

Лабораторная работа №02: Организация данных и системный каталог

Выполнил: студент 4 курса группы ПИЖ-б-о-22-1 Плугатырев Владислав Алексеевич

Тема: 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 метаданных. Исследование низкоуровневых аспектов хранения, включая TOAST.

Порядок выполнения работы

Модуль 1. Базы данных и схемы

1. Создание и проверка БД.

```
postgres@localhost:postgres> CREATE DATABASE lab02_db;
CREATE DATABASE
Time: 0.201s
postgres@localhost:postgres> \c lab02_db;
You are now connected to database "lab02_db" as user "postgres"
Time: 0.017s
postgres@localhost:lab02_db> SELECT pg_size_pretty(pg_database_size("lab02_db"));
столбец "lab02_db" не существует
LINE 1: SELECT pg_size_pretty(pg_database_size("lab02_db"))
                                             ^
Time: 0.013s
postgres@localhost:lab02_db> SELECT pg_size_pretty(pg_database_size('lab02_db'));
+-----+
| pg_size_pretty |
+-----+
| 7660 kB        |
+-----+
SELECT 1
Time: 0.021s
```

Рисунок 1.1 - Создание БД и проверка ее размера

2. Работа со схемами.

```

[postgres@localhost:lab02_db> CREATE schema app;
CREATE SCHEMA
Time: 0.015s
[postgres@localhost:lab02_db> CREATE schema lofanb14;
CREATE SCHEMA
Time: 0.001s

```

Рисунок 1.2 - Создание схем

```

[postgres@localhost:lab02_db> CREATE TABLE lofanb14.students (id SERIAL PRIMARY KEY, name VARCHAR(50), surname VARCHAR(50));
CREATE TABLE
Time: 0.027s
[postgres@localhost:lab02_db> CREATE TABLE app.app_table (id SERIAL PRIMARY KEY, username VARCHAR(30), description TEXT);
CREATE TABLE
Time: 0.012s

```

Рисунок 1.3 - Создание таблиц в схемах

```

postgres@localhost:lab02_db> INSERT INTO lofanb14.students (name, surname) VALUES ('Владислав', 'Плутатырев');
INSERT 0 1
Time: 0.007s
postgres@localhost:lab02_db> INSERT INTO app.app_table (username, description) VALUES ('BigLofanb14', 'Самый крутой юзер');
INSERT 0 1
Time: 0.003s

```

Рисунок 1.4 - Вставка данных в таблицу

3. Контроль размера.

```

[postgres@localhost:lab02_db> SELECT pg_size_pretty(pg_database_size('lab02_db'));
+-----+
| pg_size_pretty |
+-----+
| 7788 kB        |
+-----+
SELECT 1
Time: 0.016s

```

Рисунок 1.5 - Проверка размера БД

Объяснение: Размер БД увеличится, так как мы добавили новые объекты (схемы, таблицы, последовательности для SERIAL) и данные. Системные таблицы (pg_class , pg_attribute , где pg_class содержит информацию о таблицах и подобных им объектов, а pg_attribute содержит каталог столбцов) также пополнились записями о новых объектах.

4. Управление путем поиска.

```

[postgres@localhost:lab02_db> SHOW search_path;
+-----+
| search_path |
+-----+
| "$user", public |
+-----+
SHOW 1
Time: 0.009s
[postgres@localhost:lab02_db> SET search_path TO lofanb14, app, public;
SET
Time: 0.001s
[postgres@localhost:lab02_db> SHOW search_path;
+-----+
| search_path |
+-----+
| lofanb14, app, public |
+-----+
SHOW 1
Time: 0.008s

```

Рисунок 1.6 - Изменение параметра search_path

```

[postgres@localhost:lab02_db> SELECT * FROM students;
+----+-----+-----+
| id | name      | surname |
+----+-----+-----+
| 1  | Владислав | Плугатырев |
+----+-----+-----+
SELECT 1
Time: 0.010s
[postgres@localhost:lab02_db> SELECT * FROM app_table;
+----+-----+-----+
| id | username   | description |
+----+-----+-----+
| 1  | BigLofanb14 | Самый крутой юзер |
+----+-----+-----+
SELECT 1
Time: 0.010s
postgres@localhost:lab02_db> █

