

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВЕРО–КАВКАЗСКАЯ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ПЕРСПЕКТИВНОЙ ИНЖЕНЕРИИ
ДЕПАРТАМЕНТ ЦИФРОВЫХ, РОБОТОТЕХНИЧЕСКИХ СИСТЕМ И
ЭЛЕКТРОНИКИ
МЕЖИНСТИТУТСКАЯ БАЗОВАЯ КАФЕДРА

Дисциплина: Тестирование и отладка программного обеспечения

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

Выполнил:

студент 4 курса направления
подготовки 09.03.04 «Программная
инженерия», направленность
«Разработка и сопровождение
программного обеспечения»

группы ПИЖ-б-о-22-1

Хетагуров Тамерлан Аланович

(Подпись)

Проверил:

Ассистент департамента цифровых,
робототехнических систем и
электроники

Щеголев Алексей Алексеевич

(Подпись)

Работа защищена с оценкой:

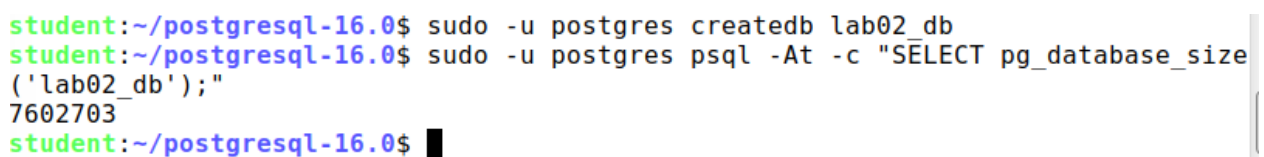
Ставрополь, 2025г.

Цель: Всестороннее изучение логической и физической структуры хранения данных в PostgreSQL. Получение практических навыков управления базами данных, схемами, табличными пространствами. Глубокое освоение работы с системным каталогом для извлечения метайнформации. Исследование низкоуровневых аспектов хранения, включая TOAST.

Модуль 1: Базы данных и схемы

1. Создание и проверка БД: Создайте новую базу данных lab02_db.

Проверьте ее начальный размер с помощью `pg_database_size('lab02_db')`.



```
student:~/postgresql-16.0$ sudo -u postgres createdb lab02_db
student:~/postgresql-16.0$ sudo -u postgres psql -At -c "SELECT pg_database_size('lab02_db');"
7602703
student:~/postgresql-16.0$
```

Рисунок 1 – Создание и проверка базы данных

2. Работа со схемами: Подключитесь к lab02_db. Создайте две схемы: app и схему с именем вашего пользователя ОС (напр., student). В каждой схеме создайте по одной таблице и вставьте в них данные.

```

student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db
psql (16.10 (Ubuntu 16.10-1.pgdg24.04+1))
Type "help" for help.

lab02_db=# ^C
lab02_db=# CREATE SCHEMA app;
CREATE SCHEMA
lab02_db=# CREATE SCHEMA student;
CREATE SCHEMA
lab02_db=# CREATE TABLE app.products (id serial PRIMARY KEY, name text, price numeric);
CREATE TABLE
lab02_db=# CREATE TABLE student.notes (id serial PRIMARY KEY, txt text);
CREATE TABLE
lab02_db=# INSERT INTO app.products (name, price) VALUES ('apple', 1.2), ('book', 12.5);
INSERT 0 2
lab02_db=# INSERT INTO student.notes (txt) VALUES ('note1'), ('note2');
INSERT 0 2
lab02_db=# SELECT * FROM app.products;
 id | name  | price
-----+-----+-----
  1 | apple |   1.2
  2 | book  |  12.5
(2 rows)

lab02_db=# SELECT * FROM student.notes;
 id | txt
-----+-----
  1 | note1
  2 | note2
(2 rows)

lab02_db=# █

```

Рисунок 2 – Создание схем и баз данных в них

3. Контроль размера: Снова проверьте размер базы данных. Объясните его изменение.

```

student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db -c "SELECT pg_database_size('lab02_db') AS size_bytes, pg_size_pretty(pg_database_size('lab02_db'));"
 size_bytes | pg_size_pretty
-----+-----
  7909859 | 7724 kB
(1 row)

student:~/postgresql-16.0$ █

```

Рисунок 3 – Проверка размера базы данных

Размер файла вырос из-за добавления новых таблиц и данных в них.

