

Организация данных Системный каталог



16

Авторские права

© Postgres Professional, 2017–2024

Авторы: Егор Рогов, Павел Лузанов, Илья Баштанов, Алексей Береснев

Фото: Олег Бартунов (монастырь Пху и пик Бхрикути, Непал)

Использование материалов курса

Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

Обратная связь

Отзывы, замечания и предложения направляйте по адресу:

edu@postgrespro.ru

Отказ от ответственности

Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или косвенным, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Что такое системный каталог и как к нему обращаться

Объекты системного каталога и их расположение

Правила именования объектов

Специальные типы данных

Набор таблиц и представлений,
описывающих все объекты кластера баз данных

Схемы

основная схема: `pg_catalog`

альтернативное представление: `information_schema` (стандарт SQL)

SQL-доступ

просмотр: `SELECT`

изменение: `CREATE, ALTER, DROP`

Доступ в `psql`

специальные команды для удобства просмотра

Системный каталог представляет собой набор таблиц и представлений с описанием всех объектов СУБД, «метаинформация» о содержимом кластера: <https://postgrespro.ru/docs/postgresql/16/catalogs>. Начиная с 14-й версии PostgreSQL для большинства таблиц системного каталога добавлены первичные ключи и ограничения уникальности.

Для доступа к этой информации используются обычные запросы SQL. При помощи команд `SELECT` можно получить описание любых объектов, а при помощи команд DDL (Data Definition Language) можно добавлять и изменять объекты.

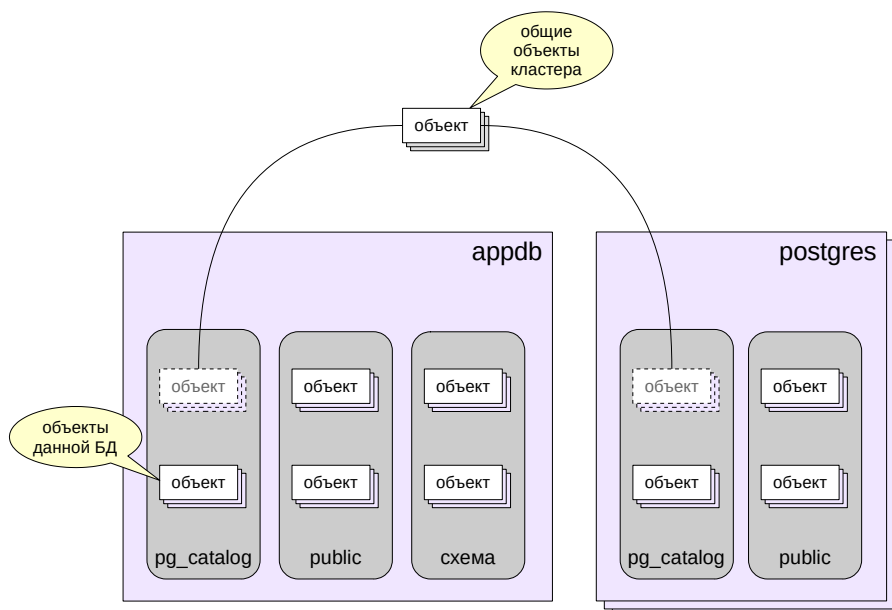
Все таблицы и представления системного каталога располагаются в схеме `pg_catalog`. Существует и другая схема, предписанная стандартом SQL: `information_schema`. Она более стабильна и переносима, чем `pg_catalog`, но не отражает специфику PostgreSQL.

Клиентские программы могут читать содержимое системного каталога и показывать его пользователю в удобном виде. Например, графические среды разработки и управления обычно изображают иерархический «навигатор» объектов.

Программа `psql` тоже предлагает ряд удобных встроенных команд для работы с системным каталогом. Как правило, эти команды начинаются на `\d` (от `describe`). Подробности об этих командах в документации: <https://postgrespro.ru/docs/postgresql/16/app-psql#APP-PSQL-META-COMMANDS>

Наиболее часто используемые из них мы посмотрим в демонстрации. В материалах курса есть файл `catalogs.pdf` со схемой основных таблиц системного каталога и команд `psql` для работы с ними.

Общие объекты кластера



В каждой базе данных кластера создается свой набор таблиц системного каталога. Однако существует несколько объектов каталога, которые являются общими для всего кластера. Наиболее очевидный пример — список самих баз данных.

Эти таблицы хранятся вне какой-либо базы данных, но при этом одинаково видны из каждой БД.

