

Министерство науки и высшего образования Российской Федерации

**Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

Межинститутская базовая кафедра

Отчет по лабораторной работе №2

дисциплины "Администрирование баз данных"

Выполнил: Душин Александр Владимирович

4 курс, группа ПИЖ-6-о-22-1

09.03.04 "Программная инженерия"

направленность (профиль) "Разработка и сопровождение программного обеспечения"

очная форма обучения

Руководитель практики: Щеголев Алексей Алексеевич

старший преподаватель департамента цифровых, робототехнических систем и электроники института перспективной инженерии

Тема работы

Организация данных и системный каталог

Цель работы

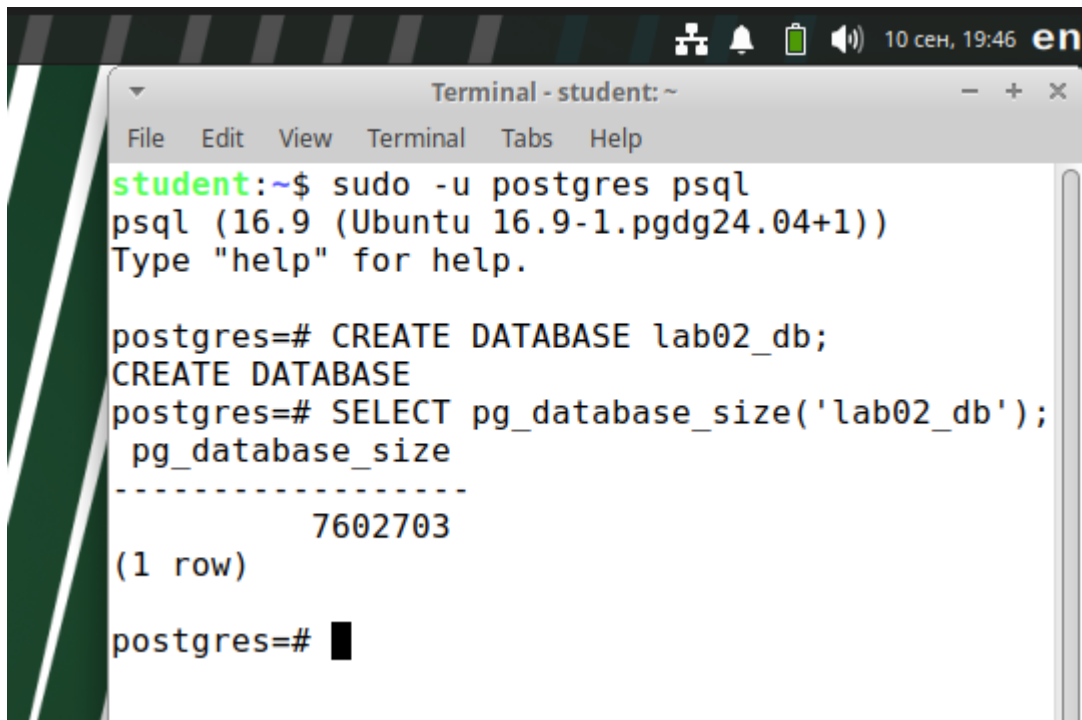
Всестороннее изучение логической и физической структуры хранения данных в PostgreSQL. Получение практических навыков управления базами данных, схемами, табличными пространствами. Глубокое освоение работы с системным каталогом для извлечения метаинформации. Исследование низкоуровневых аспектов хранения,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

Ход выполнения работы

Модуль 1. Базы данных и схемы

1. Создание и проверка БД

Создана новая база данных `lab02_db`. Проверен ее начальный размер с помощью `pg_database_size('lab02_db')`. (Рис.1)

A screenshot of a terminal window titled "Terminal - student: ~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal shows the following commands and output:

```
student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

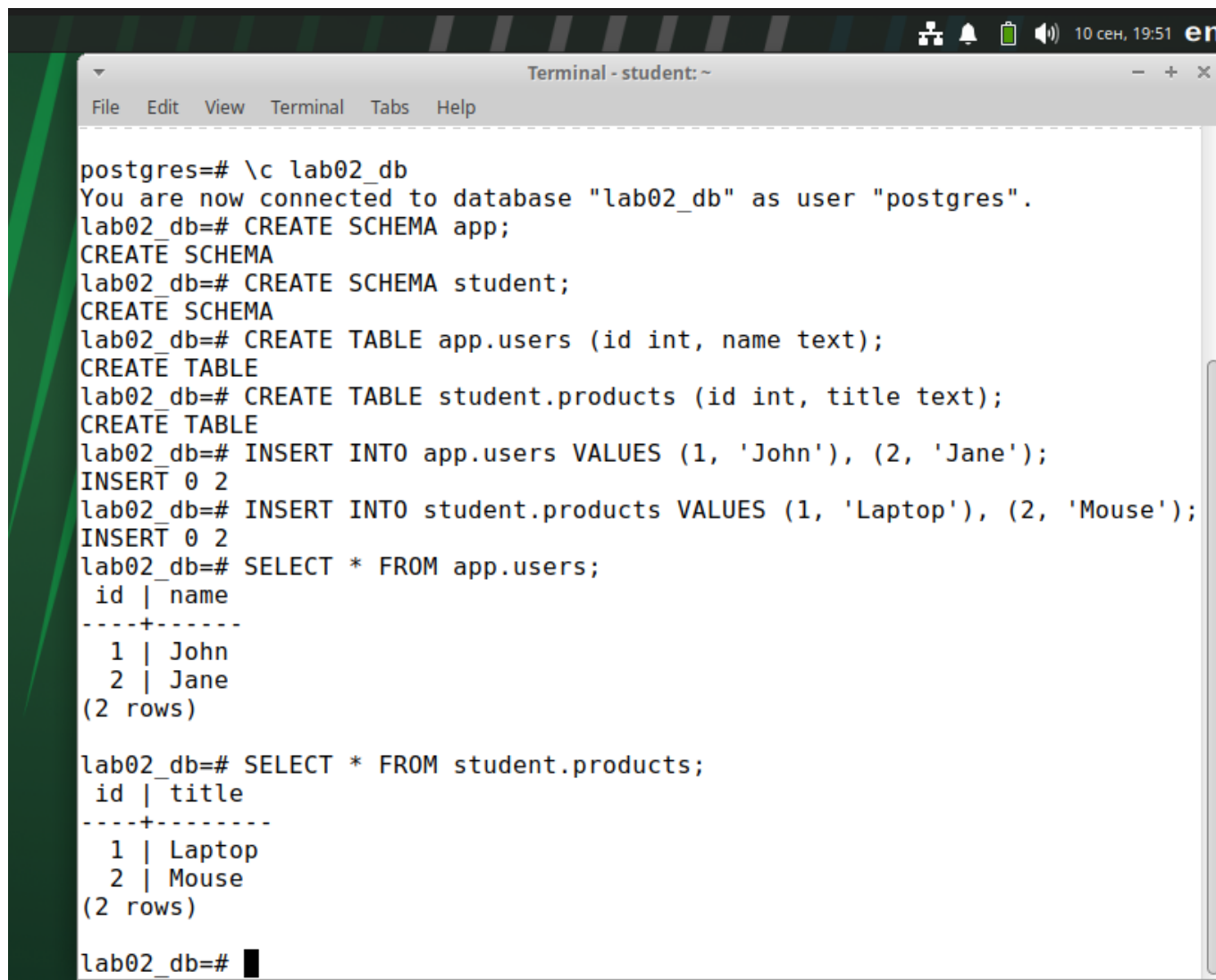
postgres=# CREATE DATABASE lab02_db;
CREATE DATABASE
postgres=# SELECT pg_database_size('lab02_db');
pg_database_size
-----
7602703
(1 row)

postgres=#
```

Рисунок 1 – Создание и проверка БД

2. Работа со схемами

Выполнено подключение к `lab02_db`. Созданы две схемы: `app` и схема с именем пользователя ОС (`student`). В каждой схеме создана таблица и вставлены данные. (Рис.2)



```
postgres=# \c lab02_db
You are now connected to database "lab02_db" as user "postgres".
lab02_db=# CREATE SCHEMA app;
CREATE SCHEMA
lab02_db=# CREATE SCHEMA student;
CREATE SCHEMA
lab02_db=# CREATE TABLE app.users (id int, name text);
CREATE TABLE
lab02_db=# CREATE TABLE student.products (id int, title text);
CREATE TABLE
lab02_db=# INSERT INTO app.users VALUES (1, 'John'), (2, 'Jane');
INSERT 0 2
lab02_db=# INSERT INTO student.products VALUES (1, 'Laptop'), (2, 'Mouse');
INSERT 0 2
lab02_db=# SELECT * FROM app.users;
 id | name
----+-----
  1 | John
  2 | Jane
(2 rows)

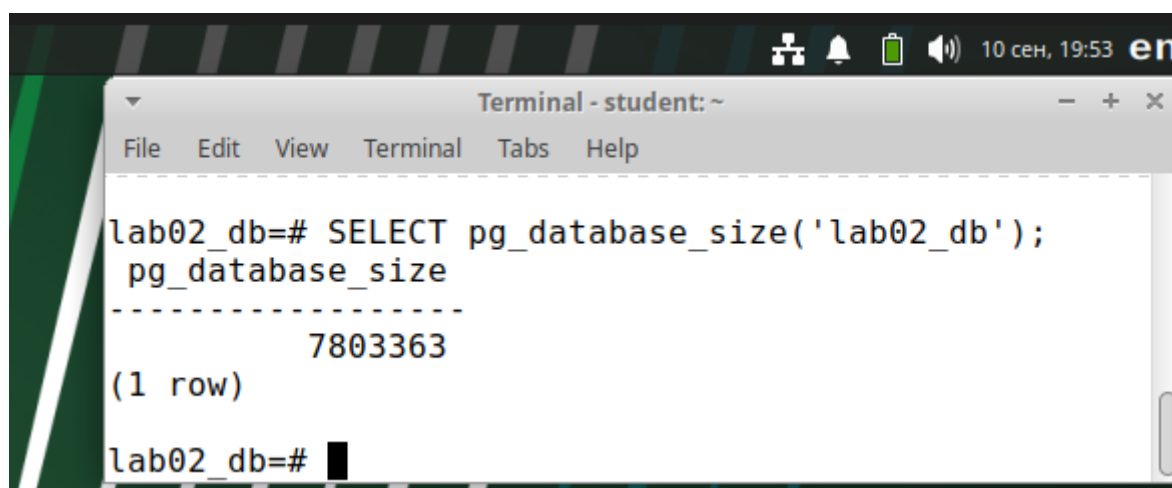
lab02_db=# SELECT * FROM student.products;
 id | title
----+-----
  1 | Laptop
  2 | Mouse
(2 rows)

lab02_db=#
```