4. Управление путем поиска: Настройте параметр `search_path` для текущего сеанса так, чтобы при обращении по неполному имени

приоритет имела ваша пользовательская схема, а затем схема app. Продемонстрируйте работу, обратившись к таблицам без указания схемы.

```
student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db
psql (16.10 (Ubuntu 16.10-1.pgdg24.04+1))
Type "help" for help.

lab02_db=# SHOW search_path;
      search_path
-----
 "$user", public
(1 row)

lab02_db=# SET search_path = student, app, public;
SET
lab02_db=# SELECT * FROM products;
 id | name  | price
-----+-----+-----
  1 | apple |   1.2
  2 | book  |  12.5
(2 rows)

lab02_db=# SELECT * FROM notes;
 id | txt
-----+-----
  1 | note1
  2 | note2
(2 rows)

lab02_db=# RESET search_path;
RESET
lab02_db=# █
```

Рисунок 4 – Обращение к таблице без указания схемы

5. Практика+ (Настройка параметра БД): Для базы lab02_db установите значение параметра temp_buffers так, чтобы в каждом новом сеансе, подключенном к этой БД, оно было в 4 раза больше значения по умолчанию. Проверьте работу.

```

student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db -c "SHOW temp_buffers;"
temp_buffers
-----
8MB
(1 row)

student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db
psql (16.10 (Ubuntu 16.10-1.pgdg24.04+1))
Type "help" for help.

lab02_db=# \q
student:~/postgresql-16.0$ sudo -u postgres psql
psql (16.10 (Ubuntu 16.10-1.pgdg24.04+1))
Type "help" for help.

postgres=# ALTER DATABASE lab02_db SET temp_buffers = '32MB';
ALTER DATABASE
postgres=# █

```

Рисунок 5 – Получение и увеличение temp_buffers

```

student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db -c "SELECT * FROM pg_db_role_setting WHERE setdatabase = (SELECT oid FROM pg_database WHERE datname = 'lab02_db');"
setdatabase | setrole |      setconfig
-----+-----+-----
16390 | 0 | {temp_buffers=32MB}
(1 row)

student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db -c "SHOW temp_buffers;"
temp_buffers
-----
32MB
(1 row)

student:~/postgresql-16.0$ █

```

Рисунок 6 – Проверка нового значения temp_buffers

Модуль 2: Системный каталог

1. Исследование pg_class: Получите описание системной таблицы pg_class (команда \d pg_class).

Table "pg_catalog.pg_class"				
Column	Type	Collation	Nullable	Default
oid	oid		not null	
relname	name		not null	
relnamespace	oid		not null	
reltype	oid		not null	
reloftype	oid		not null	
relowner	oid		not null	
relam	oid		not null	
relfilenode	oid		not null	
reltablespace	oid		not null	
relpages	integer		not null	
reltuples	real		not null	
relallvisible	integer		not null	
reltoastrelid	oid		not null	
relhasindex	boolean		not null	
relisshared	boolean		not null	
relpersistence	"char"		not null	
relkind	"char"		not null	
relnatts	smallint		not null	
relchecks	smallint		not null	
relhasrules	boolean		not null	
relhastriggers	boolean		not null	
relhassubclass	boolean		not null	
relrowsecurity	boolean		not null	
relforcerowsecurity	boolean		not null	
relispopulated	boolean		not null	
relreplident	"char"		not null	
relispartition	boolean		not null	
relrewrite	oid		not null	
relfrozenxid	xid		not null	
relminmxid	xid		not null	
relacl	aclitem[]			
reloptions	text[]	C		
relpartbound	pg_node_tree	C		
Indexes:				
"pg_class_oid_index" PRIMARY KEY, btree (oid)				

Рисунок 7 – Системная таблица pg_class

- Исследование pg_tables: Получите подробное описание представления pg_tables (команда \d+ pg_tables). Объясните разницу между таблицей и представлением.

```
psql (16.10 (Ubuntu 16.10-1.pgdg24.04+1))
Type "help" for help.
```

```
lab02_db=# \d+ pg_tables
```