Префиксы имен объектов (таблиц, представлений)
и столбцов

`pg_database.datname`



Названия объектов всегда хранятся в нижнем регистре

Все таблицы и представления системного каталога начинаются с префикса «pg_». Для предотвращения потенциальных конфликтов, не рекомендуется создавать собственные объекты, начинающиеся с «pg_».

Названия столбцов имеют трехбуквенный префикс, который, как правило, соответствует имени таблицы. После префикса нет знака подчеркивания. Имеются некоторые исключения из этого правила, например, столбец `oid` и другие.

Названия объектов хранятся в нижнем регистре.

Некоторые объекты системного каталога

Создадим базу данных и тестовые объекты:

```
=> CREATE DATABASE data_catalog;
```

CREATE DATABASE

```
=> \c data_catalog
```

You are now connected to database "data_catalog" as user "student".

```
=> CREATE TABLE employees(  
    id integer GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    name text,  
    manager integer  
);
```

CREATE TABLE

```
=> CREATE VIEW top_managers AS  
    SELECT * FROM employees WHERE manager IS NULL;
```

CREATE VIEW

```
=> CREATE TEMP TABLE emp_salaries(  
    employee integer,  
    salary numeric  
);
```

CREATE TABLE

Некоторые таблицы системного каталога нам уже знакомы из предыдущей темы. Это базы данных:

```
=> SELECT * FROM pg_database WHERE datname = 'data_catalog' \gx
```

```
-[ RECORD 1 ]--+-  
oid          | 16386  
datname      | data_catalog  
datdba       | 16384  
encoding     | 6  
datlocprovider | c  
datistemplate | f  
dataallowconn | t  
datconnlimit  | -1  
datfrozenxid  | 722  
datminmxid    | 1  
dattablespace | 1663  
datcollate    | en_US.UTF-8  
datctype      | en_US.UTF-8  
datclocale    |  
datcurules    |  
datcollversion | 2.35  
datacl       |
```

И схемы:

```
=> SELECT * FROM pg_namespace WHERE nspname = 'public' \gx
```

```
-[ RECORD 1 ]-----  
oid          | 2200  
nspname      | public  
nspowner     | 6171  
nspacl       | {pg_database_owner=UC/pg_database_owner,=U/pg_database_owner}
```

Важная таблица `pg_class` хранит описание целого ряда объектов: таблиц, представлений, индексов, последовательностей. Все эти объекты называются в PostgreSQL общим словом «отношение» (relation), отсюда и префикс «rel» в названии столбцов:

```
=> SELECT relname, relkind, relnamespace, relfilenode, relowner, relpersistence  
FROM pg_class WHERE relname ~ '^(emp|top)';
```

relname	relkind	relnamespace	relfilenode	relowner	relpersistence
employees_id_seq	S	2200	16387	16384	p
employees	r	2200	16388	16384	p
employees_pkey	i	2200	16393	16384	p
top_managers	v	2200	0	16384	p
emp_salaries	r	16399	16401	16384	t

(5 rows)

Тип объекта определяется столбцом relkind, по значению в relpersistence можно отличить временные объекты от постоянных.

При активном использовании временных объектов в таблицах системного каталога будет возникать большое количество неактуальных версий строк, что может привести к снижению производительности на всех этапах выполнения запроса. В таком случае следует позаботиться о своевременной очистке таблиц системного каталога.

Конечно, для каждого типа объектов в pg_class имеет смысл только часть столбцов; кроме того, удобнее смотреть не на идентификаторы (relnamespace, relowner, и т. д.), а на названия соответствующих объектов. Для этого существуют различные системные представления, например:

```
=> SELECT schemaname, tablename, tableowner
FROM pg_tables WHERE schemaname ~ '(public|pg_temp.+);'
```

schemaname	tablename	tableowner
public	employees	student
pg_temp_4	emp_salaries	student

(2 rows)

```
=> SELECT *
FROM pg_views WHERE schemaname = 'public';
```

schemaname	viewname	viewowner	definition
public	top_managers	student	SELECT id, name, manager FROM employees WHERE (manager IS NULL);

(1 row)

Использование команд psql

В psql встроен набор команд для получения информации из системного каталога. Это короткие команды, пользоваться которыми проще, чем писать запросы к системным таблицам или представлениям.