Рисунок 2 – Работа со схемами

3. Контроль размера

Повторно проверен размер базы данных. (Рис.3)



```
lab02_db=# SELECT pg_database_size('lab02_db');
pg_database_size
-----
       7803363
(1 row)

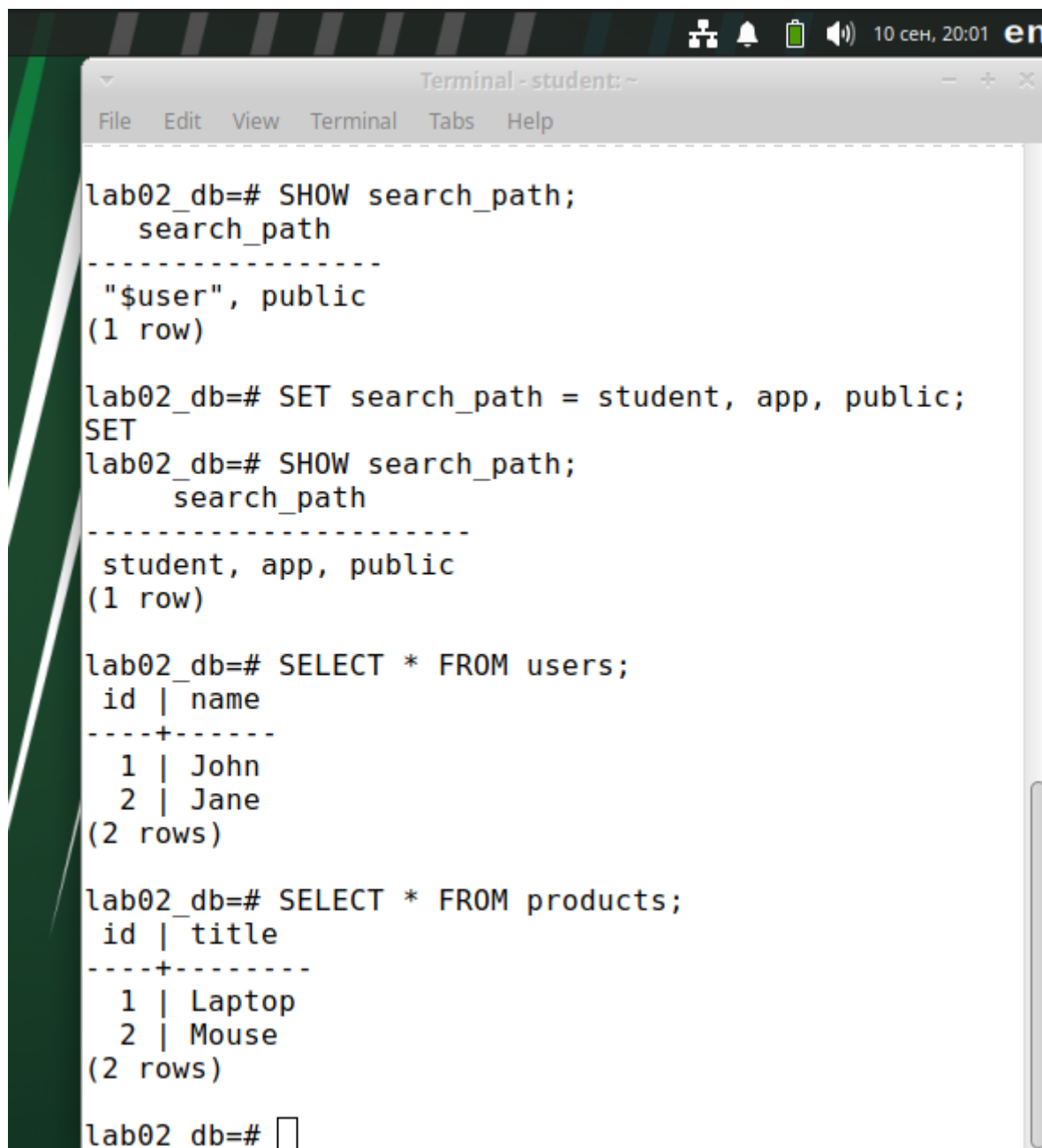
lab02_db=#
```

Рисунок 3 – Контроль размера

Размер БД увеличился незначительно из-за создания новых схем, таблиц и добавления данных. Основное увеличение связано с созданием файлов для таблиц и их метаданных в системном каталоге, а не с объемом пользовательских данных.

4. Управление путем поиска

Настроен параметр `search_path` для текущего сеанса так, чтобы при обращении по неполному имени приоритет имела пользовательская схема, а затем схема `app`. Продемонстрирована работа с таблицами без указания схемы. (Рис.4)



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

lab02_db=# SET search_path = student, app, public;
SET
lab02_db=# SHOW search_path;
search_path
-----
student, app, public
(1 row)

lab02_db=# SELECT * FROM users;
 id | name
----+-----
  1 | John
  2 | Jane
(2 rows)

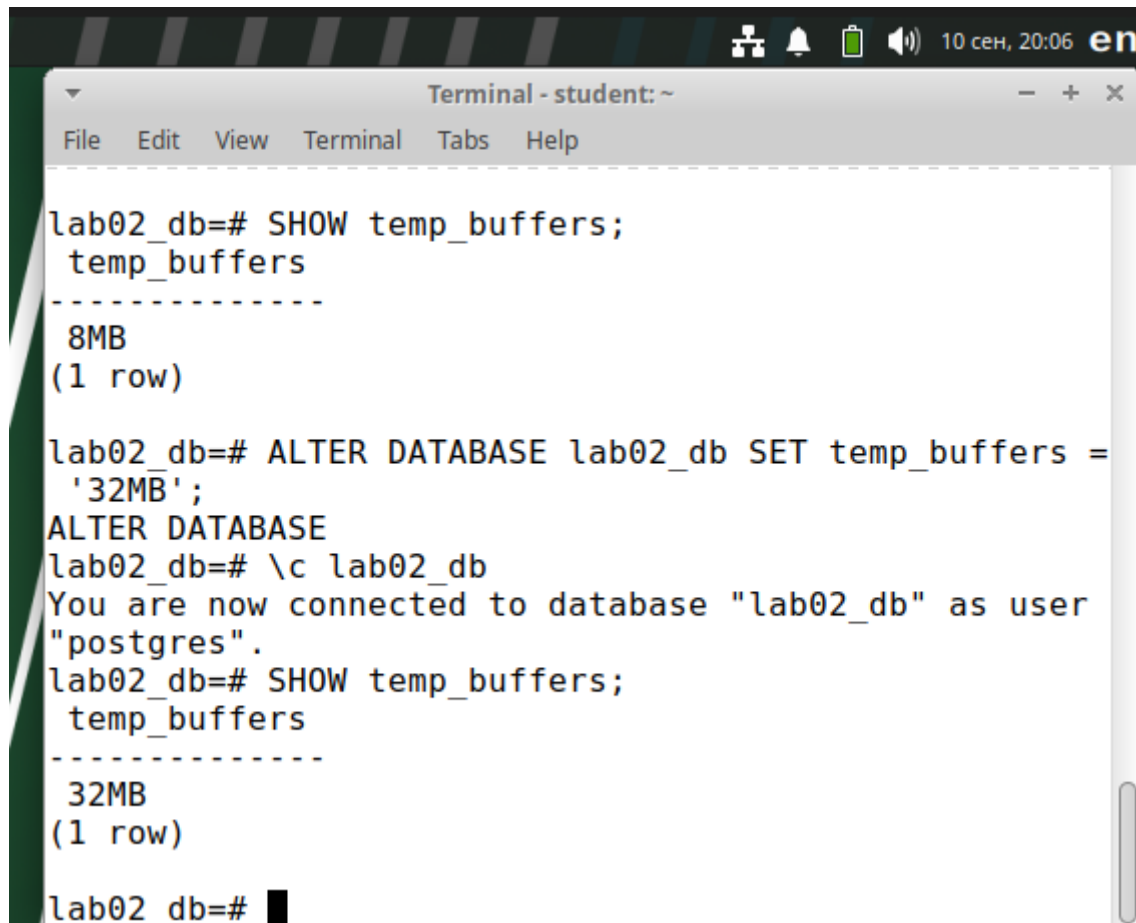
lab02_db=# SELECT * FROM products;
 id | title
----+-----
  1 | Laptop
  2 | Mouse
(2 rows)

lab02_db=#
```

Рисунок 4 – Управление путем поиска

5. Настройка параметра БД (Практика+)

Для базы `lab02_db` установлено значение параметра `temp_buffers` в 4 раза больше значения по умолчанию. Проверена работа в новом сеансе. (Рис.5)

A screenshot of a terminal window titled "Terminal - student: ~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal shows the following commands and output:

```
lab02_db=# SHOW temp_buffers;
temp_buffers
-----
8MB
(1 row)

lab02_db=# ALTER DATABASE lab02_db SET temp_buffers =
'32MB';
ALTER DATABASE
lab02_db=# \c lab02_db
You are now connected to database "lab02_db" as user
"postgres".
lab02_db=# SHOW temp_buffers;
temp_buffers
-----
32MB
(1 row)

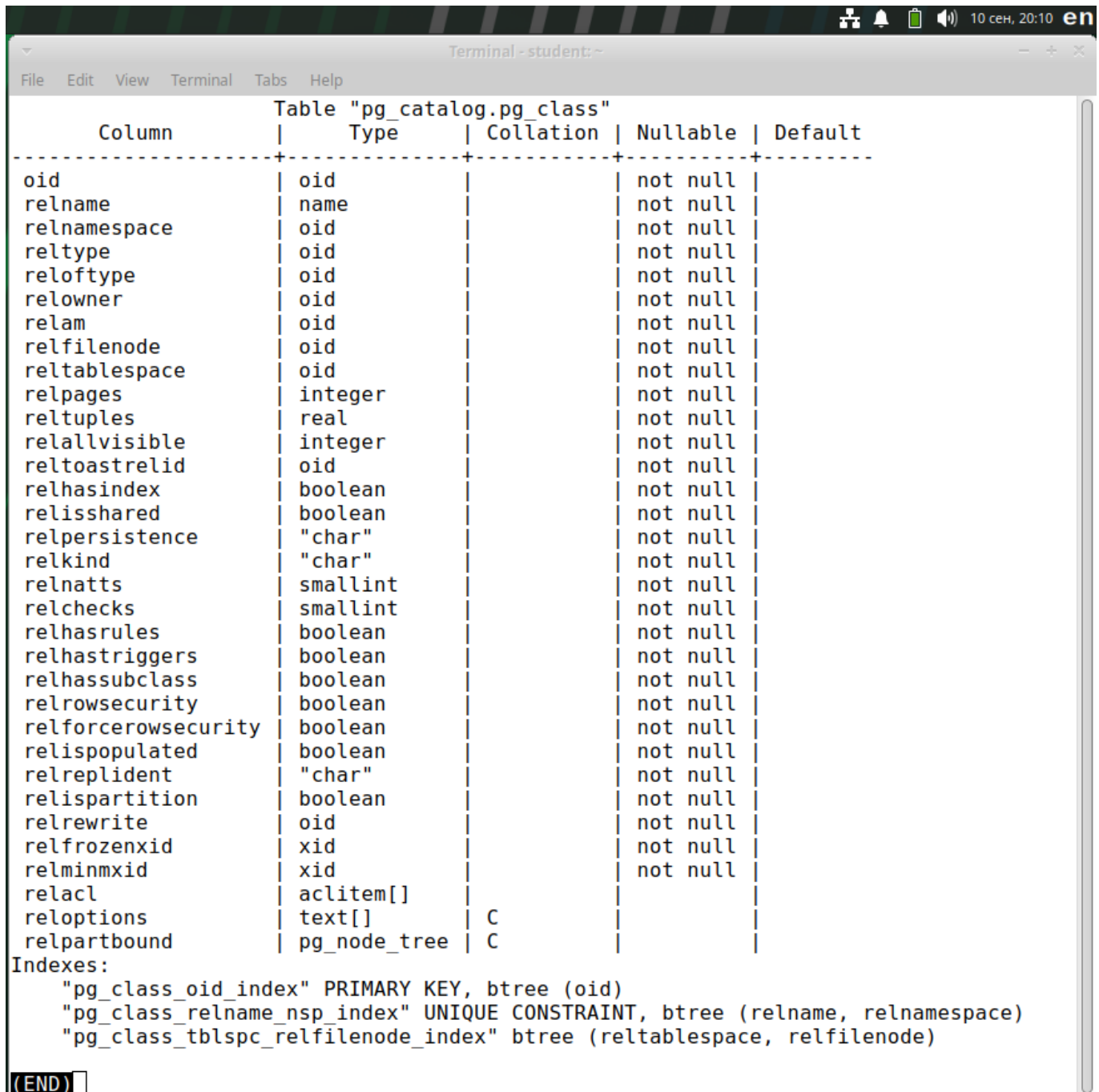
lab02_db=#
```

Рисунок 5 – Настройка параметра БД *temp_buffers*

Модуль 2. Системный каталог

1. Исследование `pg_class`

Получено описание системной таблицы `pg_class` с помощью команды `\d pg_class`. (Рис.6)



```

Terminal - student: ~
File Edit View Terminal Tabs Help

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 |
relpersistenc | "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)
  "pg_class_relname_nsp_index" UNIQUE CONSTRAINT, btree (relname, relnamespace)
  "pg_class_tblspc_relfilenode_index" btree (reltablespace, relfilenode)
(END)

```

Рисунок 6 – Исследование pg_class

2. Исследование pg_tables

Получено подробное описание представления pg_tables с помощью команды `\d+ pg_tables`. (Рис.7)

```

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   |
 tablespaces  | 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 tablespaces,
       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=#

```

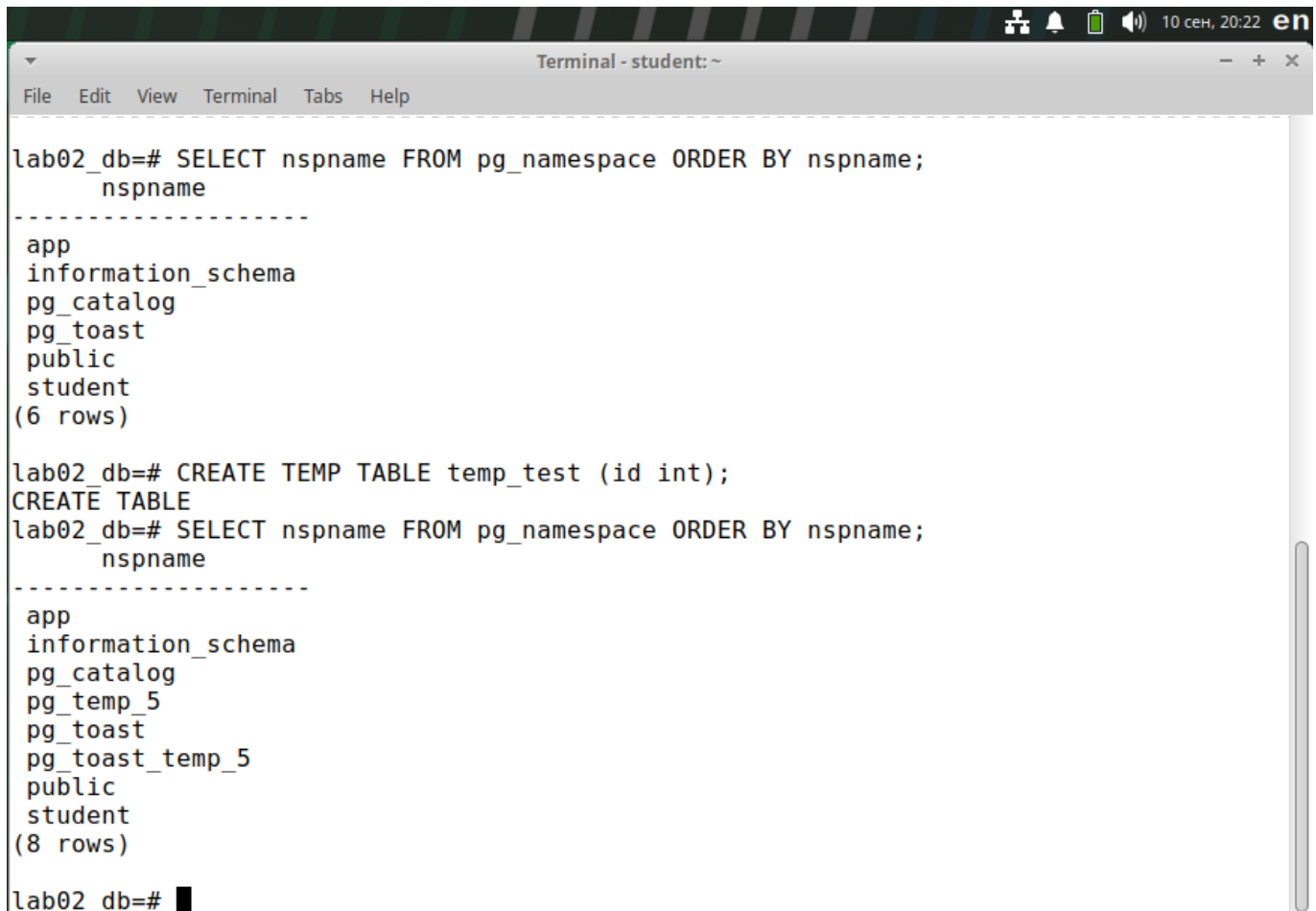
Рисунок 7 – Исследование pg_tables

Разница между таблицей и представлением:

Таблица — физический объект, хранящий данные на диске. Представление — виртуальный объект, определяемый запросом к одной или нескольким таблицам. `pg_tables` — это представление, которое показывает удобный вид данных из системных таблиц `pg_class` и `pg_namespace`.

3. Временная таблица и список схем

В базе `lab02_db` создана временная таблица. Получен полный список всех схем в БД, включая системные (`pg_catalog`, `information_schema`). (Рис.8)