```

Рисунок 1.7 - Демонстрация работы

5. Практика (настройка параметра БД)

```

postgres@localhost:lab02_db> SHOW temp_buffers;
+-----+
| temp_buffers |
+-----+
| 8MB          |
+-----+
SHOW 1
Time: 0.010s

```

Рисунок 1.8 - Текущее значение параметра temp_buffers

```

postgres@localhost:lab02_db> \c postgres;
You are now connected to database "postgres" as user "postgres"
Time: 0.011s
postgres@localhost:postgres> ALTER DATABASE lab02_db SET temp_buffers = '32MB';
You're about to run a destructive command.
Do you want to proceed? [y/N]: y
Your call!
ALTER DATABASE
Time: 0.007s
postgres@localhost:postgres> \c lab02_db;
You are now connected to database "lab02_db" as user "postgres"
Time: 0.008s
postgres@localhost:lab02_db> SHOW temp_buffers;
+-----+
| temp_buffers |
+-----+
| 32MB         |
+-----+
SHOW 1
Time: 0.007s

```

Рисунок 1.9 - Подключение к БД postgres, изменение параметра temp_buffers для БД lab02_db, проверка изменений

Модуль 2. Системный каталог

1. Исследование pg_class

```
postgres@localhost:lab02_db> \d pg_class;
```

Column	Type	Modifiers
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[]	collate C
relpartbound	pg_node_tree	collate 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)

Time: 0.037s

Рисунок 2.1 - Структура таблицы pg_class

Вывод: Покажет структуру таблицы pg_class : столбцы

(oid, relname, relnamespace, relkind и т.д.) и их типы. Эта таблица хранит информацию обо всех отношениях (таблицах, индексах, последовательностях, представлениях).

2. Исследование pg_tables .

```
[postgres@localhost:lab02_db> \d+ pg_tables;
```

Column	Type	Modifiers	Storage	Description
schemaname	name		plain	<null>
tablename	name		plain	<null>
tableowner	name		plain	<null>
tablespace	name		plain	<null>
hasindexes	boolean		plain	<null>
hasrules	boolean		plain	<null>
hastriggers	boolean		plain	<null>
rowsecurity	boolean		plain	<null>

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"]);
```

Time: 0.025s

```
postgres@localhost:lab02_db> █
```

Рисунок 2.2 - Определения представления pg_tables

Вывод: Покажет определение представления pg_tables (столбцы и SQL-запрос, который его формирует).

Объяснение разницы: pg_class - это системная **таблица**, хранящая сырые данные. pg_tables - это **представление** (view), которое является сохранённым SQL-запросом, выбирающим и формирующим данные из одной или нескольких системных таблиц (включая pg_class) для удобного просмотра информации именно о таблицах.

3. Временная таблица и список схем.

```
[postgres@localhost:lab02_db> CREATE TEMP TABLE temp_example(id INT);  
CREATE TABLE  
Time: 0.013s
```

Рисунок 2.3 - Создание временной таблицы

```
postgres@localhost:lab02_db> SELECT nspname FROM pg_catalog.pg_namespace ORDER BY nspname;
```

nspname
app
information_schema
lofanbl4
pg_catalog
pg_temp_4
pg_toast
pg_toast_temp_4
public

```
SELECT 8
Time: 0.010s
```

Рисунок 2.4 - Список всех схем (пространств имен)

Вывод: Среди прочих видны

схемы `pg_catalog`, `information_schema`, `public`, `app`, `lofanbl4`, а также схему с именем типа `pg_temp_4`.

Объяснение: Временные таблицы создаются в специальной временной схеме, которая уникальна для каждого сеанса. Это нужно для изоляции временных данных разных сеансов.

4. Представления `information_schema`.

```
postgres@localhost:lab02_db> SELECT table_name FROM information_schema.views WHERE table_schema = 'information_schema' ORDER BY table_name;
```

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

Рисунок 2.5 - Список представлений в схеме `information_schema`

5. Анализ метакоманды `\d+ pg_views`.