View "pg_catalog.pg_tables"						
Column	Type	Collation	Nullable	Default	Storage	Description
schemaname	name				plain	
tablename	name				plain	
tableowner	name				plain	
tablespace	name				plain	
hasindexes	boolean				plain	
hasrules	boolean				plain	
hastriggers	boolean				plain	
rowsecurity	boolean				plain	

```
View definition:
```

```
SELECT n.nspname AS schemaname,
       c.relname AS tablename,
       pg_get_userbyid(c.relowner) AS tableowner,
       t.spcname AS tablespace,
       c.relhasindex AS hasindexes,
       c.relhasrules AS hasrules,
       c.relhastriggers AS hastriggers,
       c.relrowsecurity AS rowsecurity
FROM pg_class c
     LEFT JOIN pg_namespace n ON n.oid = c.relnamespace
     LEFT JOIN pg_tablespace t ON t.oid = c.reltablespace
WHERE c.relkind = ANY (ARRAY['r'::"char", 'p'::"char"]);
```

```
lab02_db=#
```

Рисунок 8 – Таблица представления pg_tables

pg_tables — это представление над системными каталогами; таблица — физическая структура данных, представление — виртуальная, вычисляемая при обращении.

3. Временная таблица и список схем: в базе lab02_db создайте временную таблицу. Получите полный список всех схем в этой БД, включая системные (pg_catalog, information_schema). Объясните наличие временной схемы.

```

lab02_db=# CREATE TEMP TABLE tmp_example (id serial PRIMARY KEY, v text);
CREATE TABLE
lab02_db=# INSERT INTO tmp_example (v) VALUES ('a'),('b');
INSERT 0 2
lab02_db=# SELECT nspname FROM pg_namespace ORDER BY nspname;
          nspname
-----
 app
information_schema
pg_catalog
pg_temp_11
pg_toast
pg_toast_temp_11
public
student
(8 rows)

lab02_db=# SELECT nspname, oid FROM pg_namespace WHERE nspname LIKE 'pg_temp%';
 nspname | oid
-----+-----
 pg_temp_11 | 16411
(1 row)

lab02_db=# █

```

Рисунок 9 – Создание временной таблицы

Временная схема создаётся автоматически для временных объектов и привязана к сессии; она помогает изолировать temp-таблицы.

4. Представления information_schema: Получите список всех представлений в схеме information_schema.

table_name
pg_foreign_data_wrappers
pg_foreign_servers
pg_foreign_table_columns
pg_foreign_tables
pg_user_mappings
administrable_role_authorizations
applicable_roles
attributes
character_sets
check_constraint_routine_usage
check_constraints
collation_character_set_applicability
collations
column_column_usage
column_domain_usage
column_options
column_privileges
column_udt_usage
columns
constraint_column_usage
constraint_table_usage
data_type_privileges
domain_constraints
domain_udt_usage
domains
element_types
enabled_roles
foreign_data_wrapper_options
foreign_data_wrappers
foreign_server_options
foreign_servers
foreign_table_options
foreign_tables
information_schema_catalog_name
key_column_usage
parameters
:

Рисунок 10 – Представления в схеме information_schema

- Анализ метакоманды: Выполните в psql команду \d+ pg_views. Изучите вывод и объясните, какие запросы к системному каталогу скрыты за этой командой.

```
lab02_db=# \d+ pg_views
```

View "pg_catalog.pg_views"							
Column	Type	Collation	Nullable	Default	Storage	Description	
schemaname	name				plain		
viewname	name				plain		
viewowner	name				plain		
definition	text				extended		

```
View definition:
SELECT n.nspname AS schemaname,
       c.relname AS viewname,
       pg_get_userbyid(c.relowner) AS viewowner,
       pg_get_viewdef(c.oid) AS definition
FROM pg_class c
     LEFT JOIN pg_namespace n ON n.oid = c.relnamespace
WHERE c.relkind = 'v'::"char";

lab02_db=#
```

Рисунок 11 – Выполнение \d+ pg_views

\d+ pg_views показывает метаданные представления pg_views; за ним скрываются запросы к системному каталогу, например чтение из pg_class, pg_namespace и pg_proc — т.е. \d+ показывает структуру и определение view.