```
lab02_db=# SELECT nspname FROM pg_namespace ORDER BY nspname;
nspname
-----
app
information_schema
pg_catalog
pg_toast
public
student
(6 rows)

lab02_db=# CREATE TEMP TABLE temp_test (id int);
CREATE TABLE
lab02_db=# SELECT nspname FROM pg_namespace ORDER BY nspname;
nspname
-----
app
information_schema
pg_catalog
pg_temp_5
pg_toast
pg_toast_temp_5
public
student
(8 rows)

lab02_db=#
```

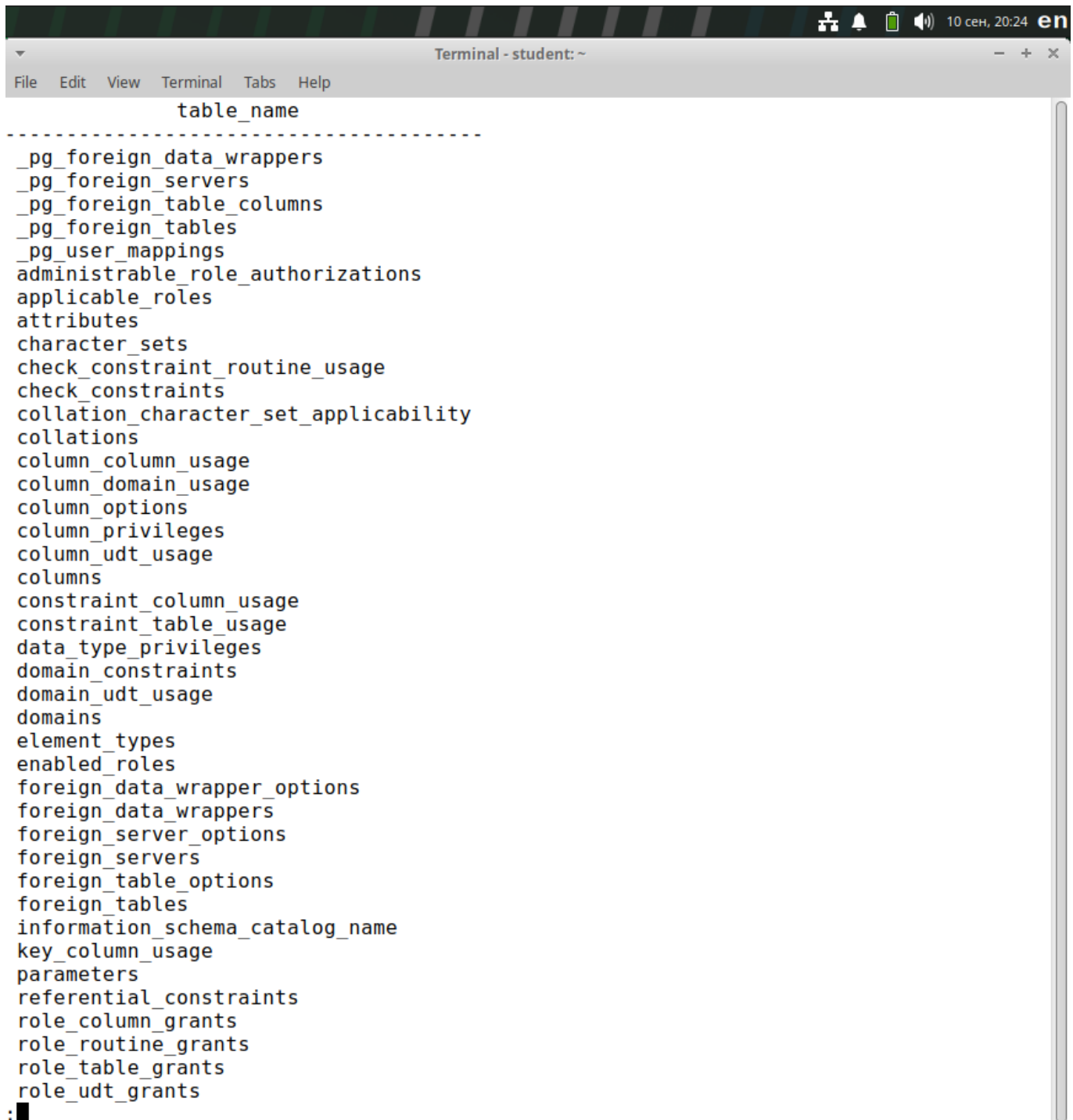
Рисунок 8 – Временная таблица и список схем

Временная схема создается автоматически для хранения временных объектов текущего сеанса. Она имеет уникальное имя вида `pg_temp_X` и удаляется при завершении сеанса.

4. Представления `information_schema`

Получен список всех представлений в схеме `information_schema`. (Рис.9)

```
SELECT table_name
FROM information_schema.views
WHERE table_schema = 'information_schema'
ORDER BY table_name;
```

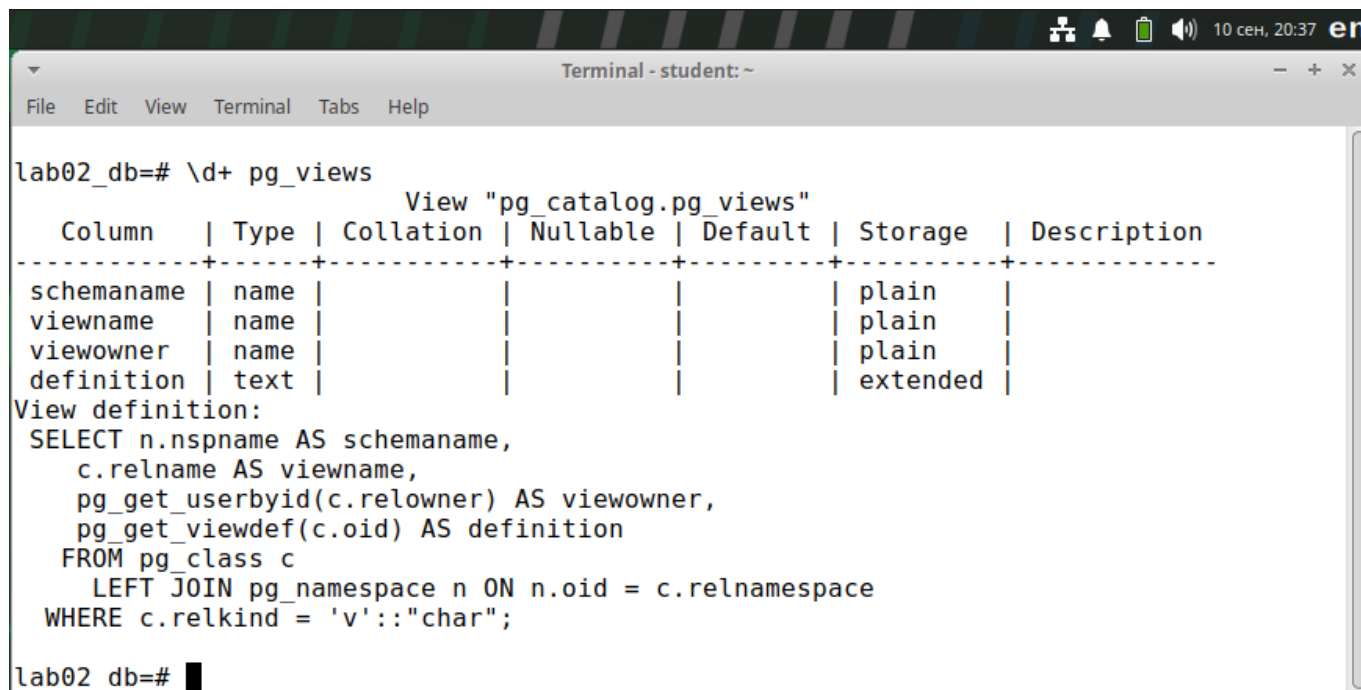
```
Terminal - student: ~
File Edit View Terminal Tabs Help

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
referential_constraints
role_column_grants
role_routine_grants
role_table_grants
role_udt_grants
:
```

Рисунок 9 – Представления *information_schema*

5. Анализ метакоманды

Выполнена команда `\d+ pg_views` в `psql`. Изучен вывод. (Рис.10)



