

Университет ИТМО  
Факультет программной инженерии и компьютерной техники

# Распределённые системы хранения данных. Лабораторная работа №1.

Группа: Р33131  
Студент: Смирнов Виктор Игоревич  
Преподаватель: Афанасьев Дмитрий Борисович  
Вариант: 776

# Ключевые слова

База данных, PostgreSQL, системный каталог.

## Содержание

<a href="#">1 Цель работы</a>	1
<a href="#">2 Текст задания</a>	1
<a href="#">3 Реализация скрипта</a>	2
<a href="#">4 Таблица</a>	8
<a href="#">5 Вывод</a>	9

## 1 Цель работы

Научиться проектировать базы данных, составлять инфологические и даталогические модели данных, реализовывать их в БД PostgreSQL, научиться выполнять запросы.

## 2 Текст задания

Используя сведения из системных каталогов получить информацию о любой таблице: Номер по порядку, Имя столбца, Атрибуты (в атрибуты столбца включить тип данных, ограничение типа CHECK).

Пример вывода:

Таблица: Н\_ЛЮДИ

Но.	Имя столбца	Атрибуты
1	ИД	Type : NUMBER(9) NOT NULL Comment : 'Уникальный номер человека'
2	ФАМИЛИЯ	Type : VARCHAR2(25) NOT NULL Comment : 'Фамилия человека'
3	ИМЯ	Type : VARCHAR2(2000) NOT NULL Comment : 'Имя человека'
4	ОТЧЕСТВО	Type : VARCHAR2(20) Comment : 'Отчество человека'
5	ДАТА_РОЖДЕНИЯ	Type : DATE NOT NULL Comment : 'Дата рождения человека'
6	ПОЛ	Type : CHAR(1) NOT NULL Constr : "AVCON_378561_ПОЛ_000" CHECK (ПОЛ IN ('М', 'Ж')) Constr : "AVCON_388176_ПОЛ_000" CHECK (ПОЛ IN ('М', 'Ж')) Comment : 'Пол человека'
7	ИНОСТРАН	Type : VARCHAR2(3) NOT NULL
8	КТО_СОЗДАЛ	Type : VARCHAR2(40) NOT NULL
9	КОГДА_СОЗДАЛ	Type : DATE NOT NULL
10	КТО_ИЗМЕНИЛ	Type : VARCHAR2(40) NOT NULL
11	КОГДА_ИЗМЕНИ	Type : DATE NOT NULL
12	ДАТА_СМЕРТИ	Type : DATE Comment : 'Дата смерти человека'
13	ПИН	Type : VARCHAR2(20)
14	ИНН	Type : VARCHAR2(20)

Далее был написан SQL скрипт, создающий таблицу, аналогичную той, что в примере.

```

1 drop table person;
2 create table person (
3     id numeric(9, 2) primary key,
4     last_name varchar(25) not null,
5     first_name varchar(2000) not null,
6     patronymic varchar(20),
7     birth_date date not null,
8     gender char(1) not null,
9     foreigner varchar(3) not null,
10    created_who varchar(40) not null,
11    created_when date not null,
12    edited_who varchar(40) not null,
13    edited_when date not null,
14    death_date date,
15    pin varchar(20),
16    inn varchar(20),
17
18    check (gender in ('M', 'F')),
19    check (gender in ('M', 'F')),
20    check (
21        length(patronymic) > 10 AND
22        length(last_name) > 10 AND
23        length(first_name) > 10
24    ),
25    unique (last_name, first_name, patronymic),
26    unique (inn),
27    unique (pin)
28 );
29
30 drop table if exists item;
31 create table item (
32     id1 integer,
33     id2 integer,
34
35     id11 integer,
36     id12 integer,
37
38     primary key (id1, id2),
39     foreign key (id11, id12) references item(id1, id2)
40 );
41
42 comment on column person.id is 'The unique number of the person';
43 comment on column person.id is 'The unique number of the person';
44 comment on column person.last_name is 'Last name of the person';
45 comment on column person.first_name is 'The name of the person';
46 comment on column person.patronymic is 'The patronymic of the person';
47 comment on column person.birth_date is 'Date of birth of a person';
48 comment on column person.death_date is 'Date of death of a person';

