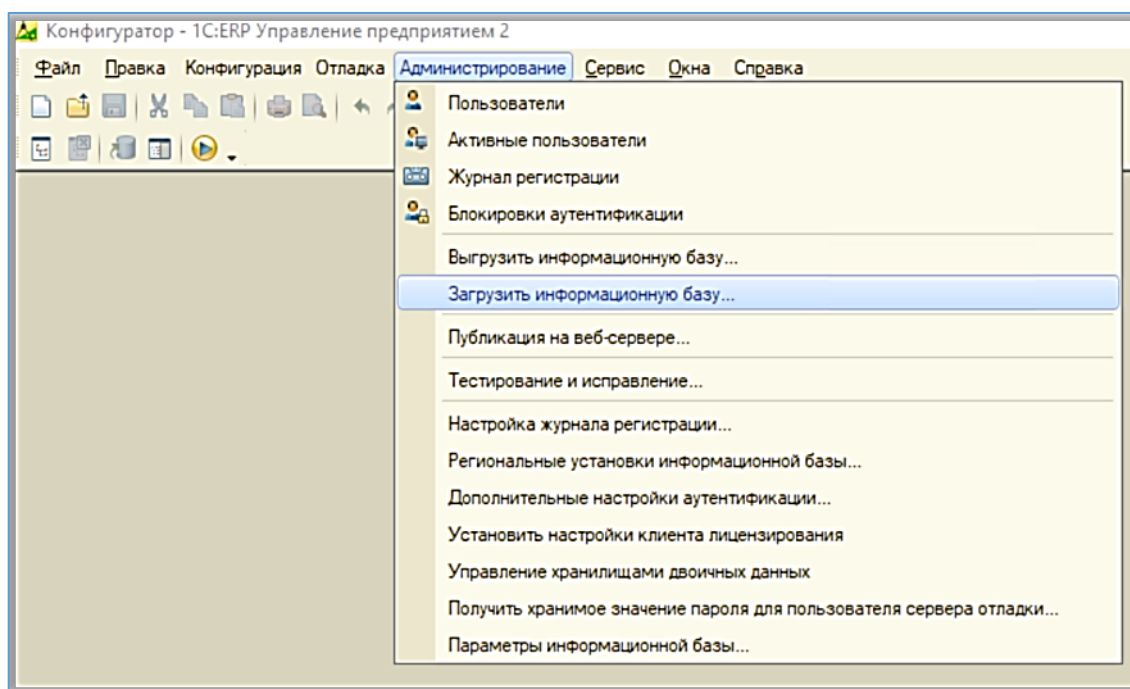


Рисунок 8.5 – Опция «Загрузить информационную базу»

- Выбрать файл дампа конфигурации и подтвердить загрузку дампа;
- По окончании загрузки можно закрыть Конфигуратор;
- На сервере СУБД использовать клиент `psql` для входа в СУБД;

```
sudo -u postgres psql
```

- В терминале ОС создать каталог для сжатого табличного пространства;



В представленном примере используется каталог `/tmp/tsp`. Для реальной БД следует выбрать более подходящее размещение

```
\! mkdir /tmp/tsp
```

- В СУБД создать табличное пространство со сжатием (`zstd` или `lz4`);

```
CREATE TABLESPACE tsp  
location '/tmp/tsp' with (compression=[zstd|lz4]);
```



Внимание, каталог `/tmp` может периодически чиститься демоном `systemd-tmpfiles-clean.timer`.

Чтобы посмотреть его статус используйте команду ниже, либо создавайте `tsp` в другом каталоге, чтобы избежать внезапной очистки.

```
#systemctl status systemd-tmpfiles-clean.timer
```

- Убедиться, что нет нагрузки на целевую БД;
- Перенести БД в новое табличное пространство SQL-командой:

```
ALTER DATABASE kip_compress set tablespace tsp;
```

- После окончания переноса перейти в целевую БД:

```
\c kip_compress
```

- Выполнить команды:

```
vacuum full;  
reindex database;  
checkpoint;
```

8.3. Запуск тестов в 1С КИП (Тест-центр)

Выполнение теста требует выполнения следующих шагов:

- На сервере под управление ОС Windows запустить клиент 1С, войти в базу нажатием кнопки 1С:Предприятие и ввести аутентификационную информацию администратора СУБД;