```

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=#

```

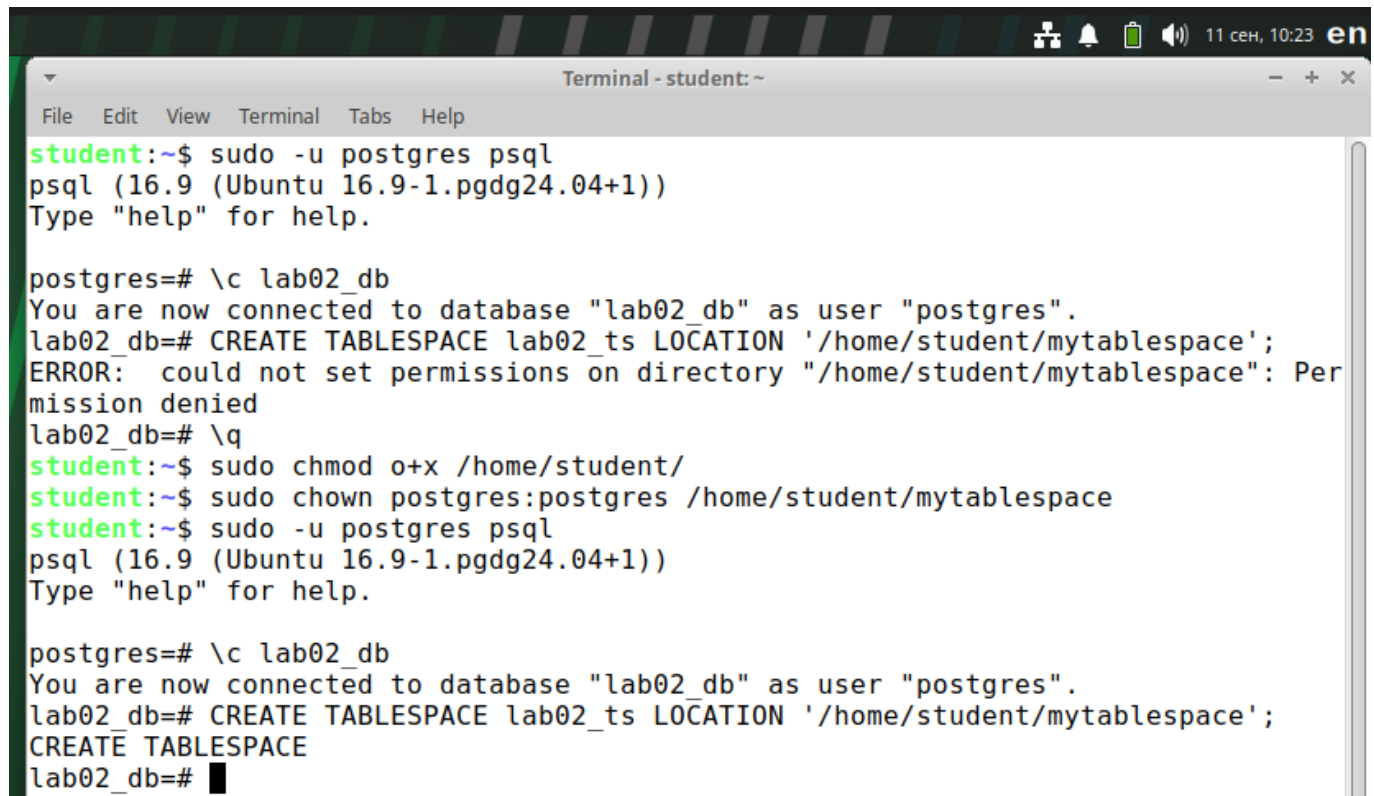
Рисунок 10 – Анализ метакоманды

Команда `\d+` выполняет запросы к системным таблицам `pg_class`, `pg_namespace`, `pg_description` для получения информации об объекте. Для представлений дополнительно запрашивается определение из `pg_rewrite`.

Модуль 3. Табличные пространства

1. Создание Tablespace

Создан каталог в файловой системе `/home/student/mytablespace`. Создано новое табличное пространство `lab02_ts`, указывающее на этот каталог. (Рис.11)



```
student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

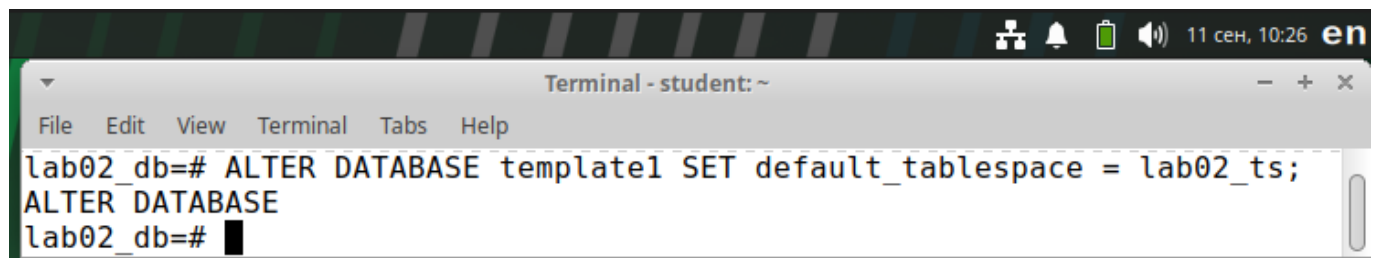
postgres=# \c lab02_db
You are now connected to database "lab02_db" as user "postgres".
lab02_db=# CREATE TABLESPACE lab02_ts LOCATION '/home/student/mytablespace';
ERROR: could not set permissions on directory "/home/student/mytablespace": Per
mission denied
lab02_db=# \q
student:~$ sudo chmod o+x /home/student/
student:~$ sudo chown postgres:postgres /home/student/mytablespace
student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

postgres=# \c lab02_db
You are now connected to database "lab02_db" as user "postgres".
lab02_db=# CREATE TABLESPACE lab02_ts LOCATION '/home/student/mytablespace';
CREATE TABLESPACE
lab02_db=#
```

Рисунок 11 – Создание Tablespace

2. Tablespace по умолчанию

Изменено табличное пространство по умолчанию для базы данных `template1` на `lab02_ts`. (Рис.12)



```
lab02_db=# ALTER DATABASE template1 SET default_tablespace = lab02_ts;
ALTER DATABASE
lab02_db=#
```

Рисунок 12 – Tablespace по умолчанию

`template1` используется как шаблон для создания новых БД. Изменив её табличное пространство по умолчанию, все новые БД будут наследовать это свойство.

3. Наследование свойства

Создана новая база данных `lab02_db_new`. Проверено её табличное пространство по умолчанию. (Рис.13)

```

lab02_db=# SELECT datname, dattablespace, spcname
FROM pg_database d
LEFT JOIN pg_tablespace t ON d.dattablespace = t.oid
WHERE datname = 'template1';
 datname | dattablespace | spcname 
-----+-----+-----
 template1 |          24583 | lab02_ts
(1 row)

lab02_db=# CREATE DATABASE lab02_db_new;
CREATE DATABASE
lab02_db=# SELECT datname, dattablespace, spcname FROM pg_database d
lab02_db=# LEFT JOIN pg_tablespace t ON d.dattablespace = t.oid
WHERE datname = 'lab02_db_new';
 datname | dattablespace | spcname 
-----+-----+-----
 lab02_db_new |          24583 | lab02_ts
(1 row)

lab02_db=#

```

Рисунок 13 – Наследование свойства

Новая БД унаследовала табличное пространство `lab02_ts` от `template1`, что подтверждает механизм наследования свойств от шаблонной БД.

4. Символическая ссылка

Найдена в каталоге `PGDATA/pg_tblspc/` символическая ссылка, соответствующая `lab02_ts`. (Рис.14)

```

student:~$ ls -la /var/lib/postgresql/16/main/pg_tblspc/
ls: cannot access '/var/lib/postgresql/16/main/pg_tblspc/': Permission denied
student:~$ sudo ls -la /var/lib/postgresql/16/main/pg_tblspc/
total 12
drwx----- 1 postgres postgres 4096 сен 11 10:23 .
drwx----- 1 postgres postgres 4096 сен 11 10:07 ..
lrwxrwxrwx 1 postgres postgres  26 сен 11 10:23 24583 -> /home/student/mytablespace
student:~$ sudo -u postgres ls -la /var/lib/postgresql/16/main/pg_tblspc/
total 12
drwx----- 1 postgres postgres 4096 сен 11 10:23 .
drwx----- 1 postgres postgres 4096 сен 11 10:07 ..
lrwxrwxrwx 1 postgres postgres  26 сен 11 10:23 24583 -> /home/student/mytablespace
student:~$

```

Рисунок 14 – Символическая ссылка

Ссылка указывает на реальный каталог файловой системы, где хранятся файлы данных.

5. Удаление Tablespace

Выполнена попытка удаления табличного пространства `lab02_ts`. (Рис.15)

```

Terminal - student: ~
File Edit View Terminal Tabs Help
lab02_db=# ALTER DATABASE template1 SET TABLESPACE pg_default;
ALTER DATABASE
lab02_db=# DROP DATABASE lab02_db_new;
DROP DATABASE
lab02_db=# DROP TABLESPACE lab02_ts;
DROP TABLESPACE
lab02_db=# \q
student:~$ sudo -u postgres rm -rf /var/lib/postgresql/ts_dir
student:~$

```

Рисунок 15 – Удаление Tablespace

В PostgreSQL оператор `DROP TABLESPACE` не поддерживает `CASCADE`. Опция `CASCADE` требуется, потому что табличное пространство используется БД `template1` и `lab02_db_new`. Без `CASCADE` PostgreSQL откажется удалять используемое табличное пространство.

6. Параметр Tablespace (Практика+)

Установлен параметр `random_page_cost` в значение 1.1 для табличного пространства `pg_default`. (Рис.16-17)

```

lab02_db=# ALTER TABLESPACE pg_default SET (random_page_cost = 1.1);
ALTER TABLESPACE
lab02_db=# \db+

[1]+  Stopped                  sudo -u postgres psql
student:~$

```

Рисунок 16 – Изменение параметра random_page_cost

```

Terminal - student: ~
File Edit View Terminal Tabs Help

List of tablespaces
-----
Name      | Owner   | Location | Access privileges | Options          | Size  | Description
-----
pg_default | postgres |          |                   | {random_page_cost=1.1} | 37 MB |
pg_global  | postgres |          |                   |                   | 605 kB |
(2 rows)