```

### 3 Реализация скрипта

```

1 DROP VIEW IF EXISTS meta_namespace CASCADE;
2 CREATE VIEW meta_namespace AS
3     SELECT
4         pg_namespace.oid AS id,
5         pg_namespace.nspname AS name
6     FROM pg_namespace;
7
8 DROP VIEW IF EXISTS meta_table CASCADE;
9 CREATE VIEW meta_table AS
10    SELECT
11        pg_class.oid AS id,
12        pg_class.relname AS name,
13        pg_class.relnamespace AS namespace_id
14    FROM pg_class;
15
16 DROP VIEW IF EXISTS meta_table_column CASCADE;
17 CREATE VIEW meta_table_column AS
18    SELECT
19        pg_attribute.attrelid AS table_id,
20        pg_attribute.attnum AS number,

```

```

21     pg_attribute.attnname          AS name,
22     pg_attribute.atttypid         AS type_id,
23     (NOT pg_attribute.attnotnull) AS is_nullable
24 FROM pg_attribute;
25
26 DROP VIEW IF EXISTS meta_type CASCADE;
27 CREATE VIEW meta_type AS
28 SELECT
29     pg_type.oid          AS id,
30     pg_type.typname AS name
31 FROM pg_type;
32
33 DROP VIEW IF EXISTS meta_constraint_check CASCADE;
34 CREATE VIEW meta_constraint_check AS
35 SELECT
36     pg_constraint.oid          AS id,
37     pg_constraint.conname      AS name,
38     pg_constraint.connamespace AS namespace_id,
39     pg_constraint.conrelid     AS constrained_table_id,
40     pg_constraint.conkey       AS
41     constrained_column_numbers,
42     pg_get_expr(pg_constraint.conbin, COALESCE(pg_class.oid, 0)) AS clause
43 FROM pg_constraint
44 LEFT JOIN pg_class ON pg_class.oid = pg_constraint.conrelid
45 WHERE pg_constraint.contype = 'c';
46
47 DROP VIEW IF EXISTS meta_constraint_foreign_key CASCADE;
48 CREATE VIEW meta_constraint_foreign_key AS
49 SELECT
50     pg_constraint.oid          AS id,
51     pg_constraint.conname      AS name,
52     pg_constraint.connamespace AS namespace_id,
53     pg_constraint.conrelid     AS constrained_table_id,
54     pg_constraint.conkey       AS constrained_column_numbers,
55     pg_constraint.confrelid    AS referenced_table_id,
56     pg_constraint.confkey      AS referenced_column_numbers
57 FROM pg_constraint
58 WHERE pg_constraint.contype = 'f';
59
60 DROP VIEW IF EXISTS meta_constraint_primary_key CASCADE;
61 CREATE VIEW meta_constraint_primary_key AS
62 SELECT
63     pg_constraint.oid          AS id,
64     pg_constraint.conname      AS name,
65     pg_constraint.connamespace AS namespace_id,
66     pg_constraint.conrelid     AS constrained_table_id,
67     pg_constraint.conkey       AS constrained_column_numbers
68 FROM pg_constraint
69 WHERE pg_constraint.contype = 'p';
70
71 DROP VIEW IF EXISTS meta_constraint_unique CASCADE;
72 CREATE VIEW meta_constraint_unique AS
73 SELECT
74     pg_constraint.oid          AS id,
75     pg_constraint.conname      AS name,
76     pg_constraint.connamespace AS namespace_id,
77     pg_constraint.conrelid     AS constrained_table_id,
78     pg_constraint.conkey       AS constrained_column_numbers
79 FROM pg_constraint
80 WHERE pg_constraint.contype = 'u';
81
82 -- TODO: t = constraint trigger
83 -- TODO: x = exclusion constraint
84
85 -- SELECT * FROM meta_namespace;
86 -- SELECT * FROM meta_table;
87 -- SELECT * FROM meta_table_column;
88 -- SELECT * FROM meta_constraint_check;
89 -- SELECT * FROM meta_constraint_foreign_key;
90 -- SELECT * FROM meta_constraint_primary_key;
91 -- SELECT * FROM meta_constraint_unique;

