

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 Реализация скрипта	2
4 Таблица	9
5 Вывод	11

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    exclude (inn WITH =)
29 );
30
31 drop table if exists item;
32 create table item (
33     id1 integer,
34     id2 integer,
35
36     id11 integer,
37     id12 integer,
38
39     primary key (id1, id2),
40     foreign key (id11, id12) references item(id1, id2)
41 );
42
43 comment on column person.id is 'The unique number of the person';
44 comment on column person.id is 'The unique number of the person';
45 comment on column person.last_name is 'Last name of the person';
46 comment on column person.first_name is 'The name of the person';
47 comment on column person.patronymic is 'The patronymic of the person';
48 comment on column person.birth_date is 'Date of birth of a person';
49 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.attname        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_comment CASCADE;
27 CREATE VIEW meta_comment AS
28     SELECT
29         pg_description.objoid      AS owner_id,
30         pg_description.objsubid    AS child_id,
31         pg_description.description AS content
32     FROM pg_description;
33
34 DROP VIEW IF EXISTS meta_type CASCADE;
35 CREATE VIEW meta_type AS
36     SELECT
37         pg_type.oid      AS id,
38         pg_type.typname AS name
39     FROM pg_type;
40
41 DROP VIEW IF EXISTS meta_operator CASCADE;
42 CREATE VIEW meta_operator AS
43     SELECT
44         pg_operator.oid      AS id,
45         pg_operator.oprname  AS name
46     FROM pg_operator;
47
48 DROP VIEW IF EXISTS meta_constraint_check CASCADE;
49 CREATE VIEW meta_constraint_check AS
50     SELECT
51         pg_constraint.oid          AS id,
52         pg_constraint.conname      AS name,
53         pg_constraint.connamespace AS namespace_id,
54         pg_constraint.conrelid     AS constrained_table_id,
55         pg_constraint.conkey       AS
56         constrained_column_numbers,
57         pg_get_expr(pg_constraint.conbin, COALESCE(pg_class.oid, 0)) AS clause
58     FROM pg_constraint
59     LEFT JOIN pg_class ON pg_class.oid = pg_constraint.conrelid
60     WHERE pg_constraint.contype = 'c';
61
62 DROP VIEW IF EXISTS meta_constraint_foreign_key CASCADE;
63 CREATE VIEW meta_constraint_foreign_key AS
64     SELECT
65         pg_constraint.oid      AS id,
66         pg_constraint.conname  AS name,
67         pg_constraint.connamespace AS namespace_id,
68         pg_constraint.conrelid  AS constrained_table_id,
69         pg_constraint.conkey    AS constrained_column_numbers,
70         pg_constraint.confrelid AS referenced_table_id,
71         pg_constraint.confkey   AS referenced_column_numbers
72     FROM pg_constraint
73     WHERE pg_constraint.contype = 'f';
74
75 DROP VIEW IF EXISTS meta_constraint_primary_key CASCADE;
76 CREATE VIEW meta_constraint_primary_key AS
77     SELECT
78         pg_constraint.oid      AS id,
79         pg_constraint.conname  AS name,
80         pg_constraint.connamespace AS namespace_id,
81         pg_constraint.conrelid  AS constrained_table_id,
82         pg_constraint.conkey    AS constrained_column_numbers
83     FROM pg_constraint
84     WHERE pg_constraint.contype = 'p';
85
86 DROP VIEW IF EXISTS meta_constraint_unique CASCADE;
87 CREATE VIEW meta_constraint_unique AS
88     SELECT
89         pg_constraint.oid      AS id,
90         pg_constraint.conname  AS name,
91         pg_constraint.connamespace AS namespace_id,

```

