Практика

1. Создайте нежурналируемую таблицу в пользовательском табличном пространстве и убедитесь, что для таблицы существует слой init.

   Удалите созданное табличное пространство.

2. Создайте таблицу со столбцом типа text.
   Какая стратегия хранения применяется для этого столбца?

   Измените стратегию на external и вставьте в таблицу короткую и длинную строки.

   Проверьте, попали ли строки в toast-таблицу, выполнив прямой запрос к ней. Объясните, почему.

1. Нежурналируемая таблица

```
student$ sudo -u postgres mkdir /var/lib/postgresql/ts_dir
```

```
=> CREATE TABLESPACE ts LOCATION '/var/lib/postgresql/ts_dir';
```

```
CREATE TABLESPACE
```

```
=> CREATE DATABASE data_lowlevel;
```

```
CREATE DATABASE
```

```
=> \c data_lowlevel
```

```
You are now connected to database "data_lowlevel" as user "student".
```

```
=> CREATE UNLOGGED TABLE u(n integer) TABLESPACE ts;
```

```
CREATE TABLE
```

```
=> INSERT INTO u(n) SELECT n FROM generate_series(1,1000) n;
```

```
INSERT 0 1000
```

```
=> SELECT pg_relation_filepath('u');
```

```
            pg_relation_filepath
-----------------------------------------
 pg_tblspc/16386/PG_16_202307071/16387/16388
(1 row)
```

Посмотрим на файлы таблицы.

Обратите внимание, что следующая команда ls выполняется от имени пользователя postgres. Чтобы повторить такую команду, удобно сначала открыть еще одно окно терминала и переключиться в нем на другого пользователя командой:

```
student$ sudo -i -u postgres
```

И затем в этом же окне выполнить:

```
postgres$ ls -l /var/lib/postgresql/16/main/pg_tblspc/16386/PG_16_202307071/16387/16388*
```

```
-rw------- 1 postgres postgres 40960 июн 23 23:33
/var/lib/postgresql/16/main/pg_tblspc/16386/PG_16_202307071/16387/16388
-rw------- 1 postgres postgres 24576 июн 23 23:33
/var/lib/postgresql/16/main/pg_tblspc/16386/PG_16_202307071/16387/16388_fsm
-rw------- 1 postgres postgres     0 июн 23 23:33
/var/lib/postgresql/16/main/pg_tblspc/16386/PG_16_202307071/16387/16388_init
```

Удалим созданное табличное пространство:

```
=> DROP TABLE u;
```

```
DROP TABLE
```

```
=> DROP TABLESPACE ts;
```

```
DROP TABLESPACE
```

```
student$ sudo -u postgres rm -rf /var/lib/postgresql/ts_dir
```

2. Таблица с текстовым столбцом

```
=> CREATE TABLE t(s text);
```

```
CREATE TABLE
```

```
=> \d+ t
```

```
                                        Table "public.t"
 Column | Type | Collation | Nullable | Default | Storage  | Compression | Stats target |
Description
--------+------+-----------+----------+---------+----------+-------------+--------------+-
------------
 s      | text |           |          |         | extended |             |              |
Access method: heap
```

По умолчанию для типа text используется стратегия extended.

Изменим стратегию на external:

```
=> ALTER TABLE t ALTER COLUMN s SET STORAGE external;
```

```
ALTER TABLE
```

```
=> INSERT INTO t(s) VALUES ('Короткая строка.');
```

```
INSERT 0 1
```

```
=> INSERT INTO t(s) VALUES (repeat('A',3456));
```

```
INSERT 0 1
```

Проверим toast-таблицу:

```
=> SELECT relname FROM pg_class WHERE oid = (
  SELECT reltoastrelid FROM pg_class WHERE relname='t'
);
```

```
    relname
---------------
 pg_toast_16391
(1 row)
```

Toast-таблица «спрятана», так как находится в схеме, которой нет в пути поиска. И это правильно, поскольку TOAST работает прозрачно для пользователя. Но заглянуть в таблицу все-таки можно:

```
=> SELECT chunk_id, chunk_seq, length(chunk_data)
FROM pg_toast.pg_toast_16391
ORDER BY chunk_id, chunk_seq;
```

```
 chunk_id | chunk_seq | length
----------+-----------+--------
    16396 |         0 |   1996
    16396 |         1 |   1460
(2 rows)
```

Видно, что в TOAST-таблицу попала только длинная строка (два фрагмента, общий размер совпадает с длиной строки). Короткая строка не вынесена в TOAST просто потому, что в этом нет необходимости — версия строки и без этого помещается в страницу.