```

```

1 DROP VIEW IF EXISTS meta_display_constraint_check CASCADE;
2 CREATE VIEW meta_display_constraint_check AS
3 SELECT
4     meta_constraint_check.id AS id,
5     meta_constraint_check.name AS name,
6     meta_constraint_check.namespace_id AS namespace_id,
7     meta_constraint_check.constrained_table_id AS constrained_table_id,
8     meta_constraint_check.constrained_column_numbers AS constrained_column_numbers,
9     meta_constraint_check.clause AS clause
10 FROM meta_constraint_check;
11
12 DROP VIEW IF EXISTS meta_display_constraint_check_single CASCADE;
13 CREATE VIEW meta_display_constraint_check_single AS
14 SELECT
15     meta_display_constraint_check.id AS id,
16     meta_display_constraint_check.name AS name,
17     meta_display_constraint_check.namespace_id AS namespace_id,
18     meta_display_constraint_check.constrained_table_id AS constrained_table_id,
19     meta_display_constraint_check.constrained_column_numbers[1] AS
20     constrained_column_number,
21     meta_display_constraint_check.clause AS clause
22 FROM meta_display_constraint_check
23 WHERE cardinality(meta_display_constraint_check.constrained_column_numbers) = 1;
24
25 DROP VIEW IF EXISTS meta_display_constraint_check_multiple CASCADE;
26 CREATE VIEW meta_display_constraint_check_multiple AS
27 SELECT
28     meta_display_constraint_check.id AS id,
29     meta_display_constraint_check.name AS name,
30     meta_display_constraint_check.namespace_id AS namespace_id,
31     meta_display_constraint_check.constrained_table_id AS constrained_table_id,
32     meta_display_constraint_check.constrained_column_numbers AS
33     constrained_column_numbers,
34     meta_display_constraint_check.clause AS clause
35 FROM meta_display_constraint_check
36 WHERE cardinality(meta_display_constraint_check.constrained_column_numbers) != 1;
37
38 DROP VIEW IF EXISTS meta_display_constraint_foreign_key_single CASCADE;
39 CREATE VIEW meta_display_constraint_foreign_key_single AS
40 SELECT
41     meta_constraint_foreign_key.id AS id,
42     meta_constraint_foreign_key.name AS name,
43     meta_constraint_foreign_key.namespace_id AS namespace_id,
44     meta_constraint_foreign_key.constrained_table_id AS constrained_table_id,
45     meta_constraint_foreign_key.constrained_column_numbers[1] AS
46     constrained_column_number,
47     ('REFERENCES ' || meta_table_column.name::text) AS clause
48 FROM meta_constraint_foreign_key
49 JOIN meta_table ON meta_table.id = meta_constraint_foreign_key.
50 referenced_table_id
51 JOIN meta_table_column ON (
52     meta_table_column.table_id = meta_table.id AND
53     meta_table_column.number = meta_constraint_foreign_key.referenced_column_numbers[1]
54 )
55 WHERE (
56     cardinality(meta_constraint_foreign_key.constrained_column_numbers) = 1 AND
57     cardinality(meta_constraint_foreign_key.referenced_column_numbers) = 1
58 );
59
60 DROP FUNCTION IF EXISTS meta_display_column_name CASCADE;
61 CREATE FUNCTION meta_display_column_name(
62     table_id oid,
63     column_number integer
64 ) RETURNS text AS $$
65 DECLARE
66     column_name text;
67 BEGIN
68     SELECT meta_table_column.name INTO column_name
69     FROM meta_table
70     JOIN meta_table_column ON meta_table_column.table_id = meta_table.id
71     WHERE meta_table.id = meta_display_column_name.table_id
72     AND meta_table_column.number = meta_display_column_name.column_number;
73