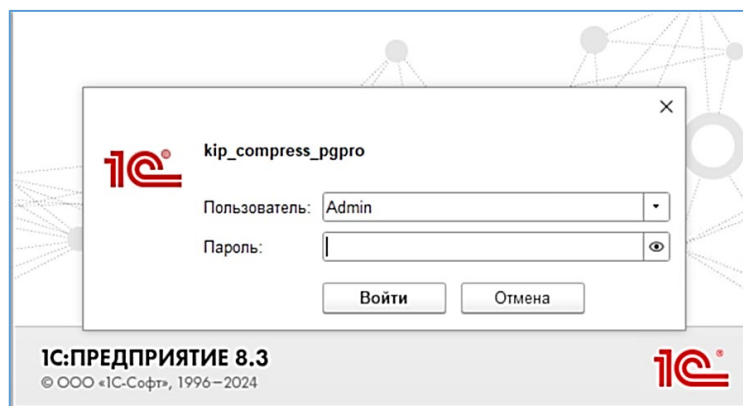


Рисунок 8.6 – Окно аутентификации

- В левом меню перейти в опции «Тест-центр» - «Агенты»;
- Если в списке уже есть значения, то следует нажать кнопку «Выгрузить» и дождаться окончания выгрузки;

- Нажать кнопку «Включить режим агента» и дождаться состояния «Подключен»;

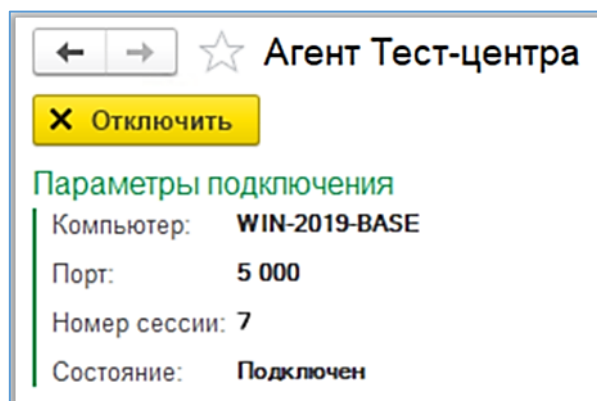


Рисунок 8.7 – Агент Тест-центра

- Не закрывая это окно 1С, запустить новый клиент, войти под тем же пользователем;
- В левом меню перейти в опции «Тест-центр» - «Сценарии тестирования»;
- Выбрать предпочтительный сценарий и нажать «Настроить запуск»;

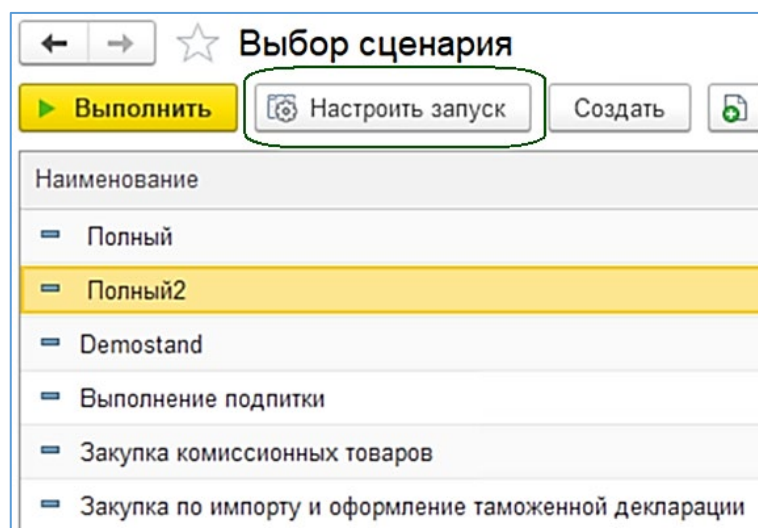


Рисунок 8.8 – Выбор сценария тестирования

- Нажать кнопку «Заполнить по запущенным агентам» (должен появиться агент). При необходимости изменить настройки тестов (количество виртуальных рабочих мест, итераций и пр.). Нажать «Запустить тест»;

Настройка запуска теста

▶ Запустить тест

Сценарий: Полный2

Общее количество одновременно запускаемых BPM: 0

Количество одновременно запускаемых BPM агента: 10

Количество пользователей

Начальное количество: 22

Добавить после удачной итерации: 4

Добавить после неудачной итерации: 2

Динамическое добавление: ☐

Коэффициент интенсивности: 1,00

Ограничения

Максимальное количество итераций: 1

Пороговый APDEX: 0,000

Критическое количество ошибок теста: 0

Критическое количество ошибок на итерации: 1

Расширенные настройки

Распределение BPM по таблице весов:

Добавить ↑ ↓ Заполнить по запущенным агентам

Компьютер
WIN-2019-BASE

Рисунок 8.9 – Окно «Настройка запуска тестов»

— После чего будет выведено окно с результатами тестов.