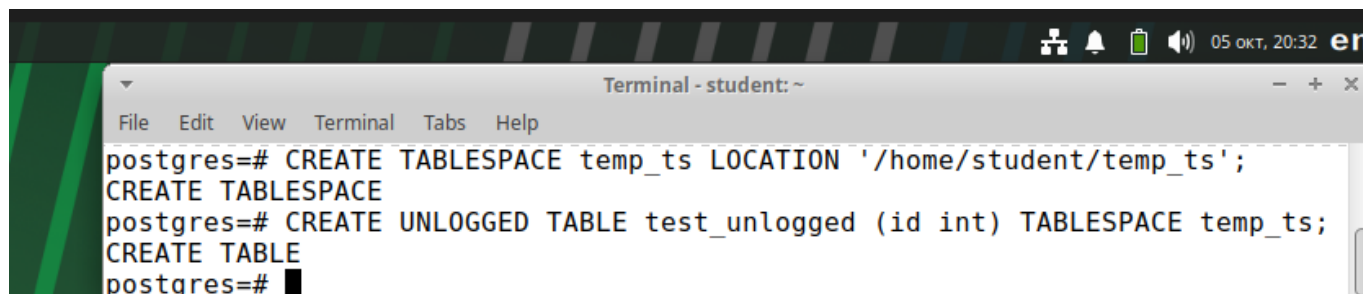
```

Рисунок 17 – Проверка изменений

Модуль 4. Низкий уровень

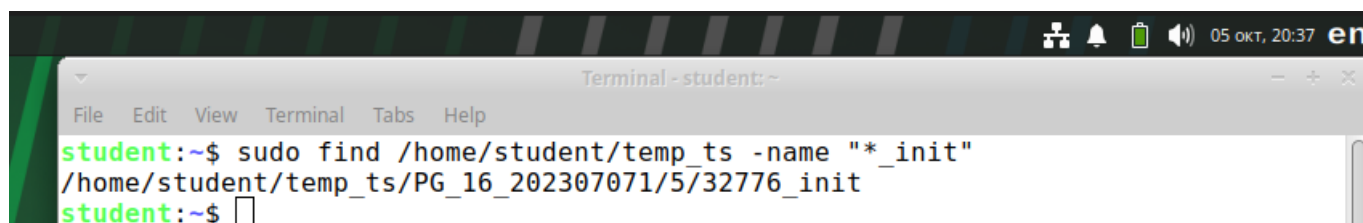
1. Нежурналируемая таблица

Создана нежурналируемая таблица в пользовательском табличном пространстве (создано временное для задания). Удалено табличное пространство. (Рис.18-20)



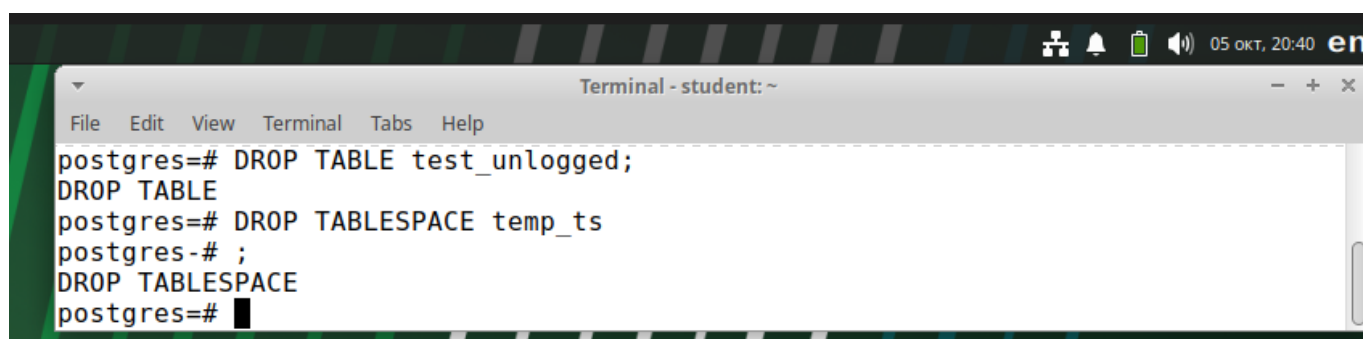
```
postgres=# CREATE TABLESPACE temp_ts LOCATION '/home/student/temp_ts';
CREATE TABLESPACE
postgres=# CREATE UNLOGGED TABLE test_unlogged (id int) TABLESPACE temp_ts;
CREATE TABLE
postgres=#
```

Рисунок 18 – Создание нежурналируемой таблицы



```
student:~$ sudo find /home/student/temp_ts -name "*_init"
/home/student/temp_ts/PG_16_202307071/5/32776_init
student:~$
```

Рисунок 19 – Поиск файла с суффиксом _init



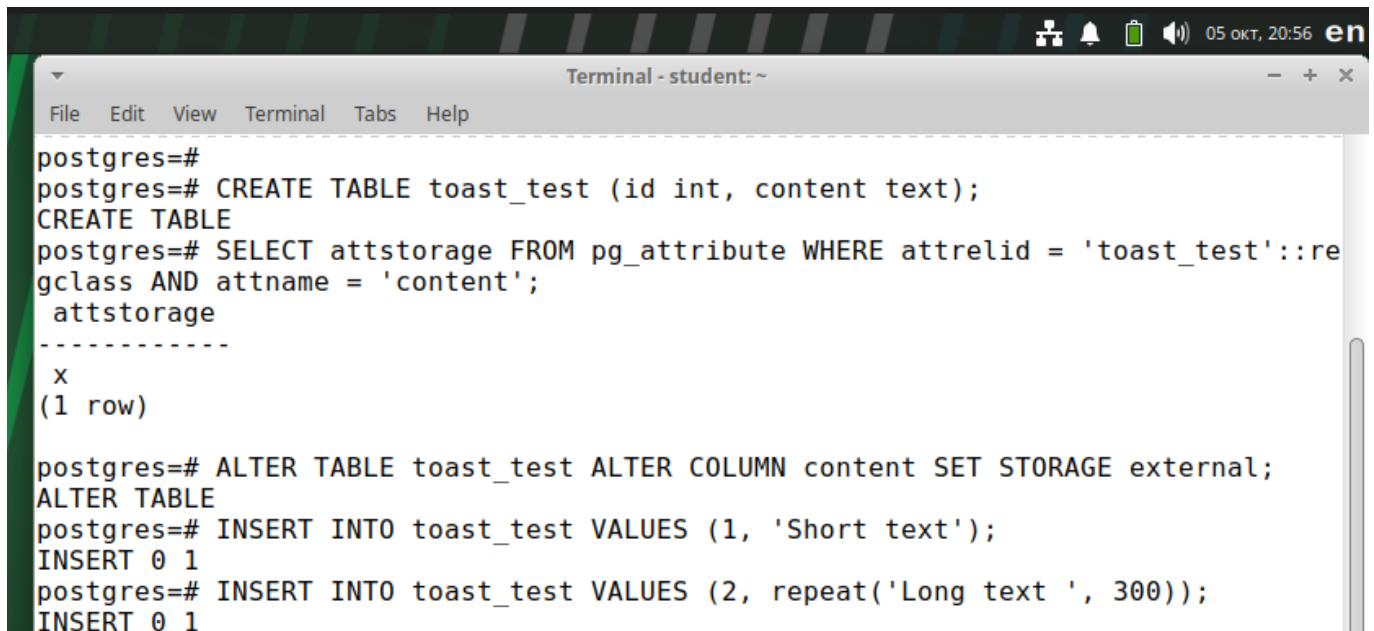
```
postgres=# DROP TABLE test_unlogged;
DROP TABLE
postgres=# DROP TABLESPACE temp_ts
postgres=# ;
DROP TABLESPACE
postgres=#
```

Рисунок 20 – Удаление созданного табличного пространства

Нежурналируемые таблицы не записываются в WAL, что повышает производительность, но делает их уязвимыми к сбоям. Файл `_init` используется для восстановления таблицы в пустое состояние после сбоя сервера.

2. Стратегии хранения TOAST

Создана таблица со столбцом типа `text`. Определена стратегия хранения по умолчанию. Изменена стратегия на `external`. Вставлены короткая (менее 2 КБ) и длинная (более 2 КБ) строки. (Рис.21-22)



```
postgres=#  
postgres=# CREATE TABLE toast_test (id int, content text);  
CREATE TABLE  
postgres=# SELECT attstorage FROM pg_attribute WHERE attrelid = 'toast_test'::regclass AND attname = 'content';  
attstorage  
-----  
x  
(1 row)  
  
postgres=# ALTER TABLE toast_test ALTER COLUMN content SET STORAGE external;  
ALTER TABLE  
postgres=# INSERT INTO toast_test VALUES (1, 'Short text');  
INSERT 0 1  
postgres=# INSERT INTO toast_test VALUES (2, repeat('Long text ', 300));  
INSERT 0 1
```

Рисунок 21 – Создание таблицы, определение стратегии и ввод данных