Практика+

1. Создайте базу данных.

   Сравните размер базы данных, возвращаемый функцией pg_database_size, с общим размеров всех таблиц в этой базе.

   Объясните полученный результат.
2. В TOAST поддерживаются два метода сжатия: pglz и lz4. Проверьте средствами SQL, был ли PostgreSQL скомпилирован с поддержкой этих методов.
3. Создайте текстовый файл размером больше 10 Мбайт. Загрузите его содержимое в таблицу с текстовым полем, сначала без сжатия, а затем используя каждый из алгоритмов. Сравните итоговый размер таблицы и время загрузки данных для трех вариантов.

1. Список таблиц базы данных можно получить из таблицы pg_class системного каталога.

2. С помощью представления pg_config можно узнать, какие параметры были установлены скрипту configure при сборке ПО сервера. Строка, содержащая список параметров, велика; выделить необходимые параметры можно функцией string_to_table().

3. Чтобы получить текст для эксперимента, можно взять достаточно большой двоичный файл (например, исполняемый файл postgres) и преобразовать его в текст. Для преобразования можно использовать алгоритм Base32 (ключ `-w0` отменяет переносы строк):

```
base32 -w0 < двоичный-файл > текстовый-файл
```

1. Сравнение размеров базы данных и таблиц в ней

```
=> CREATE DATABASE data_lowlevel;

CREATE DATABASE

=> \c data_lowlevel

You are now connected to database "data_lowlevel" as user "student".
```

Даже пустая база данных содержит таблицы, относящиеся к системного каталогу. Полный список отношений можно получить из таблицы pg_class. Из выборки надо исключить:

- таблицы, общие для всего кластера (они не относятся к текущей базе данных);
- индексы и toast-таблицы (они будут автоматически учтены при подсчета размера).

```
=> SELECT sum(pg_total_relation_size(oid))
FROM pg_class
WHERE NOT relisshared -- локальные объекты базы
AND relkind = 'r';  -- обычные таблицы

   sum
---------
 7536640
(1 row)
```

Размер базы данных оказывается несколько больше:

```
=> SELECT pg_database_size('data_lowlevel');

 pg_database_size
------------------
          7696867
(1 row)
```

Дело в том, что функция pg_database_size возвращает размер каталога файловой системы, а в этом каталоге находятся несколько служебных файлов.

```
=> SELECT oid FROM pg_database WHERE datname = 'data_lowlevel';

  oid
-------
 16386
(1 row)
```

Обратите внимание, что следующая команда ls выполняется от имени пользователя postgres. Чтобы повторить такую команду, удобно сначала открыть еще одно окно терминала и переключиться в нем на другого пользователя командой:

```
student$ sudo -i -u postgres
```

И затем в этом же окне выполнить:

```
postgres$ ls -l /var/lib/postgresql/16/main/base/16386/[^0-9]*

-rw------- 1 postgres postgres    524 июн 23 23:33
/var/lib/postgresql/16/main/base/16386/pg_filenode.map
-rw------- 1 postgres postgres 159700 июн 23 23:33
/var/lib/postgresql/16/main/base/16386/pg_internal.init
-rw------- 1 postgres postgres      3 июн 23 23:33
/var/lib/postgresql/16/main/base/16386/PG_VERSION
```

- pg_filenode.map — отображение oid некоторых таблиц в имена файлов;
- pg_internal.init — кеш системного каталога;
- PG_VERSION — версия PostgreSQL.

Из-за того, что одни функции работают на уровне объектов базы данных, а другие — на уровне файловой системы, бывает сложно точно сопоставить возвращаемые размеры. Это относится и к функции pg_tablespace_size.