Модуль 3: Табличные пространства

1. Создание Tablespace: Создайте каталог в файловой системе. Создайте новое табличное пространство lab02_ts, указывающее на этот каталог.

```
student:~/postgresql-16.0$ sudo mkdir -p /var/lib/postgresql/mytablespace
student:~/postgresql-16.0$ sudo chown postgres:postgres /var/lib/postgresql/mytablespace
student:~/postgresql-16.0$ sudo chmod 700 /var/lib/postgresql/mytablespace
student:~/postgresql-16.0$ sudo -u postgres psql
psql (16.10 (Ubuntu 16.10-1.pgdg24.04+1))
Type "help" for help.

postgres=# CREATE TABLESPACE lab02_ts LOCATION '/var/lib/postgresql/mytablespace';
postgres=#
```

Рисунок 12 – Создание новой директории и табличного пространства

2. Tablespace по умолчанию: Измените табличное пространство по умолчанию для базы данных template1 на lab02_ts. Объясните цель этого действия.

```
postgres=# ALTER DATABASE template1 SET default_tablespace = lab02_ts;
ALTER DATABASE
postgres=# SELECT datname, dattablespace FROM pg_database WHERE datname='template1';
 datname | dattablespace 
-----+-----
 template1 | 1663
(1 row)

postgres=#
```

Рисунок 13 – Изменение табличного пространства

Это задаёт `default_tablespace` для создаваемых объектов в новых БД, созданных на основе `template1`.

3. Наследование свойства: Создайте новую базу данных `lab02_db_new`.

Проверьте ее табличное пространство по умолчанию. Объясните результат.

```
postgres=# SELECT datname, pg_tablespace.spcname
FROM pg_database JOIN pg_tablespace ON pg_database.dattablespace = pg_tablespace
.oid
WHERE datname='lab02_db_new';
 datname | spcname 
-----+-----
 lab02_db_new | pg_default
(1 row)

postgres=#
```

Рисунок 14 – Проверка табличного пространства по умолчанию

Новая БД унаследует `default_tablespace`, если `template1` был изменён.

4. Символическая ссылка: Найдите в каталоге `PGDATA/pg_tblspc/` символическую ссылку, соответствующую `lab02_ts`. Куда она ведет?

```
student:~/postgresql-16.0$ sudo -u postgres ls -l $(sudo -u postgres psql -At
-c "SHOW data_directory")/pg_tblspc
total 0
lrwxrwxrwx 1 postgres postgres 32 окт  4 14:12 16424 -> /var/lib/postgresql/m
ytablespace
student:~/postgresql-16.0$
```

Рисунок 15 – Ссылка соответствующая `lab02_ts`

В pg_tblspc лежат симлинки с именами OID и через него попадает к реальной директории tablespace.

5. Удаление Tablespace: Удалите табличное пространство lab02_ts с опцией CASCADE. Объясните необходимость использования CASCADE.

```
postgres=# DROP TABLESPACE lab02_ts;  
DROP TABLESPACE  
postgres=# █
```

Рисунок 16 – Удаление табличного пространства

Удаление табличного пространства не поддерживает команду CASCADE.

6. Практика+ (Параметр Tablespace): Установите параметр random_page_cost в значение 1.1 для табличного пространства pg_default.

```
student:~/postgresql-16.0$ sudo -u postgres psql -c "ALTER SYSTEM SET random_  
page_cost = 1.1;"  
ALTER SYSTEM  
student:~/postgresql-16.0$ sudo -u postgres psql -c "SELECT pg_reload_conf();"  
"  
pg_reload_conf  
-----  
t  
(1 row)  
  
student:~/postgresql-16.0$ sudo -u postgres psql -c "SHOW random_page_cost;"  
random_page_cost  
-----  
1.1  
(1 row)  
  
student:~/postgresql-16.0$ █
```

Рисунок 17 – Установка значения для параметра random_page_cost

Модуль 4: Низкий уровень

1. Нежурналируемая таблица: Создайте нежурналируемую таблицу в пользовательском табличном пространстве. Убедитесь, что для таблицы существует слой init (файл с суффиксом _init). Удалите табличное пространство.

```

student:~/postgresql-16.0$ sudo mkdir /var/lib/postgresql/ts_unlogged
student:~/postgresql-16.0$ sudo chown postgres:postgres /var/lib/postgresql/t
s_unlogged
student:~/postgresql-16.0$ sudo chmod 700 /var/lib/postgresql/ts_unlogged
student:~/postgresql-16.0$ sudo -u postgres psql
psql (16.10 (Ubuntu 16.10-1.pgdg24.04+1))
Type "help" for help.

postgres=# CREATE TABLESPACE ts_unlogged LOCATION '/var/lib/postgresql/ts_unl
ogged';
CREATE TABLESPACE
postgres=# CREATE UNLOGGED TABLE unlog_test (id serial PRIMARY KEY, txt text)
TABLESPACE ts_unlogged;
CREATE TABLE
postgres=# INSERT INTO unlog_test (txt) SELECT md5(random()::text) FROM gener
ate_series(1,1000);
INSERT 0 1000
postgres=# █