```
[postgres@localhost:lab02_db> \d+ pg_views;
```

Column	Type	Modifiers	Storage	Description
schemaname	name		plain	<null>
viewname	name		plain	<null>
viewowner	name		plain	<null>
definition	text		extended	<null>

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";
```

Time: 0.017s

Рисунок 2.6 - Выполнение метакоманды

Объяснение: Команда `\d+` в `psql` является метакомандой. Она формирует и выполняет запросы к системному каталогу, чтобы представить информацию в удобочитаемом виде. За этой командой скрываются запросы к таким таблицам, как `pg_class` (для получения общего описания), `pg_attribute` (для получения информации о столбцах) и другим, в зависимости от объекта.

Модуль 3. Табличные пространства

1. Создание Tablespace .

Табличное пространство (Tablespace) в PostgreSQL — это абстракция, которая позволяет определять, **где на диске** будут храниться файлы данных базы данных. Это механизм для управления физическим размещением данных.

```
[admin@MacBook-Air-Admin ~ % mkdir -p /tmp/mytablespace
admin@MacBook-Air-Admin ~ %
```

Рисунок 2.7 - Создание каталога

```
[postgres@localhost:postgres> CREATE TABLESPACE lab02_ts LOCATION '/tmp/mytablespace';
CREATE TABLESPACE
Time: 0.010s
```

Рисунок 2.8 - Создание табличного пространства

2. Tablespace по умолчанию для template1.

```
[postgres@localhost:postgres> CREATE TABLESPACE lab02_ts LOCATION '/tmp/mytablespace';
CREATE TABLESPACE
Time: 0.010s
[postgres@localhost:postgres> ALTER DATABASE template1 SET TABLESPACE lab02_ts;
You're about to run a destructive command.
Do you want to proceed? [y/N]: y
Your call!
ALTER DATABASE
Time: 0.163s
```

Рисунок 2.9 - Создание табличного пространства и изменение табличного пространства по умолчанию для БД template1

Цель действия: База данных `template1` используется как шаблон по умолчанию при создании новых баз данных. Установив для неё tablespace по умолчанию, мы гарантируем, что все новые БД, созданные без явного указания tablespace, будут использовать `lab02_ts`.

3. Наследование свойства.

```
[postgres@localhost:postgres> CREATE DATABASE lab02_db_new;
CREATE DATABASE
Time: 0.194s
```

Рисунок 3.1 - Создание новой БД

```
[postgres@localhost:postgres> SELECT d.datname, t.spcname
FROM pg_database d
LEFT JOIN pg_tablespace t ON d.dattablespace = t.oid
WHERE d.datname = 'lab02_db_new';
```

datname	spcname
lab02_db_new	lab02_ts

```
SELECT 1
Time: 0.017s
```

Рисунок 3.2 - Проверка табличного пространства для созданной БД

Объяснение результата: Новая БД `lab02_db_new` унаследует табличное пространство `lab02_ts`, потому что она была создана на основе `template1`, для которого установлено `lab02_ts` как пространство по умолчанию.

4. Символическая ссылка.

Символическая ссылка (symbolic link или symlink) — это специальный файл в файловой системе, который **содержит путь к другому файлу или директории**. Это своего рода "ярлык" или "указатель" на оригинальный файл.

```

[admin@MacBook-Air-Admin ~ % ls -la /opt/homebrew/var/postgresql@16/pg_tblspc/
total 0
drwx-----@ 3 admin admin 96 25 сен 17:28 .
drwx-----@ 27 admin admin 864 25 сен 16:38 ..
lrwx-----@ 1 admin admin 17 25 сен 17:28 16412 -> /tmp/mytablespace

```

Рисунок 3.3 - Определение OID tablespace

```

[postgres@localhost:postgres> SELECT oid FROM pg_tablespace WHERE spcname = 'lab02_ts';
+-----+
|  oid  |
+-----+
| 16412 |
+-----+
SELECT 1
Time: 0.015s

```

Рисунок 3.4 - Определение OID через psql

```

[admin@MacBook-Air-Admin ~ % ls -la /opt/homebrew/var/postgresql@16/pg_tblspc/16412
lrwx-----@ 1 admin admin 17 25 сен 17:28 /opt/homebrew/var/postgresql@16/pg_tblspc/16412 -> /tmp/mytablespace

```

Рисунок 3.5 - Проверка куда ведет символическая ссылка

Вывод: Символическая ссылка будет вести на каталог `/tmp/mytablespace`, который мы указали при создании tablespace.