```

```

70 RETURN column_name;
71 END;
72 $$ LANGUAGE plpgsql;
73
74 DROP VIEW IF EXISTS meta_display_constraint_foreign_key_multiple CASCADE;
75 CREATE VIEW meta_display_constraint_foreign_key_multiple AS
76 SELECT
77     meta_constraint_foreign_key.id AS id,
78     meta_constraint_foreign_key.name AS name,
79     meta_constraint_foreign_key.namespace_id AS namespace_id,
80     meta_constraint_foreign_key.constrained_table_id AS constrained_table_id,
81     meta_constraint_foreign_key.constrained_column_numbers AS constrained_column_numbers
82 ,
83     meta_constraint_foreign_key.referenced_table_id AS referenced_table_id,
84     meta_constraint_foreign_key.referenced_column_numbers AS referenced_column_numbers,
85     (
86         SELECT string_agg(meta_display_column_name(constrained_table_id,
87             constrained_column_number), ', ')
88         FROM unnest(meta_constraint_foreign_key.constrained_column_numbers)
89         AS constrained_column_number
90     ) || ' REFERENCES ' || (
91         SELECT string_agg(meta_display_column_name(referenced_table_id,
92             referenced_column_number), ', ')
93         FROM unnest(meta_constraint_foreign_key.referenced_column_numbers)
94         AS referenced_column_number
95     )
96     AS clause
97 FROM meta_constraint_foreign_key
98 WHERE (
99     cardinality(meta_constraint_foreign_key.constrained_column_numbers) != 1 AND
100     cardinality(meta_constraint_foreign_key.referenced_column_numbers) != 1
101 );
102
103 DROP VIEW IF EXISTS meta_display_constraint_primary_key_single CASCADE;
104 CREATE VIEW meta_display_constraint_primary_key_single AS
105 SELECT
106     meta_constraint_primary_key.id AS id,
107     meta_constraint_primary_key.name AS name,
108     meta_constraint_primary_key.namespace_id AS namespace_id,
109     meta_constraint_primary_key.constrained_table_id AS constrained_table_id,
110     meta_constraint_primary_key.constrained_column_numbers[1] AS
111     constrained_column_number,
112     'PRIMARY KEY' AS clause
113 FROM meta_constraint_primary_key
114 WHERE cardinality(meta_constraint_primary_key.constrained_column_numbers) = 1;
115
116 DROP VIEW IF EXISTS meta_display_constraint_primary_key_multiple CASCADE;
117 CREATE VIEW meta_display_constraint_primary_key_multiple AS
118 SELECT
119     meta_constraint_primary_key.id AS id,
120     meta_constraint_primary_key.name AS name,
121     meta_constraint_primary_key.namespace_id AS namespace_id,
122     meta_constraint_primary_key.constrained_table_id AS constrained_table_id,
123     meta_constraint_primary_key.constrained_column_numbers AS
124     constrained_column_numbers,
125     (
126         'PRIMARY KEY ' || (
127             SELECT string_agg(meta_display_column_name(constrained_table_id,
128                 constrained_column_number), ', ')
129             FROM unnest(meta_constraint_primary_key.constrained_column_numbers)
130             AS constrained_column_number
131         )
132     )
133     AS clause
134 FROM meta_constraint_primary_key
135 WHERE cardinality(meta_constraint_primary_key.constrained_column_numbers) != 1;
136
137 DROP VIEW IF EXISTS meta_display_constraint_unique_single CASCADE;
138 CREATE VIEW meta_display_constraint_unique_single AS
139 SELECT
140     meta_constraint_unique.id AS id,
141     meta_constraint_unique.name AS name,
142     meta_constraint_unique.namespace_id AS namespace_id,

```

```

137     meta_constraint_unique.constrained_table_id           AS constrained_table_id,
138     meta_constraint_unique.constrained_column_numbers[1] AS constrained_column_number,
139     'UNIQUE'                                             AS clause
140 FROM meta_constraint_unique
141 WHERE cardinality(meta_constraint_unique.constrained_column_numbers) = 1;
142
143 DROP VIEW IF EXISTS meta_display_constraint_unique_multiple CASCADE;
144 CREATE VIEW meta_display_constraint_unique_multiple AS
145     SELECT
146         meta_constraint_unique.id                AS id,
147         meta_constraint_unique.name              AS name,
148         meta_constraint_unique.namespace_id      AS namespace_id,
149         meta_constraint_unique.constrained_table_id AS constrained_table_id,
150         meta_constraint_unique.constrained_column_numbers AS constrained_column_numbers,
151         (
152             'UNIQUE ' || (
153                 SELECT string_agg(meta_display_column_name(constrained_table_id,
154                     constrained_column_number), ', ')
155                 FROM unnest(meta_constraint_unique.constrained_column_numbers)
156                 AS constrained_column_number
157             )
158         ) AS clause
159 FROM meta_constraint_unique
160 WHERE cardinality(meta_constraint_unique.constrained_column_numbers) != 1;
161
162 DROP VIEW IF EXISTS meta_display_constraint_single CASCADE;
163 CREATE VIEW meta_display_constraint_single AS
164     (
165         SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
166         clause
167         FROM meta_display_constraint_check_single
168     ) UNION ALL (
169         SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
170         clause
171         FROM meta_display_constraint_foreign_key_single
172     ) UNION ALL (
173         SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
174         clause
175         FROM meta_display_constraint_primary_key_single
176     ) UNION ALL (
177         SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
178         clause
179         FROM meta_display_constraint_unique_single
180     );
181
182 DROP VIEW IF EXISTS meta_display_constraint_multiple CASCADE;
183 CREATE VIEW meta_display_constraint_multiple AS
184     (
185         SELECT id, name, namespace_id, constrained_table_id, clause
186         FROM meta_display_constraint_check_multiple
187     ) UNION ALL (
188         SELECT id, name, namespace_id, constrained_table_id, clause
189         FROM meta_display_constraint_foreign_key_multiple
190     ) UNION ALL (
191         SELECT id, name, namespace_id, constrained_table_id, clause
192         FROM meta_display_constraint_primary_key_multiple
193     ) UNION ALL (
194         SELECT id, name, namespace_id, constrained_table_id, clause
195         FROM meta_display_constraint_unique_multiple
196     );
197
198 1 DROP VIEW IF EXISTS main_table_column_constraint CASCADE;
199 2 CREATE VIEW main_table_column_constraint AS
200 3     SELECT
201 4         meta_namespace.name                AS schema_name,
202 5         meta_table.name                   AS table_name,
203 6         meta_table_column.name            AS column_name,
204 7         meta_display_constraint_single.name AS constraint_name,
205 8         meta_display_constraint_single.clause AS constraint_clause
206 9 FROM meta_table
20710 JOIN meta_namespace ON meta_table.namespace_id = meta_namespace.id
20811 JOIN meta_table_column
20912     ON meta_table_column.table_id = meta_table.id

