

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

1	Цель работы	1
2	Текст задания	1
3	Создание локального окружения для тестирования	2
4	Реализация скрипта	2
5	Вывод	6
6	Вывод	7

## 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 ЧЕКС).

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" ЧЕКС (ПОЛ IN ('М', 'Ж')) Constr : "AVCON_388176_ПОЛ_000" ЧЕКС (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)

### 3 Создание локального окружения для тестирования

Для локального взаимодействия я решил поднять PostgreSQL в Docker контейнере.

```
1 version: '3.8'
2 services:
3   database:
4     container_name: database
5     image: postgres
6     restart: always
7     ports:
8       - 5432:5432
9     volumes:
10      - ../workspace
11     environment:
12       POSTGRES_USER: postgres
13       POSTGRES_PASSWORD: postgres
14       POSTGRES_DB: postgres
15     networks:
16       - common
17 networks:
18   common:
19     name: common
20     driver: bridge
```

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 );
21
22 comment on column person.id is 'The unique number of the person';
23 comment on column person.id is 'The unique number of the person';
24 comment on column person.last_name is 'Last name of the person';
25 comment on column person.first_name is 'The name of the person';
26 comment on column person.patronymic is 'The patronymic of the person';
27 comment on column person.birth_date is 'Date of birth of a person';
28 comment on column person.death_date is 'Date of death of a person';
```

### 4 Реализация скрипта

```
1 create or replace function table_column(
2   table_schema text,
3   table_name text
4 ) returns table (
5   number integer,
6   name text,
7   type text,
8   is_nullable boolean,
9   comment text,
10  constraint_name text,
11  check_clause text
12 ) as $$
13 select
```

```

14 attnum as number,
15 information_schema.columns.column_name as name,
16 (
17     data_type || (
18         case
19             when character_maximum_length is not null then
20                 ' (' || cast(character_maximum_length as text) || ')',
21             else
22                 ''
23         end
24     ) || (
25         (
26             case
27                 when (
28                     numeric_precision is not null or
29                     numeric_scale is not null
30                 ) then
31                     ' ('
32                 else
33                     ''
34             end
35         ) || (
36             case
37                 when numeric_precision is not null then
38                     cast(numeric_precision as text)
39                 else
40                     ''
41             end
42         ) || (
43             case
44                 when numeric_scale is not null then
45                     ', ' || cast(numeric_scale as text)
46                 else
47                     ''
48             end
49         ) || (
50             case
51                 when (
52                     numeric_precision is not null or
53                     numeric_scale is not null
54                 ) then
55                     '),'
56                 else
57                     ''
58             end
59         )
60     )
61 ) as type,
62 (
63     information_schema.columns.is_nullable = 'YES'
64 ) as is_nullable,
65 pg_description.description as comment,
66 information_schema.check_constraints.constraint_name as constraint_name,
67 information_schema.check_constraints.check_clause as check_clause
68 from information_schema.columns
69 left join information_schema.constraint_column_usage on (
70     constraint_column_usage.table_schema = table_column.table_schema and
71     constraint_column_usage.table_name = table_column.table_name and
72     constraint_column_usage.column_name = information_schema.columns.column_name
73 )
74 left join information_schema.check_constraints on (
75     check_constraints.constraint_schema = constraint_column_usage.constraint_schema and
76     check_constraints.constraint_name = constraint_column_usage.constraint_name
77 )
78 left join pg_attribute on (
79     pg_attribute.attrelid = (
80         SELECT oid
81         FROM pg_class
82         WHERE (
83             pg_class.relname = table_column.table_name and
84             pg_class.relnamespace = (
85                 select oid
86                 from pg_namespace

```

```

87         where pg_namespace.nspname = table_column.table_schema
88     )
89 )
90 ) and
91 pg_attribute.attname = information_schema.columns.column_name
92 )
93 left join pg_description on (
94     pg_description.objoid = pg_attribute.attrelid and
95     pg_description.objsubid = pg_attribute.attnum
96 )
97 where (
98     information_schema.columns.table_schema = table_column.table_schema and
99     information_schema.columns.table_name = table_column.table_name
100 );
101 $$ language sql;
102
103 create or replace function table_column_pretty(
104     table_schema text,
105     table_name text
106 ) returns table (
107     number integer,
108     name text,
109     attributes text
110 ) as $$
111 select
112     number,
113     name,
114     type || (
115         case
116             when constr is not null then
117                 e'\n' || constr
118             else
119                 ''
120         end
121     ) || (
122         case
123             when comment is not null then
124                 e'\n' || comment
125             else
126                 ''
127         end
128     ) as attributes
129 from (
130     select
131         number,
132         name,
133         'Type: ' || cast(type as text) || ' ' || (
134             case
135                 when not is_nullable then
136                     'NOT NULL'
137             else
138                 ''
139             end
140         ) as type,
141         string_agg(
142             ('Constr: ' || constraint_name || ' ' || check_clause), e'\n'
143         ) as constr,
144         ('Comment: ' || comment) as comment
145     from table_column(table_schema, table_name)
146     group by number, name, type, comment, is_nullable
147 ) as qqq
148 order by number;
149 $$ language sql;
150
151 create or replace function print_table_info(
152     table_schema text,
153     table_name text
154 ) returns void as $$
155 declare
156     col record;
157     col_contr record;
158
159     C1W integer;

```