Результат теста 000000060 от 22.11.2024 9:03:41

Сценарий: Сервисные и регламентные операции | Дата окончания: 22.11.2024 9:40:27

Результат: Выполнено успешно | Состояние: Неактивен | Прерван

Производительность | Показатели | Протокол | Параметры выполнения | Производительность итераций | Структура итераций | Описание | Сводный отчет

Результаты замеров производительности по итерациям теста

Итерация	Приоритет	Целевое время	Количество замеров	APDEX	Сре
Итерация №1. BPM: 20		1,00	408	0,755	
Ключевая операция					
Внеоборотные активы. формирование заданий к закрытию месяца		1,00	8	1,000	
Внеоборотные активы. формирование заданий к закрытию месяца. удельный		1,00	8	1,000	
Обработка. помощник продаж. форма. форма. соглашение с клиентом. при изменении		1,00	4	0,500	
Обработка. помощник продаж. форма. элемент. партнер. при изменении		1,00	4	0,500	
Обработка. помощник продаж. форма. элемент. соглашение. при изменении		1,00	4	0,500	
Общее время запуска приложения		1,00	20		
Отчет. ведомость по товарам на складах в ценах номенклатуры. ведомость в ценах номенклатуры. формирование		1,00	4	0,250	
Отчет. ведомость по товарам на складах. ведомость по товарам на складах. формирование		1,00	4	0,125	
Отчет. заполненность склада. заполненность склада. формирование		1,00	4	0,500	
Отчет. карточка расчетов с клиентами. карточка расчетов с клиентами по документам контекст. формирование		1,00	4	0,500	
Отчет. контроль оформления документов товародвижений. контроль оформления документов товародвижений. формирование		1,00	4	0,125	
Отчет. оборачиваемость запасов. оборачиваемость запасов. формирование		1,00	4	0,125	
Отчет. остатки и доступность товаров. использовать назначения без заказа. формирование		1,00	4	0,500	

Рисунок 8.10 – Вывод результатов теста

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Администратор СУБД – субъект доступа, выполняющий административные функции в СУБД и наделенный правами:

- Создавать учетные записи пользователей системы управления базами данных;
- Модифицировать, блокировать и удалять учетные записи пользователей системы управления базами данных;
- Назначать права доступа пользователям системы управления базами данных к объектам доступа системы управления базами данных;
- Управлять конфигурацией системы управления базами данных;
- Создавать, подключать базы данных.

Администратор СУБД имеет атрибут SUPERUSER и/или обладает системной учетной записью «postgres».

Алгоритм адресации блока – у блока есть номер, по нему вычисляется сегмент и смещение в этом сегменте с помощью нехитрой математики.

ТП (табличное пространство) — это понятие, используемое в системах управления базами данных (СУБД) для организации логического пространства, в котором совместно хранятся определённые объекты базы данных, такие как индексы, таблицы и другие. При создании табличного пространства определяется его имя, которое используется для указания на пространство в SQL-запросах.

TOAST (The Oversized Attribute Storage Technique) — это специальный механизм в PostgreSQL для хранения больших значений в отдельных таблицах («тост-таблицах»). Он используется для хранения сложных типов данных, таких как текст, бинарные последовательности, JSONB объекты и другие. TOAST работает автоматически и не требует вмешательства пользователя, но имеет некоторые ограничения, связанные с размером таблиц и количеством значений.

GUID (Globally Unique Identifier) — это статистически уникальный 128-битный идентификатор. Главная особенность GUID — уникальность, которая позволяет создавать расширяемые сервисы и приложения без риска конфликтов. GUID записывается в виде

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

строки из тридцати двух шестнадцатеричных цифр, разделённых дефисами и опционально заключённых в фигурные скобки.

Zstd — это алгоритм сжатия данных без потерь, разработанный Яном Колле в 2015 году. Он сочетает словарный алгоритм сжатия данных типа LZ77 и эффективное энтропийное кодирование типа ANS (FSE). Алгоритм предназначен для достижения коэффициентов сжатия, сопоставимых или превосходящих классический алгоритм deflate, при более высокой скорости сжатия и распаковки.

LZ4 — алгоритм сжатия данных без потерь, ориентированный на высокую скорость сжатия и распаковки. Он относится к семейству методов сжатия LZ77, работающих с байтовыми потоками. Отличается компактным кодом для распаковки.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

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

[illegible]

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