Получить список таблиц можно командой:

```
=> \dt
```

List of relations			
Schema	Name	Type	Owner
pg_temp_4	emp_salaries	table	student
public	employees	table	student

(2 rows)

Список всех представлений в схеме public:

```
=> \dv public.*
```

List of relations			
Schema	Name	Type	Owner
public	top_managers	view	student

(1 row)

Список таблиц, представлений, индексов и последовательностей:

```
=> \dtvis
```

List of relations				
Schema	Name	Type	Owner	Table
pg_temp_4	emp_salaries	table	student	employees
public	employees	table	student	
public	employees_id_seq	sequence	student	
public	employees_pkey	index	student	
public	top_managers	view	student	

(5 rows)

Эти команды можно снабдить модификатором «+», чтобы получить больше информации:

=> \dt+

List of relations							
Schema	Name	Type	Owner	Persistence	Access method	Size	Description
pg_temp_4	emp_salaries	table	student	temporary	heap	8192 bytes	
public	employees	table	student	permanent	heap	8192 bytes	

(2 rows)

Чтобы получить детальную информацию об отдельном объекте, надо воспользоваться командой \d (без дополнительной буквы):

=> \d top_managers

View "public.top_managers"				
Column	Type	Collation	Nullable	Default
id	integer			
name	text			
manager	integer			

Модификатор «+» остается в силе:

=> \d+ top_managers

View "public.top_managers"						
Column	Type	Collation	Nullable	Default	Storage	Description
id	integer				plain	
name	text				extended	
manager	integer				plain	

View definition:

```
SELECT id,
       name,
       manager
FROM employees
WHERE manager IS NULL;
```

Помимо отношений, аналогичным образом можно смотреть и на другие объекты, такие как схемы (\dn) или функции (\df).

Еще один модификатор «S» позволяет вывести не только пользовательские, но и системные объекты. С помощью шаблона можно ограничить выборку:

=> \dfs pg*size

List of functions				
Schema	Name	Result data type	Argument data types	Type
pg_catalog	pg_column_size	integer	"any"	func
pg_catalog	pg_database_size	bigint	name	func
pg_catalog	pg_database_size	bigint	oid	func
pg_catalog	pg_indexes_size	bigint	regclass	func
pg_catalog	pg_relation_size	bigint	regclass	func
pg_catalog	pg_relation_size	bigint	regclass, text	func
pg_catalog	pg_table_size	bigint	regclass	func
pg_catalog	pg_tablespace_size	bigint	name	func
pg_catalog	pg_tablespace_size	bigint	oid	func
pg_catalog	pg_total_relation_size	bigint	regclass	func

(10 rows)

Как правило, эти команды psql имеют мнемонические имена. Например, \df — describe function, \sf — show function:

```
=> \sf pg_catalog.pg_database_size(oid)

CREATE OR REPLACE FUNCTION pg_catalog.pg_database_size(oid)
  RETURNS bigint
  LANGUAGE internal
  PARALLEL SAFE STRICT
AS $function$pg_database_size_oid$function$
```

Полный список всегда можно посмотреть в документации или командой psql \?.

Изучение структуры системного каталога

Все команды psql, описывающие объекты, обращаются к таблицам системного каталога. Чтобы увидеть эти запросы, следует установить переменную psql ECHO_HIDDEN. Получим, например, информацию о таблице employees:

```
=> \set ECHO_HIDDEN on

=> \dt employees

***** QUERY *****
SELECT n.nspname as "Schema",
       c.relname as "Name",
       CASE c.relkind WHEN 'r' THEN 'table' WHEN 'v' THEN 'view' WHEN 'm' THEN 'materialized
view' WHEN 'i' THEN 'index' WHEN 'S' THEN 'sequence' WHEN 't' THEN 'TOAST table' WHEN 'f'
THEN 'foreign table' WHEN 'p' THEN 'partitioned table' WHEN 'I' THEN 'partitioned index'
END as "Type",
       pg_catalog.pg_get_userbyid(c.relowner) as "Owner"
FROM pg_catalog.pg_class c
     LEFT JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
     LEFT JOIN pg_catalog.pg_am am ON am.oid = c.relam
WHERE c.relkind IN ('r','p','t','s','')
     AND c.relname OPERATOR(pg_catalog.~) '(employees)$' COLLATE pg_catalog.default
     AND pg_catalog.pg_table_is_visible(c.oid)
ORDER BY 1,2;
*****

          List of relations
 Schema |   Name   | Type  | Owner
-----+-----+-----+-----
 public | employees | table | student
(1 row)
```

```
=> \unset ECHO_HIDDEN
```