```

Рисунок 18 – Создание нежурналируемой таблицы

2. Стратегии хранения TOAST:

Создайте таблицу со столбцом типа text. Определите стратегию хранения по умолчанию для этого столбца. Измените стратегию хранения на external. Вставьте в таблицу короткую (менее 2 КБ) и длинную (более 2 КБ) строки. Проверьте, попали ли строки в TOAST-таблицу, выполнив запрос к pg_toast.pg_toast_<oid_таблицы>. Объясните результат.

```

postgres=# CREATE TABLE t_toast (id serial PRIMARY KEY, txt text);
CREATE TABLE
postgres=# SELECT attname, attstorage FROM pg_attribute JOIN pg_class ON pg_a
ttribute.attrelid = pg_class.oid
WHERE pg_class.relname='t_toast' AND attname='txt';
 attname | attstorage
-----+-----
 txt     | x
(1 row)

postgres=# ALTER TABLE t_toast ALTER COLUMN txt SET STORAGE EXTERNAL;
ALTER TABLE
postgres=# INSERT INTO t_toast (txt) VALUES ('short text');
INSERT 0 1
postgres=# INSERT INTO t_toast (txt) VALUES (repeat('ABCDEFGH', 300));
INSERT 0 1
postgres=# SELECT oid FROM pg_class WHERE relname='t_toast';
 oid
-----
 16441
(1 row)

postgres=# SELECT reltoastrelid FROM pg_class WHERE relname='t_toast';
 reltoastrelid
-----
          16445
(1 row)

```

Рисунок 19 – Стратегии хранения TOAST

```

postgres=# SELECT count(*) FROM pg_toast.pg_toast_16441;
 count
-----
      2
(1 row)

```

Рисунок 20 – Проверка наличия строк в TOAST-таблице

TOAST - это скрытая таблица, куда PostgreSQL складывает «раздутые» значения. Имя её формируется как pg_toast.pg_toast_<OID_основной_таблицы>.

3. Практика+ (Анализ размера БД): Сравните размер базы lab02_db, возвращаемый pg_database_size, с суммой размеров всех ее пользовательских таблиц (pg_total_relation_size). Объясните расхождение.

```

postgres=# SELECT pg_size_pretty(pg_database_size('lab02_db')) AS db_size;
 db_size
-----
7732 kB
(1 row)

postgres=# SELECT pg_size_pretty(sum(pg_total_relation_size(quote_ident(schemaname)||'.'||quote_ident(tablename)))) AS sum_tables
FROM pg_tables
WHERE schemaname NOT IN ('pg_catalog','information_schema');
 sum_tables
-----
48 kB
(1 row)

postgres=# SELECT schemaname, tablename, pg_size_pretty(pg_total_relation_size(schemaname||'.'||tablename)) AS size
FROM pg_tables
WHERE schemaname NOT IN ('pg_catalog','information_schema')
ORDER BY pg_total_relation_size(schemaname||'.'||tablename) DESC;
 schemaname | tablename | size
-----+-----+-----
public      | t_toast   | 48 kB
(1 row)

postgres=# █

```

Рисунок 21 – Анализ размера БД

Расхождение присутствует из-за наличия метаданных, индексов, TOAST, свободного пространства, каталогов системы, файлов WAL/pg_wal и т.д.