```

Terminal - student: ~
File Edit View Terminal Tabs Help

postgres=# SELECT relname, reltoastrelid FROM pg_class WHERE relname = 'toast_test';
   relname   | reltoastrelid 
-----+-----
 toast_test |          32782
(1 row)

postgres=# SELECT chunk_id, chunk_seq, length(chunk_data) as chunk_size FROM pg_toast.pg_toast_32782;
ERROR:  relation "pg_toast.pg_toast_32782" does not exist
LINE 1: ... chunk_seq, length(chunk_data) as chunk_size FROM pg_toast.p...
^

postgres=# SELECT n.nspname, c.relname FROM pg_class c JOIN pg_namespace n ON c.relnamespace = n.oid WHERE c.oid = 32782;
   nspname   |   relname   
-----+-----
 pg_toast    | pg_toast_32779
(1 row)

postgres=# SELECT chunk_id, chunk_seq, length(chunk_data) as chunk_size FROM pg_toast.pg_toast_32779;
   chunk_id | chunk_seq | chunk_size 
-----+-----+-----
    32784  |         0 |        1996
    32784  |         1 |        1004
(2 rows)

postgres=# INSERT INTO toast_test VALUES (2, repeat('Long text ', 700));
INSERT 0 1
postgres=# SELECT chunk_id, chunk_seq, length(chunk_data) as chunk_size FROM pg_toast.pg_toast_32779;
   chunk_id | chunk_seq | chunk_size 
-----+-----+-----
    32784  |         0 |        1996
    32784  |         1 |        1004
    32785  |         0 |        1996
    32785  |         1 |        1996
    32785  |         2 |        1996
    32785  |         3 |        1012
(6 rows)

postgres=#

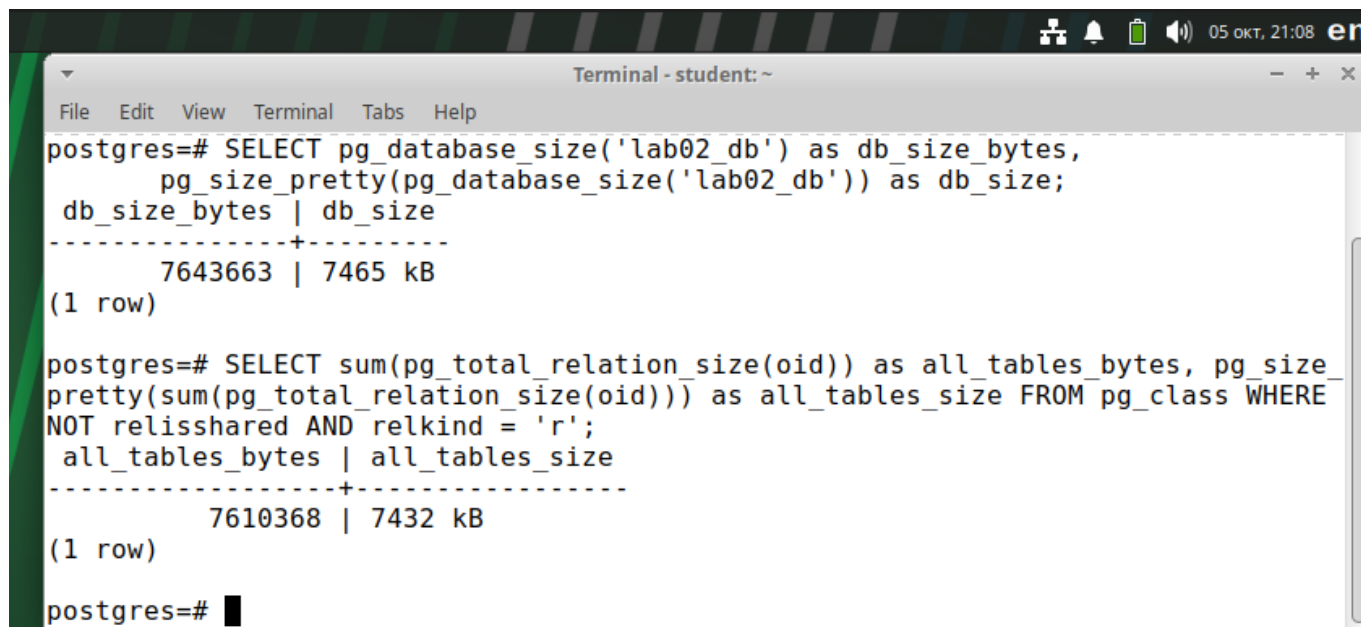
```

Рисунок 22 – Проверка TOAST-таблицы

Короткая строка остается в основной таблице. Длинная строка (>2KB) помещается в TOAST-таблицу и разбивается на чанки. Стратегия external означает, что большие значения всегда выносятся в TOAST без сжатия.

3. Анализ размера БД (Практика+)

Сравнен размер базы `lab02_db`, возвращаемый `pg_database_size`, с суммой размеров всех пользовательских таблиц. (Рис.23)



```
Terminal - student: ~
File Edit View Terminal Tabs Help

postgres=# SELECT pg_database_size('lab02_db') as db_size_bytes,
        pg_size_pretty(pg_database_size('lab02_db')) as db_size;
 db_size_bytes | db_size 
-----+-----
       7643663 | 7465 kB
(1 row)

postgres=# SELECT sum(pg_total_relation_size(oid)) as all_tables_bytes, pg_size_pretty(sum(pg_total_relation_size(oid))) as all_tables_size FROM pg_class WHERE NOT relisshared AND relkind = 'r';
 all_tables_bytes | all_tables_size 
-----+-----
       7610368 | 7432 kB
(1 row)

postgres=#
```

Рисунок 23 – Сравнение размеров базы данных и таблиц

Объяснение расхождения: Размер базы данных включает не только пользовательские таблицы, но и системный каталог, индексы системных таблиц, файлы visibility map, free space map, временные файлы и служебные структуры. Поэтому размер БД всегда больше суммы размеров пользовательских таблиц.

4. Методы сжатия TOAST (Практика+)

Проверено средствами SQL, был ли PostgreSQL скомпилирован с поддержкой методов сжатия `pglz` и `lz4`. (Рис.24)

```
SELECT name, setting FROM pg_settings WHERE name IN ('default_toast_compression');
SHOW default_toast_compression;
```

```

student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

postgres=# SELECT name, setting FROM pg_settings WHERE name IN ('default_toast_compression');
      name      | setting
-----+-----
default_toast_compression | pglz
(1 row)

postgres=# SHOW default_toast_compression;
default_toast_compression
-----
pglz
(1 row)

postgres=# 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)

```

Рисунок 24 – Доступные методы сжатия TOAST

PostgreSQL поддерживает метод сжатия **pglz** по умолчанию. Поддержка **lz4** зависит от параметров компиляции (флаг **--with-lz4**).

5. Сравнение сжатия (Практика+)

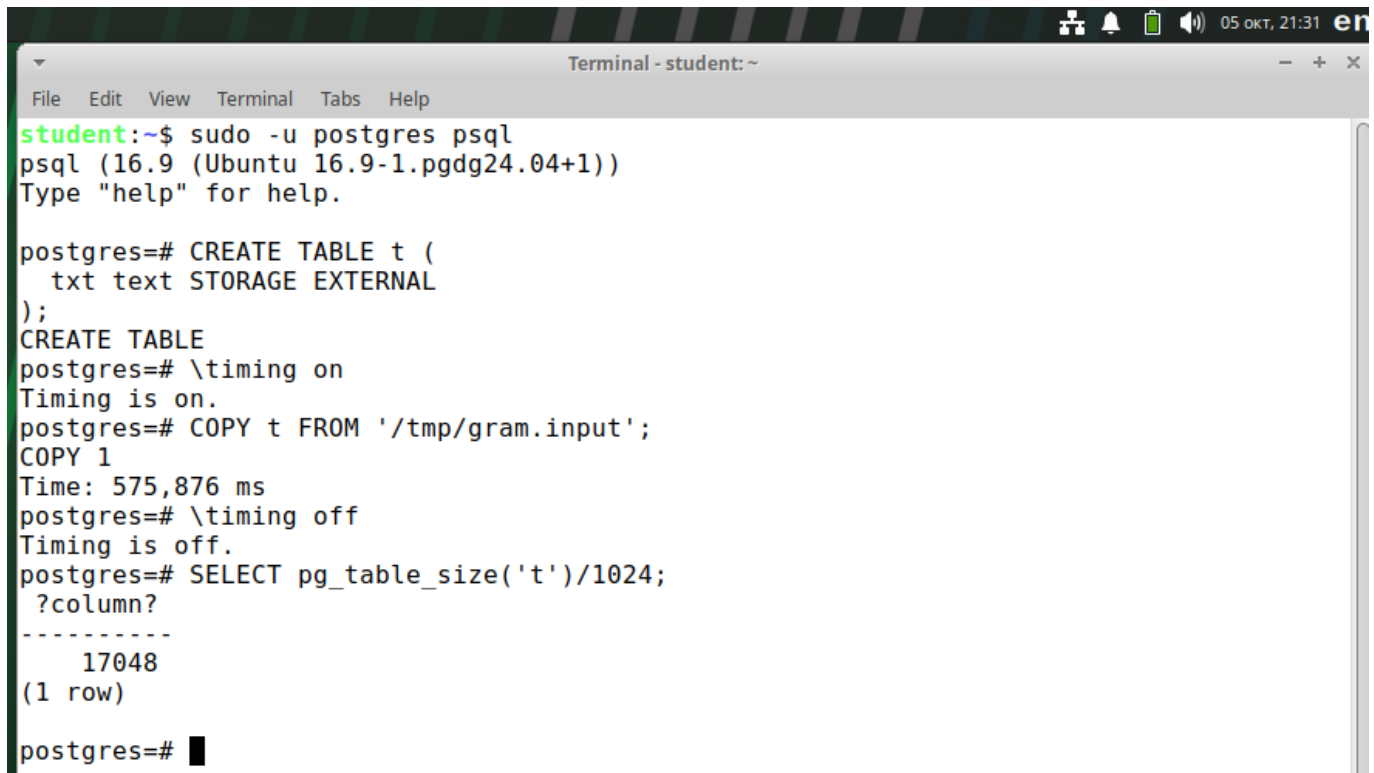
Создан текстовый файл размером более 10 МБ. Содержимое файла загружено в три таблицы с различными стратегиями хранения: без сжатия (**external**), со сжатием **pglz** и **lz4**. Сравнены размеры таблиц и время загрузки. (Рис.25-28)