Для большинства таблиц системного каталога имеются первичные ключи (обычно это столбец oid) и ограничения уникальности. Например, таблица pg_attribute с информацией об атрибутах отношений имеет ограничения:

```
=> \d pg_attribute
```

Table "pg_catalog.pg_attribute"				
Column	Type	Collation	Nullable	Default
attrelid	oid		not null	
attname	name		not null	
atttypid	oid		not null	
attlen	smallint		not null	
attnum	smallint		not null	
attcacheoff	integer		not null	
atttypmod	integer		not null	
attndims	smallint		not null	
attbyval	boolean		not null	
attalign	"char"		not null	
attstorage	"char"		not null	
attcompression	"char"		not null	
attnotnull	boolean		not null	
atthasdef	boolean		not null	
atthasmissing	boolean		not null	
attidentity	"char"		not null	
attgenerated	"char"		not null	
attisdropped	boolean		not null	
attislocal	boolean		not null	
attinhcount	smallint		not null	
attstattarget	smallint		not null	
attcollation	oid		not null	
attacl	aclitem[]			
attoptions	text[]	C		
attfdwoptions	text[]	C		
atmissingval	anyarray			

Indexes:

"pg_attribute_relid_attnum_index" PRIMARY KEY, btree (attrelid, attnum)

"pg_attribute_relid_attnam_index" UNIQUE CONSTRAINT, btree (attrelid, attname)

Ссылочная целостность обеспечивается с помощью ограничений, похожих на внешние ключи, но немного более сложных: ссылающийся столбец может быть массивом ссылающихся элементов, а неопределенность может обозначаться в нем нулем. Получить список таких псевдо-внешних ключей можно функцией `pg_get_catalog_foreign_keys()`. Например, `pg_attribute` "ссылается" на:

```
=> SELECT *
FROM   pg_get_catalog_foreign_keys()
WHERE  fktable = 'pg_attribute'::regclass;
```

fktable	fkcols	pktable	pkcols	is_array	is_opt
pg_attribute	{attrelid}	pg_class	{oid}	f	f
pg_attribute	{atttypid}	pg_type	{oid}	f	t
pg_attribute	{attcollation}	pg_collation	{oid}	f	t

(3 rows)

- `fktable`, `fkcols` — ссылающаяся таблица и ее столбцы;
- `pktable`, `pkcols` — ключ, на который ссылаются;
- `is_array` — является ли ссылающийся столбец массивом;
- `is_opt` — может ли ссылающийся столбец содержать 0.

Тип oid — идентификатор объекта

столбец oid обеспечивает уникальность в таблицах системного каталога
целочисленный тип данных с автоинкрементом

Типы reg*

псевдонимы oid для *некоторых* таблиц системного каталога
(regclass для pg_class и т. п.)

приведение текстового имени объекта к типу oid и обратно

Большинство таблиц системного каталога используют в качестве первичного ключа столбец с именем oid и одноименным типом данных.

Тип oid (object identifier) представляет собой целочисленный тип данных с разрядностью 32 бита (около 4 млрд. значений) и автоинкрементом.

Существует несколько специальных типов данных (фактически псевдонимов oid), начинающихся на «reg», которые позволяют преобразовывать имена объектов в oid и обратно.

<https://postgrespro.ru/docs/postgresql/16/datatype-oid>

Тип oid и reg-типы

Как мы видели, описания таблиц и представлений хранятся в таблице pg_class, а описание столбцов располагаются в отдельной таблице pg_attribute. Чтобы получить список столбцов конкретной таблицы, надо соединить pg_class и pg_attribute:

```
=> SELECT a.attname, a.atttypid
FROM pg_attribute a
WHERE a.attrelid = (
    SELECT oid FROM pg_class WHERE relname = 'employees'
)
AND a.attnum > 0;
```

attname	atttypid
id	23
name	25
manager	23

(3 rows)

Используя reg-типы, запрос можно написать проще, без явного обращения к pg_class:

```
=> SELECT a.attname, a.atttypid
FROM pg_attribute a
WHERE a.attrelid = 'employees'::regclass
AND a.attnum > 0;
```

attname	atttypid
id	23
name	25
manager	23

(3 rows)

Здесь мы преобразовали строку 'employees' к типу oid. Аналогично мы можем вывести oid как текстовое значение:

```
=> SELECT a.attname, a.atttypid::regtype
FROM pg_attribute a
WHERE a.attrelid = 'employees'::regclass
AND a.attnum > 0;
```

attname	atttypid
id	integer
name	text
manager	integer

(3 rows)