```

```

13 LEFT JOIN meta_display_constraint_single ON (
14     meta_display_constraint_single.constrained_table_id = meta_table.id AND
15     meta_display_constraint_single.constrained_column_number = meta_table_column.number
16 );
17
18 DROP VIEW IF EXISTS main_table_constraint CASCADE;
19 CREATE VIEW main_table_constraint AS
20 SELECT
21     meta_namespace.name AS schema_name,
22     meta_table.name AS table_name,
23     meta_display_constraint_multiple.name AS constraint_name,
24     meta_display_constraint_multiple.clause AS constraint_clause
25 FROM meta_table
26 JOIN meta_namespace ON meta_table.namespace_id = meta_namespace.id
27 LEFT JOIN meta_display_constraint_multiple ON (
28     meta_display_constraint_multiple.constrained_table_id = meta_table.id
29 );
30
31 DROP PROCEDURE IF EXISTS main_table_column_pretty;
32 CREATE PROCEDURE main_table_print_pretty (
33     table_schema text,
34     table_name text
35 ) AS $$
36 DECLARE
37     col record;
38     col_constr record;
39
40     C1W integer;
41     C2W integer;
42     C31W integer;
43     C32W integer;
44     REM integer;
45 BEGIN
46     C1W := 2;
47     C2W := 12;
48     C31W := 8;
49     C32W := 64 + 8;
50     REM := 11;
51
52     ----- HEADER -----
53     RAISE INFO
54         '%',
55         rpad(
56             '|--- Table "' || table_schema || '.' || table_name || '" Information ',
57             C1W + C2W + C31W + C32W + REM,
58             '-_'
59         ) || '|';
60
61     RAISE INFO
62         '| % | % | % |',
63         rpad('N', C1W, ' '),
64         rpad('Name', C2W, ' '),
65         rpad('Attributes', C31W + C32W + 2, ' ');
66
67     RAISE INFO
68         '%',
69         rpad('|', C1W + C2W + C31W + C32W + REM, '-_') || '|';
70
71
72     ----- ROWS -----
73     FOR col IN
74         SELECT
75             meta_table_column.name AS column_name,
76             meta_type.name AS type_name
77         FROM meta_table
78         JOIN meta_namespace ON meta_namespace.id = meta_table.namespace_id
79         JOIN meta_table_column ON meta_table.id = meta_table_column.table_id
80         JOIN meta_type ON meta_type.id = meta_table_column.type_id
81         WHERE meta_namespace.name = main_table_print_pretty.table_schema
82             AND meta_table.name = main_table_print_pretty.table_name
83     LOOP
84         RAISE INFO
85             '| % | % | % |',