5. Удаление Tablespace

```

[postgres@localhost:postgres> DROP DATABASE lab02_db_new;
You're about to run a destructive command.
Do you want to proceed? [y/N]: y
Your call!
DROP DATABASE
Time: 0.056s
[postgres@localhost:postgres> DROP TABLESPACE lab02_ts;
You're about to run a destructive command.
Do you want to proceed? [y/N]: y
Your call!
DROP TABLESPACE
Time: 0.004s

```

Рисунок 3.6 - Удаление БД и табличного пространства

6. Практика+ (Параметр Tablespace).

```

postgres@localhost:postgres> ALTER TABLESPACE pg_default SET (random_page_cost = 1.1);
You're about to run a destructive command.
Do you want to proceed? [y/N]: n
Error: invalid input
You're about to run a destructive command.
Do you want to proceed? [y/N]: y
Your call!
ALTER TABLESPACE
Time: 0.004s
postgres@localhost:postgres> SELECT spcname, spcoptions FROM pg_tablespace WHERE spcname = 'pg_default';
+-----+-----+
| spcname | spcoptions |
+-----+-----+
| pg_default | ['random_page_cost=1.1'] |
+-----+-----+
SELECT 1
Time: 0.006s

```

Рисунок 3.7 - Установка параметра `random_page_cost` и проверка

Модуль 4. Низкий уровень

1. Нежурналируемая таблица.

Нежурналируемая таблица (англ. *unlogged table*) — это специальный тип таблицы в PostgreSQL, данные которой не записываются в журнал предзаписи (Write-Ahead Log, WAL). Это даёт существенный прирост производительности при записи, но в случае сбоя (аварийного завершения, отключения питания и т.п.) все данные в таких таблицах будут утеряны.

```

admin@MacBook-Air-Admin ~ % mkdir -p /tmp/ts_unlogged
admin@MacBook-Air-Admin ~ % pgcli -h localhost -u postgres -d postgres
Using local time zone Europe/Moscow (server uses Europe/Moscow)
Use 'set time zone <TZ>' to override, or set 'use_local_timezone = False' in the config
Server: PostgreSQL 16.10 (Homebrew)
Version: 4.3.0
Home: http://pgcli.com
postgres@localhost:postgres> CREATE TABLESPACE ts_unlogged LOCATION '/tmp/ts_unlogged';
CREATE TABLESPACE
Time: 0.006s
postgres@localhost:postgres> CREATE UNLOGGED TABLE unlogged_table (id SERIAL, data TEXT) TABLESPACE ts_unlogged;
CREATE TABLE
Time: 0.027s
postgres@localhost:postgres> \! find /tmp/ts_unlogged -name "*_init"
/tmp/ts_unlogged/PG_16_202307071/5/16422_init
/tmp/ts_unlogged/PG_16_202307071/5/16421_init
/tmp/ts_unlogged/PG_16_202307071/5/16417_init
Time: 0.018s

```

Рисунок 3.8 - Создание временного табличного пространства, нежурналируемой таблицы и проверка файла с суффиксом `_init`.

Слой `_init` используется для быстрой инициализации нежурналируемых таблиц.

2. Стратегии хранения TOAST.

TOAST (The Oversized-Attribute Storage Technique) — это встроенная в PostgreSQL технология для эффективного хранения больших значений полей, которые не помещаются в стандартную страницу данных (обычно 8 КБ).

```
postgres@localhost:postgres> CREATE TABLE toast_test (id SERIAL, large_text TEXT);
CREATE TABLE
Time: 0.011s
postgres@localhost:postgres> SELECT attname, attstorage FROM pg_attribute WHERE attrelid = 'toast_test'::regclass AND attname = 'large_text';
+-----+-----+
| attname | attstorage |
+-----+-----+
| large_text | x          |
+-----+-----+
SELECT 1
Time: 0.016s
postgres@localhost:postgres> ALTER TABLE toast_test ALTER COLUMN large_text SET STORAGE EXTERNAL;
You're about to run a destructive command.
Do you want to proceed? [y/N]: y
Your call!
ALTER TABLE
Time: 0.004s
```