Полный список reg-типов:

```
=> \dT reg*
```

List of data types		
Schema	Name	Description
pg_catalog	regclass	registered class
pg_catalog	regcollation	registered collation
pg_catalog	regconfig	registered text search configuration
pg_catalog	regdictionary	registered text search dictionary
pg_catalog	regnamespace	registered namespace
pg_catalog	regoper	registered operator
pg_catalog	regoperator	registered operator (with args)
pg_catalog	regproc	registered procedure
pg_catalog	regprocedure	registered procedure (with args)
pg_catalog	regrole	registered role
pg_catalog	regtype	registered type

(11 rows)

Системный каталог — метаинформация о кластере
в самом кластере

SQL-доступ и дополнительные команды `psql`

Часть таблиц системного каталога хранится в базах данных,
часть — общая для всего кластера

Системный каталог использует специальные типы данных

1. Получите описание таблицы `pg_class`.
2. Получите *подробное* описание представления `pg_tables`.
3. Создайте базу данных и временную таблицу в ней.
Получите полный список схем в базе, включая системные.
4. Получите список представлений в схеме `information_schema`.
5. Какие запросы выполняет следующая команда `psql`?
`\d+ pg_views`

1. Описание pg_class

=> \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	
relhas subclass	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)

2. Подробное описание pg_tables

=> \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"]);
```

3. Полный список схем

```
=> CREATE DATABASE data_catalog;
```

```
CREATE DATABASE
```

```
=> \c data_catalog
```

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

```
=> CREATE TEMP TABLE t(n integer);
```

```
CREATE TABLE
```

```
=> \dnS
```

```
          List of schemas
      Name                | Owner
-----+-----
information_schema       | postgres
pg_catalog                | postgres
pg_temp_4                 | postgres
pg_toast                  | postgres
pg_toast_temp_4          | postgres
public                    | pg_database_owner
(6 rows)
```

Временная таблица расположена в схеме pg_temp_N, где N — некоторое число. Такие схемы создаются для каждого сеанса, в котором появляются временные объекты, поэтому их может быть несколько. Имя схемы для временных объектов текущего сеанса можно получить, обратившись к системной функции:

```
=> SELECT pg_my_temp_schema()::regnamespace;
```

```
pg_my_temp_schema
-----
pg_temp_4
(1 row)
```

Однако в большинстве случаев точное имя схемы знать не нужно, поскольку при необходимости к временному объекту можно обратиться, используя имя схемы pg_temp:

```
=> SELECT * FROM pg_temp.t;
```

```
 n
---
(0 rows)
```

Предназначение некоторых других схем нам уже известно, а с оставшимися (pg_toast*) познакомимся позже.

4. Список представлений в information_schema

Используем шаблон:

```
=> \dv information_schema.*
```


Schema	List of relations		Type	Owner
	Name			
information_schema	_pg_foreign_data_wrappers		view	postgres
information_schema	_pg_foreign_servers		view	postgres
information_schema	_pg_foreign_table_columns		view	postgres
information_schema	_pg_foreign_tables		view	postgres
information_schema	_pg_user_mappings		view	postgres
information_schema	administrable_role_authorizations		view	postgres
information_schema	applicable_roles		view	postgres
information_schema	attributes		view	postgres
information_schema	character_sets		view	postgres
information_schema	check_constraint_routine_usage		view	postgres
information_schema	check_constraints		view	postgres
information_schema	collation_character_set_applicability		view	postgres
information_schema	collations		view	postgres
information_schema	column_column_usage		view	postgres
information_schema	column_domain_usage		view	postgres
information_schema	column_options		view	postgres
information_schema	column_privileges		view	postgres
information_schema	column_udt_usage		view	postgres
information_schema	columns		view	postgres
information_schema	constraint_column_usage		view	postgres
information_schema	constraint_table_usage		view	postgres
information_schema	data_type_privileges		view	postgres
information_schema	domain_constraints		view	postgres
information_schema	domain_udt_usage		view	postgres
information_schema	domains		view	postgres
information_schema	element_types		view	postgres
information_schema	enabled_roles		view	postgres
information_schema	foreign_data_wrapper_options		view	postgres
information_schema	foreign_data_wrappers		view	postgres
information_schema	foreign_server_options		view	postgres
information_schema	foreign_servers		view	postgres
information_schema	foreign_table_options		view	postgres
information_schema	foreign_tables		view	postgres
information_schema	information_schema_catalog_name		view	postgres
information_schema	key_column_usage		view	postgres
information_schema	parameters		view	postgres
information_schema	referential_constraints		view	postgres
information_schema	role_column_grants		view	postgres
information_schema	role_routine_grants		view	postgres
information_schema	role_table_grants		view	postgres
information_schema	role_udt_grants		view	postgres
information_schema	role_usage_grants		view	postgres
information_schema	routine_column_usage		view	postgres
information_schema	routine_privileges		view	postgres
information_schema	routine_routine_usage		view	postgres
information_schema	routine_sequence_usage		view	postgres
information_schema	routine_table_usage		view	postgres
information_schema	routines		view	postgres
information_schema	schemata		view	postgres
information_schema	sequences		view	postgres
information_schema	table_constraints		view	postgres
information_schema	table_privileges		view	postgres
information_schema	tables		view	postgres
information_schema	transforms		view	postgres
information_schema	triggered_update_columns		view	postgres
information_schema	triggers		view	postgres
information_schema	udt_privileges		view	postgres
information_schema	usage_privileges		view	postgres
information_schema	user_defined_types		view	postgres
information_schema	user_mapping_options		view	postgres
information_schema	user_mappings		view	postgres
information_schema	view_column_usage		view	postgres
information_schema	view_routine_usage		view	postgres
information_schema	view_table_usage		view	postgres
information_schema	views		view	postgres

(65 rows)

5. Запросы к системному каталогу

Чтобы увидеть запросы, которые выполняют команды `psql`, включим переменную `ECHO_HIDDEN`.