```

160 C2W integer;
161 C31W integer;
162 C32W integer;
163 REM integer;
164 begin
165 C1W := 2;
166 C2W := 12;
167 C31W := 8;
168 C32W := 64 + 8;
169 REM := 11;
170
171 if not exists (
172     select *
173     from table_column(table_schema, table_name)
174     limit 1
175 ) then
176     raise exception '%', ('Table " ' || table_schema || '.' || table_name || '" was not
177     found!');
178 end if;
179
180 raise info
181 '%',
182 rpad(
183     '|--- Table " ' || table_schema || '.' || table_name || '" Information ',
184     C1W + C2W + C31W + C32W + REM,
185     '-') || '|';
186
187 raise info
188 '| % | % | % | % |',
189 rpad('N', C1W, ' '),
190 rpad('Name', C2W, ' '),
191 rpad('Attributes', C31W + C32W + 2, ' ');
192
193 raise info '%', rpad('|', C1W + C2W + C31W + C32W + REM, '-') || '|';
194
195 for col in
196     select distinct number, name, type, comment
197     from table_column(table_schema, table_name)
198     order by number
199 loop
200     raise info
201     '| % | % | % | % |',
202     rpad(cast(col.number as text), C1W, ' '),
203     rpad(col.name, C2W, ' '),
204     (rpad('Type', C31W, ' ') || ': ' || rpad(col.type, C32W, ' '));
205
206 for col_contr in
207     select
208         constraint_name as name,
209         check_clause as clause
210     from table_column(table_schema, table_name)
211     where (
212         table_column.number = col.number and
213         constraint_name is not null
214     )
215 loop
216     raise info
217     '| % | % | % | % |',
218     rpad('', C1W, ' '),
219     rpad('', C2W, ' '),
220     (
221         rpad('Constr', C31W, ' ') || ': ' || rpad(
222             (col_contr.name || ' ' || col_contr.clause), C32W, ' '
223         )
224     );
225 end loop;
226
227 if col.comment is not null then
228     raise info
229     '| % | % | % | % |',
230     rpad('', C1W, ' '),
231     rpad('', C2W, ' '),
232     (rpad('Comment', C31W, ' ') || ': ' || rpad(col.comment, C32W, ' '));
233 end if;
234
235 raise info '%', rpad('|', C1W + C2W + C31W + C32W + REM, '-') || '|';
236 end loop;
237 end;

```

```

232 $$ language plpgsql;
233
234 select print_table_info('public', 'person');

```

## 5 Вывод

```

1 psql:get_table_info.sql:226: INFO: |--- Table "public.person" Information
2 psql:get_table_info.sql:226: INFO: | N | Name | Attributes
3 psql:get_table_info.sql:226: INFO: |-----|
4 psql:get_table_info.sql:226: INFO: | 1 | id | Type : numeric (9, 2)
5 psql:get_table_info.sql:226: INFO: | | | Comment : The unique number of
6 psql:get_table_info.sql:226: INFO: |-----|
7 psql:get_table_info.sql:226: INFO: | 2 | last_name | Type : character varying
8 psql:get_table_info.sql:226: INFO: | | | Comment : Last name of the
9 psql:get_table_info.sql:226: INFO: |-----|
10 psql:get_table_info.sql:226: INFO: | 3 | first_name | Type : character varying
11 psql:get_table_info.sql:226: INFO: | | | Comment : The name of the
12 psql:get_table_info.sql:226: INFO: |-----|
13 psql:get_table_info.sql:226: INFO: | 4 | patronymic | Type : character varying
14 psql:get_table_info.sql:226: INFO: | | | Comment : The patronymic of
15 psql:get_table_info.sql:226: INFO: |-----|
16 psql:get_table_info.sql:226: INFO: | 5 | birth_date | Type : date
17 psql:get_table_info.sql:226: INFO: | | | Comment : Date of birth of a
18 psql:get_table_info.sql:226: INFO: |-----|
19 psql:get_table_info.sql:226: INFO: | 6 | gender | Type : character (1)
20 psql:get_table_info.sql:226: INFO: | | | Constr : person_gender_check
21 psql:get_table_info.sql:226: INFO: | | | Constr : person_gender_check1
22 psql:get_table_info.sql:226: INFO: |-----|
23 psql:get_table_info.sql:226: INFO: | 7 | foreigner | Type : character varying
24 psql:get_table_info.sql:226: INFO: |-----|
25 psql:get_table_info.sql:226: INFO: | 8 | created_who | Type : character varying
26 psql:get_table_info.sql:226: INFO: |-----|
27 psql:get_table_info.sql:226: INFO: | 9 | created_when | Type : date
28 psql:get_table_info.sql:226: INFO: |-----|

```

```

29 psql:get_table_info.sql:226: INFO:  | 10 | edited_who | Type      : character varying
    (40)
30 psql:get_table_info.sql:226: INFO:
    |-----|
31 psql:get_table_info.sql:226: INFO:  | 11 | edited_when | Type      : date
    |
32 psql:get_table_info.sql:226: INFO:
    |-----|
33 psql:get_table_info.sql:226: INFO:  | 12 | death_date | Type      : date
    |
34 psql:get_table_info.sql:226: INFO:  |    |              | Comment   : Date of death of a
    person
35 psql:get_table_info.sql:226: INFO:
    |-----|
36 psql:get_table_info.sql:226: INFO:  | 13 | pin         | Type      : character varying
    (20)
37 psql:get_table_info.sql:226: INFO:
    |-----|
38 psql:get_table_info.sql:226: INFO:  | 14 | inn         | Type      : character varying
    (20)
39 psql:get_table_info.sql:226: INFO:
    |-----|

```

## 6 Вывод

Данная лабораторная работа помогла мне изучить системный каталог PostgreSQL.

## Список литературы