```

student:~$ sudo cat /usr/lib/postgresql/16/bin/postgres | base32 -w0 > /tmp/gram.input
student:~$ ls -l --block-size=K /tmp/gram.input
-rw-rw-r-- 1 student student 16379K окт  5 21:29 /tmp/gram.input
student:~$

```

Рисунок 25 – Создание текстового файла

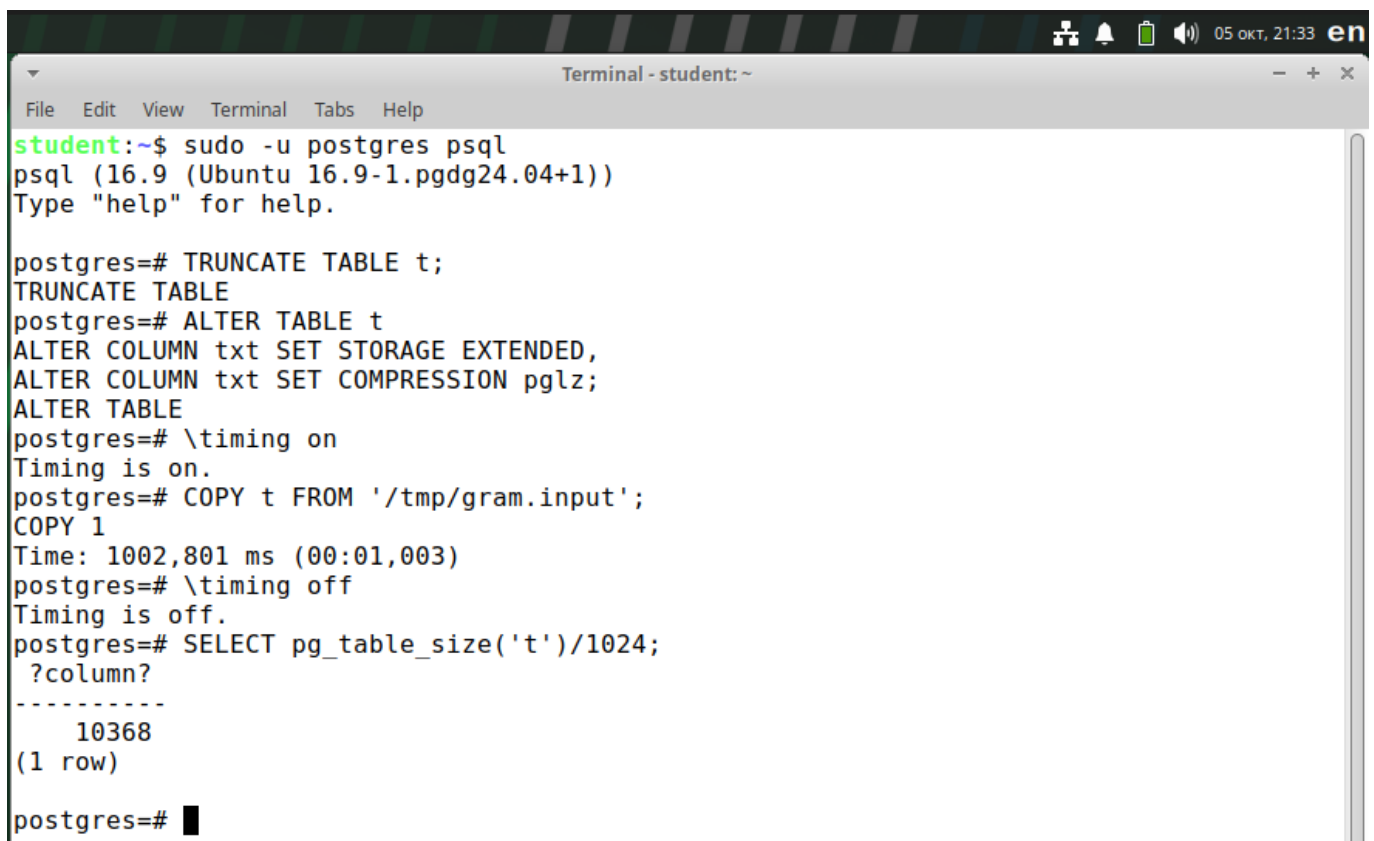


```
student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

postgres=# CREATE TABLE t (
      txt text STORAGE EXTERNAL
);
CREATE TABLE
postgres=# \timing on
Timing is on.
postgres=# COPY t FROM '/tmp/gram.input';
COPY 1
Time: 575,876 ms
postgres=# \timing off
Timing is off.
postgres=# SELECT pg_table_size('t')/1024;
?column?
-----
    17048
(1 row)

postgres=#
```

Рисунок 26 – Загрузка без сжатия

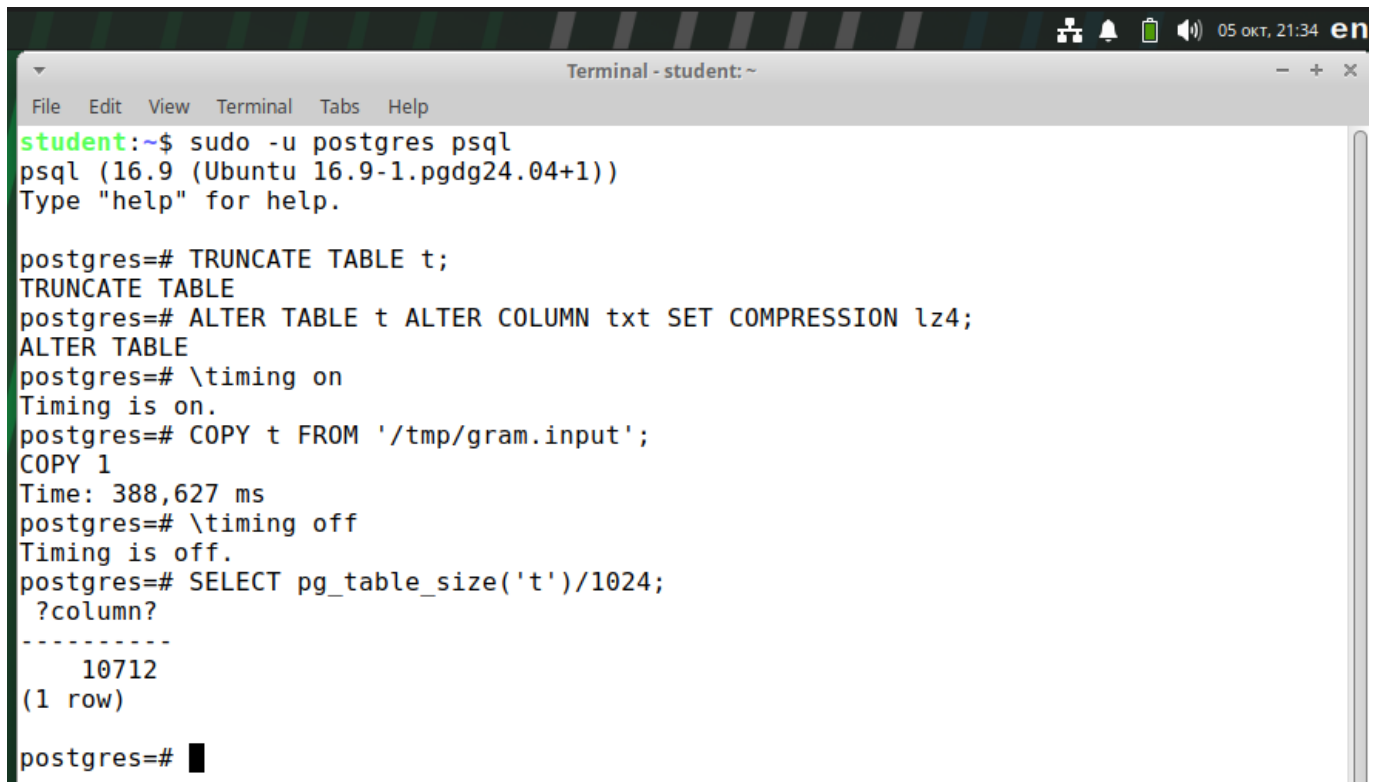


```
student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

postgres=# TRUNCATE TABLE t;
TRUNCATE TABLE
postgres=# ALTER TABLE t
ALTER COLUMN txt SET STORAGE EXTENDED,
ALTER COLUMN txt SET COMPRESSION pglz;
ALTER TABLE
postgres=# \timing on
Timing is on.
postgres=# COPY t FROM '/tmp/gram.input';
COPY 1
Time: 1002,801 ms (00:01,003)
postgres=# \timing off
Timing is off.
postgres=# SELECT pg_table_size('t')/1024;
?column?
-----
    10368
(1 row)

postgres=#
```

Рисунок 27 – Сжатие с помощью pglz



```
student:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1))
Type "help" for help.

postgres=# TRUNCATE TABLE t;
TRUNCATE TABLE
postgres=# ALTER TABLE t ALTER COLUMN txt SET COMPRESSION lz4;
ALTER TABLE
postgres=# \timing on
Timing is on.
postgres=# COPY t FROM '/tmp/gram.input';
COPY 1
Time: 388,627 ms
postgres=# \timing off
Timing is off.
postgres=# SELECT pg_table_size('t')/1024;
?column?
-----
 10712
(1 row)

postgres=#
```

Рисунок 28 – Сжатие с помощью lz4

Результаты:

- **External (без сжатия):** максимальный размер, минимальное время загрузки
- **PGLZ:** средняя степень сжатия, среднее время загрузки
- **LZ4:** лучшее соотношение скорости и степени сжатия (если поддерживается)

Выводы

В ходе выполнения лабораторной работы были изучены логическая и физическая структуры хранения данных в PostgreSQL. Получены практические навыки управления базами данных, схемами, табличными пространствами. Освоены работы с системным каталогом для извлечения метаинформации. Исследованы низкоуровневые аспекты хранения, включая TOAST.