```



```

86         rpad(' ', C1W, ' '),
87         rpad(col.column_name, C2W, ' '),
88         (rpad('Type', C31W, ' ') || ': ' || rpad(col.type_name, C32W, ' '));
89     FOR col_constr IN
90         SELECT
91             constraint_name AS name,
92             constraint_clause AS clause
93         FROM main_table_column_constraint
94         WHERE
95             main_table_column_constraint.schema_name = main_table_print_pretty.table_schema
96     AND
97         main_table_column_constraint.table_name = main_table_print_pretty.table_name AND
98         main_table_column_constraint.column_name = col.column_name
99     LOOP
100         IF NOT col_constr.name IS NULL THEN
101             RAISE INFO
102                 '| % | % | % |',
103                 rpad(' ', C1W, ' '),
104                 rpad(' ', C2W, ' '),
105                 (
106                     rpad('Constr', C31W, ' ') || ': ' || rpad(
107                         (col_constr.name || ' ' || col_constr.clause), C32W, ' '
108                     )
109                 );
110         END IF;
111     END LOOP;
112 END LOOP;
113
114 FOR col IN
115     SELECT
116         main_table_constraint.constraint_name AS constraint_name,
117         main_table_constraint.constraint_clause AS constraint_clause
118     FROM main_table_constraint
119     WHERE
120         main_table_constraint.schema_name = main_table_print_pretty.table_schema AND
121         main_table_constraint.table_name = main_table_print_pretty.table_name
122     LOOP
123         RAISE INFO
124             '| % |',
125             (rpad('Constr', C31W, ' ') || ': ' ||
126              (col.constraint_name || ' ' || col.constraint_clause))
127         ;
128     END LOOP;
129 END;
130 $$ language plpgsql;
131 CALL main_table_print_pretty('public', 'person');

```

## 4 Таблица

```

1 psql:main.sql:131: INFO: |--- Table "public.person" Information
2 psql:main.sql:131: INFO: | N | Name | Attributes
3 psql:main.sql:131: INFO: |-----|
4 psql:main.sql:131: INFO: | | tableoid | Type : oid
5 psql:main.sql:131: INFO: | | cmax | Type : cid
6 psql:main.sql:131: INFO: | | xmax | Type : xid
7 psql:main.sql:131: INFO: | | cmin | Type : cid
8 psql:main.sql:131: INFO: | | xmin | Type : xid
9 psql:main.sql:131: INFO: | | ctid | Type : tid
10 psql:main.sql:131: INFO: | | id | Type : numeric
11 psql:main.sql:131: INFO: | | | Constr : person_pkey PRIMARY KEY

```

```

12 psql:main.sql:131: INFO: |      | last_name | Type      : varchar
13 psql:main.sql:131: INFO: |      | first_name  | Type      : varchar
14 psql:main.sql:131: INFO: |      | patronymic  | Type      : varchar
15 psql:main.sql:131: INFO: |      | birth_date  | Type      : date
16 psql:main.sql:131: INFO: |      | gender       | Type      : bpchar
17 psql:main.sql:131: INFO: |      |              | Constr    : person_gender_check (gender =
    ANY (ARRAY['M'::bpchar, 'F'::bpchar]))
18 psql:main.sql:131: INFO: |      |              | Constr    : person_gender_check1 (gender =
    ANY (ARRAY['M'::bpchar, 'F'::bpchar]))
19 psql:main.sql:131: INFO: |      | foreigner   | Type      : varchar
20 psql:main.sql:131: INFO: |      | created_who  | Type      : varchar
21 psql:main.sql:131: INFO: |      | created_when | Type      : date
22 psql:main.sql:131: INFO: |      | edited_who   | Type      : varchar
23 psql:main.sql:131: INFO: |      | edited_when  | Type      : date
24 psql:main.sql:131: INFO: |      | death_date  | Type      : date
25 psql:main.sql:131: INFO: |      | pin          | Type      : varchar
26 psql:main.sql:131: INFO: |      |              | Constr    : person_pin_key UNIQUE
27 psql:main.sql:131: INFO: |      | inn          | Type      : varchar
28 psql:main.sql:131: INFO: |      |              | Constr    : person_inn_key UNIQUE
29 psql:main.sql:131: INFO: | Constr : person_check ((length((patronymic)::text) > 10)
    AND (length((last_name)::text) > 10) AND (length((first_name)::text) > 10)) |
30 psql:main.sql:131: INFO: | Constr : person_last_name_first_name_patronymic_key UNIQUE
    last_name, first_name, patronymic |

```

## 5 Вывод

Данная лабораторная работа помогла мне изучить системный каталог PostgreSQL.

## Список литературы