4. Практика+ (Методы сжатия TOAST): Проверьте средствами SQL, был ли PostgreSQL скомпилирован с поддержкой методов сжатия pglz и lz4.

```
student:~/postgresql-16.0$ sudo -u postgres pg_config --configure 2>/dev/null
| grep -i lz4 || true
'--build=x86_64-linux-gnu' '--prefix=/usr' '--includedir=${prefix}/include'
'--mandir=${prefix}/share/man' '--infodir=${prefix}/share/info' '--sysconfdir
/etc' '--localstatedir=/var' '--disable-option-checking' '--disable-silent-r
ules' '--libdir=${prefix}/lib/x86_64-linux-gnu' '--runstatedir=/run' '--disab
le-maintainer-mode' '--disable-dependency-tracking' '--with-tcl' '--with-perl
' '--with-python' '--with-pam' '--with-openssl' '--with-libxml' '--with-libx
slt' '--mandir=/usr/share/postgresql/16/man' '--docdir=/usr/share/doc/postgres
ql-doc-16' '--sysconfdir=/etc/postgresql-common' '--datarootdir=/usr/share/'
'--datadir=/usr/share/postgresql/16' '--bindir=/usr/lib/postgresql/16/bin' '--
libdir=/usr/lib/x86_64-linux-gnu/' '--libexecdir=/usr/lib/postgresql/' '--in
cludedir=/usr/include/postgresql/' '--with-extra-version= (Ubuntu 16.10-1.pgc
g24.04+1)' '--enable-nls' '--enable-thread-safety' '--enable-debug' '--disabl
e-rpath' '--with-uuid=e2fs' '--with-gnu-ld' '--with-gssapi' '--with-ldap' '--
with-pgport=5432' '--with-system-tzdata=/usr/share/zoneinfo' 'AWK=mawk' 'MKD
IR_P=/bin/mkdir -p' 'PROVE=/usr/bin/prove' 'PYTHON=/usr/bin/python3' 'TAR=/bin
/tar' 'XSLTPROC=xsltproc --nonet' 'CFLAGS=-g -O2 -fno-omit-frame-pointer -mno
-omit-leaf-frame-pointer -flto=auto -ffat-lto-objects -fstack-protector-strong
-g -fstack-clash-protection -Wformat -Werror=format-security -fcf-protection -
fno-omit-frame-pointer' 'LDFLAGS=-Wl,-Bsymbolic-functions -flto=auto -ffat-l
to-objects -Wl,-z,relro -Wl,-z,now' '--enable-tap-tests' '--with-icu' '--with
-llvm' 'LLVM_CONFIG=/usr/bin/llvm-config-19' 'CLANG=/usr/bin/clang-19' '--with
-lz4' '--with-zstd' '--with-systemd' '--with-selinux' '--enable-dtrace' 'buil
d_alias=x86_64-linux-gnu' 'CPPFLAGS=-Wdate-time -D_FORTIFY_SOURCE=3' 'CXXFLAG
S=-g -O2 -fno-omit-frame-pointer -mno-omit-leaf-frame-pointer -flto=auto -ffa
t-lto-objects -fstack-protector-strong -fstack-clash-protection -Wformat -We
rror=format-security -fcf-protection'
```

Рисунок 22 – Проверка поддерживаемых методов сжатия

Из полученных данных видно, что PostgreSQL скомпилирован с поддержкой методов сжатия pglz, но без методов lz4

5. Практика+ (Сравнение сжатия): Создайте текстовый файл >10 МБ. Загрузите его содержимое в таблицу с текстовым полем трижды: 1) без сжатия (storage = external), 2) сжатие pglz, 3) сжатие lz4. Сравните размер таблицы и время загрузки для каждого варианта.

```

student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db -c "CREATE TABLE
compress_none (id serial, t text) WITH (toast_tuple_target=2048);"
CREATE TABLE
student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db -c "CREATE TABLE
compress_pglz (id serial, t text);"
CREATE TABLE
student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db -c "CREATE TABLE
compress_lz4 (id serial, t text);"
CREATE TABLE
student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db -c "INSERT INTO
compress_none (t) VALUES (pg_read_file('/tmp/bigtext.txt'));"
INSERT 0 1
student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db -c "INSERT INTO
compress_pglz (t) VALUES (pg_read_file('/tmp/bigtext.txt'));"
INSERT 0 1
student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db -c "INSERT INTO
compress_lz4 (t) VALUES (pg_read_file('/tmp/bigtext.txt'));"
INSERT 0 1
student:~/postgresql-16.0$ sudo -u postgres psql -d lab02_db -c "
SELECT relname AS table_name,
       pg_size_pretty(pg_total_relation_size(oid)) AS total_size
FROM pg_class
WHERE relname IN ('compress_none','compress_pglz','compress_lz4');"
"
  table_name  | total_size
-----+-----
compress_pglz | 256 kB
compress_lz4  | 256 kB
compress_none | 256 kB
(3 rows)
student:~/postgresql-16.0$ █

```

Рисунок 23 – Проверка вариантов сжатия

Процессы сжатия не дали видимого результата, так как сами таблицы пусты, а 256 кВ – минимальный возможный размер файлов.