2. Поддержка методов сжатия TOAST

Представление pg_config показывает параметры, которые были переданы скрипту configure при сборке PostgreSQL.

```
=> SELECT * FROM (
  SELECT string_to_table(setting, ''' ''') AS setting
  FROM pg_config WHERE name = 'CONFIGURE'
)
WHERE setting ~ '(lz|zs)';
```

```
   setting
-------------
 --with-lz4
 --with-zstd
(2 rows)
```

Какой метод сжатия TOAST используется по умолчанию?

```
=> \dconfig *toast*
```

```
 List of configuration parameters
         Parameter         | Value
---------------------------+-------
 default_toast_compression | pglz
(1 row)
```

Какие методы можно применять?

```
=> SELECT setting, enumvals FROM pg_settings WHERE name = 'default_toast_compression';
```

```
 setting |  enumvals
---------+------------
 pglz    | {pglz,lz4}
(1 row)
```

3. Сравнение методов сжатия

Сравним методы сжатия на примере текстовых данных.

Чтобы получить текст большого объема, возьмем исполняемый файл postgres и преобразуем его в текст с помощью алгоритма Base 32, который применяется в электронной почте.

```
student$ sudo cat /usr/lib/postgresql/16/bin/postgres | base32 -w0 > /tmp/gram.input
```

Получившийся текстовый файл имеет достаточный размер.

```
student$ ls -l --block-size=K /tmp/gram.input
```

```
-rw-rw-r-- 1 student student 16379K июн 23 23:33 /tmp/gram.input
```

Создадим таблицу для загрузки текстовых данных.

Для столбца txt установим стратегию хранения EXTERNAL, которая допускает отдельное хранение, но не сжатие.

```
=> CREATE TABLE t (
  txt text STORAGE EXTERNAL
);
```

```
CREATE TABLE
```

Загрузим данные из текстового файла.

```
=> \timing on
```

```
Timing is on.
```

```
=> COPY t FROM '/tmp/gram.input';
```

```
COPY 1
Time: 441,252 ms
```

```
=> \timing off
```

```
Timing is off.
```

Проверим размер таблицы, включая TOAST.

```
=> SELECT pg_table_size('t')/1024;
```

```
 ?column?
----------
    17040
(1 row)
```

Опустошим таблицу и зададим сжатие с помощью pglz.

```
=> TRUNCATE TABLE t;
```

```
TRUNCATE TABLE
```

```
=> ALTER TABLE t
ALTER COLUMN txt SET STORAGE EXTENDED,
ALTER COLUMN txt SET COMPRESSION pglz;
```

```
ALTER TABLE
```

Теперь используется стратегия EXTENDED, которая допускает как сжатие, так и отдельное хранение.

Снова загрузим данные.

```
=> \timing on
```

```
Timing is on.
```

```
=> COPY t FROM '/tmp/gram.input';
```

```
COPY 1
Time: 830,392 ms
```

```
=> \timing off
```

```
Timing is off.
```

```
=> SELECT pg_table_size('t')/1024;
```

```
 ?column?
----------
    10368
(1 row)
```

Размер таблицы значительно уменьшился, но при этом заметно выросло время загрузки.

Снова опустошим таблицу и зададим теперь сжатие с помощью lz4.

```
=> TRUNCATE TABLE t;
```

```
TRUNCATE TABLE
```

```
=> ALTER TABLE t ALTER COLUMN txt SET COMPRESSION lz4;
```

```
ALTER TABLE
```

Еще раз загрузим данные и сравним.

```
=> \timing on
```

```
Timing is on.
```

```
=> COPY t FROM '/tmp/gram.input';
```

```
COPY 1
Time: 293,715 ms
```

```
=> \timing off
```

```
Timing is off.
```

```
=> SELECT pg_table_size('t')/1024;
```

```
 ?column?
----------
    10712
(1 row)
```

Алгоритм lz4 слегка уступает pglz по степени сжатия, однако работает значительно быстрее.

Удалим текстовый файл.

```
student$ sudo rm -f /tmp/gram.input
```