```
=> \set ECHO_HIDDEN on
```

```
=> \d+ pg_views
```

***** QUERY *****

```
SELECT c.oid,
       n.nspname,
       c.relname
FROM pg_catalog.pg_class c
     LEFT JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
WHERE c.relname OPERATOR(pg_catalog.~) '^(pg_views)$' COLLATE pg_catalog.default
      AND pg_catalog.pg_table_is_visible(c.oid)
ORDER BY 2, 3;
*****
```

***** QUERY *****

```
SELECT c.relchecks, c.relkind, c.relhasindex, c.relhasrules, c.relhastriggers,
       c.relrowsecurity, c.relfrowsecurity, false AS relhasoids, c.relispartition,
       pg_catalog.array_to_string(c.reloptions || array(select 'toast.' || x from
pg_catalog.unnest(tc.reloptions) x), ' ', ' ')
, c.reltablename, CASE WHEN c.reloftype = 0 THEN '' ELSE
c.reloftype::pg_catalog.regtype::pg_catalog.text END, c.relpersistence, c.relreplident,
am.amname
FROM pg_catalog.pg_class c
     LEFT JOIN pg_catalog.pg_class tc ON (c.reltoastrelid = tc.oid)
     LEFT JOIN pg_catalog.pg_am am ON (c.relam = am.oid)
WHERE c.oid = '12028';
*****
```

***** QUERY *****

```
SELECT a.attname,
       pg_catalog.format_type(a.atttypid, a.atttypmod),
       (SELECT pg_catalog.pg_get_expr(d.adbin, d.adrelid, true)
        FROM pg_catalog.pg_attrdef d
        WHERE d.adrelid = a.attrelid AND d.adnum = a.attnum AND a.attahasdef),
       a.attnotnull,
       (SELECT c.collname FROM pg_catalog.pg_collation c, pg_catalog.pg_type t
        WHERE c.oid = a.attcollation AND t.oid = a.atttypid AND a.attcollation <>
t.typcollation) AS attcollation,
       a.attidentity,
       a.attgenerated,
       a.attstorage,
       pg_catalog.col_description(a.attrelid, a.attnum)
FROM pg_catalog.pg_attribute a
WHERE a.attrelid = '12028' AND a.attnum > 0 AND NOT a.attisdropped
ORDER BY a.attnum;
*****
```

***** QUERY *****

```
SELECT pg_catalog.pg_get_viewdef('12028'::pg_catalog.oid, true);
*****
```

***** QUERY *****

```
SELECT r.rulename, trim(trailing ';' from pg_catalog.pg_get_ruledef(r.oid, true))
FROM pg_catalog.pg_rewrite r
WHERE r.ev_class = '12028' AND r.rulename != '_RETURN' ORDER BY 1;
*****
```

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";
```

Для формирования вывода потребовалось выполнить пять запросов.

=> \set ECHO_HIDDEN off