[illegible]

Объяснение результата: для короткой строки запись в TOAST-таблице будет пустой или отсутствовать, так как она помещается в основную таблицу (встроенное хранение). Для длинной строки в TOAST-таблице появится запись, так как её размер превышает порог (обычно 2 КБ) и требует out-of-line хранения.

```

postgres@localhost:postgres> SELECT pg_size_pretty(pg_database_size('lab02_db')) as db_size;
+-----+
| db_size |
+-----+
| 7788 kB |
+-----+
SELECT 1
Time: 0.016s
postgres@localhost:postgres> SELECT pg_size_pretty(SUM(pg_total_relation_size(oid))) as total_tables_size
FROM pg_class
WHERE relkind = 'r' -- Обычные таблицы
AND relnamespace NOT IN (SELECT oid FROM pg_namespace WHERE nspname IN ('pg_catalog', 'information_schema'));
+-----+
| total_tables_size |
+-----+
| 64 kB             |
+-----+
SELECT 1
Time: 0.018s

```

Рисунок 4.2 - Анализ размеров

Объяснение расхождения: Размер БД всегда больше суммы размеров таблиц. В размер БД входят:

- Размер системных таблиц (pg_class , pg_attribute и др.)
- Размер индексов
- Размер TOAST-таблиц
- Размер свободного пространства (зарезервированного под будущие данные)
- Журнал транзакций (WAL), хотя он обычно учитывается отдельно.

4. Практика+ (Методы сжатия TOAST).

```

postgres@localhost:postgres> SELECT name, setting FROM pg_settings WHERE name LIKE '%compression%';
+-----+-----+
| name                               | setting |
+-----+-----+
| default_toast_compression          | pglz    |
| wal_compression                    | off     |
+-----+-----+
SELECT 2
Time: 0.029s
postgres@localhost:postgres>

```

Рисунок 4.3 - Методы сжатия

5. Практика+ (Сравнение сжатия)

```

admin@MacBook-Air-Admin ~ % base64 -i /dev/urandom | head -c 11000000 > large_text.txt
admin@MacBook-Air-Admin ~ %

```

Рисунок 4.4 - Создание большого файла

```

postgres@localhost:postgres> CREATE TABLE toast_external (id SERIAL, data TEXT);
ALTER TABLE toast_external ALTER COLUMN data SET STORAGE EXTERNAL;
You're about to run a destructive command.
Do you want to proceed? [y/N]: n
Error: invalid input
You're about to run a destructive command.
Do you want to proceed? [y/N]: y
Your call!
CREATE TABLE
ALTER TABLE
Time: 0.013s
postgres@localhost:postgres> CREATE TABLE toast_pglz (id SERIAL, data TEXT);
CREATE TABLE
Time: 0.006s

```

Рисунок 4.5 - Создание таблицы без сжатия и сжатием

```

postgres=# \set content `cat large_text.txt`
postgres=# \timing on
Секундомер включён.
postgres=# INSERT INTO toast_external (data) VALUES (:'content');
INSERT 0 1
Время: 287,317 мс
postgres=# INSERT INTO toast_pglz (data) VALUES (:'content');
INSERT 0 1
Время: 230,731 мс
postgres=# \timing off
Секундомер выключен.

```

Рисунок 4.6 - Загрузка данных и замерка времени

```

postgres=# SELECT
    'toast_external' as table_name,
    pg_size_pretty(pg_total_relation_size('toast_external')) as size
UNION ALL
SELECT
    'toast_pglz',
    pg_size_pretty(pg_total_relation_size('toast_pglz'));

```

table_name	size
toast_external	11 MB
toast_pglz	11 MB

(2 строки)

Рисунок 4.7 - Сравнение размеров

Вывод: изучил логическую и физическую структуры хранения данных в PostgreSQL. Получил практические навыки управления базами данных, схемами, табличными пространствами. Освоил работу с системным каталогом для извлечения метаданных. Исследовал низкоуровневых аспектов хранения, включая TOAST.