```

91     pg_constraint.conrelid      AS constrained_table_id,
92     pg_constraint.conkey        AS constrained_column_numbers
93 FROM pg_constraint
94 WHERE pg_constraint.contype = 'u';
95
96 DROP VIEW IF EXISTS meta_constraint_exclusion CASCADE;
97 CREATE VIEW meta_constraint_exclusion AS
98 SELECT
99     pg_constraint.oid            AS id,
100    pg_constraint.conname        AS name,
101    pg_constraint.connamespace   AS namespace_id,
102    pg_constraint.conrelid       AS constrained_table_id,
103    pg_constraint.conkey         AS constrained_column_numbers,
104    pg_constraint.conexclp       AS per_column_operator_ids
105 FROM pg_constraint
106 WHERE pg_constraint.contype = 'x';

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
        constrained_column_number,
20     meta_display_constraint_check.clause        AS clause
21 FROM meta_display_constraint_check
22 WHERE cardinality(meta_display_constraint_check.constrained_column_numbers) = 1;
23
24 DROP VIEW IF EXISTS meta_display_constraint_check_multiple CASCADE;
25 CREATE VIEW meta_display_constraint_check_multiple AS
26 SELECT
27     meta_display_constraint_check.id            AS id,
28     meta_display_constraint_check.name          AS name,
29     meta_display_constraint_check.namespace_id  AS namespace_id,
30     meta_display_constraint_check.constrained_table_id AS constrained_table_id,
31     meta_display_constraint_check.constrained_column_numbers AS
        constrained_column_numbers,
32     meta_display_constraint_check.clause        AS clause
33 FROM meta_display_constraint_check
34 WHERE cardinality(meta_display_constraint_check.constrained_column_numbers) != 1;
35
36 DROP VIEW IF EXISTS meta_display_constraint_foreign_key_single CASCADE;
37 CREATE VIEW meta_display_constraint_foreign_key_single AS
38 SELECT
39     meta_constraint_foreign_key.id            AS id,
40     meta_constraint_foreign_key.name          AS name,
41     meta_constraint_foreign_key.namespace_id  AS namespace_id,
42     meta_constraint_foreign_key.constrained_table_id AS constrained_table_id,
43     meta_constraint_foreign_key.constrained_column_numbers [1] AS
        constrained_column_number,
44     ('REFERENCES ' || meta_table_column.name::text) AS clause
45 FROM meta_constraint_foreign_key
46 JOIN meta_table ON meta_table.id = meta_constraint_foreign_key.
        referenced_table_id
47 JOIN meta_table_column ON (
48     meta_table_column.table_id = meta_table.id AND
49     meta_table_column.number = meta_constraint_foreign_key.referenced_column_numbers [1]
50 )
51 WHERE (
52     cardinality(meta_constraint_foreign_key.constrained_column_numbers) = 1 AND

```

```

53     cardinality(meta_constraint_foreign_key.referenced_column_numbers) = 1
54 );
55
56 DROP FUNCTION IF EXISTS meta_display_column_name CASCADE;
57 CREATE FUNCTION meta_display_column_name(
58     table_id        oid,
59     column_number integer
60 ) RETURNS text AS $$
61 DECLARE
62     column_name text;
63 BEGIN
64     SELECT meta_table_column.name INTO column_name
65     FROM meta_table
66     JOIN meta_table_column ON meta_table_column.table_id = meta_table.id
67     WHERE meta_table.id = meta_display_column_name.table_id
68     AND meta_table_column.number = meta_display_column_name.column_number;
69
70     RETURN column_name;
71 END;
72 $$ 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             (
87                 SELECT string_agg(meta_display_column_name(constrained_table_id,
88                     constrained_column_number), ', ')
89                 FROM unnest(meta_constraint_foreign_key.constrained_column_numbers)
90                 AS constrained_column_number
91             ) || ' REFERENCES ' || (
92                 SELECT string_agg(meta_display_column_name(referenced_table_id,
93                     referenced_column_number), ', ')
94                 FROM unnest(meta_constraint_foreign_key.referenced_column_numbers)
95                 AS referenced_column_number
96             )
97         ) AS clause
98     FROM meta_constraint_foreign_key
99     WHERE (
100         cardinality(meta_constraint_foreign_key.constrained_column_numbers) != 1 AND
101         cardinality(meta_constraint_foreign_key.referenced_column_numbers) != 1
102     );
103
104 DROP VIEW IF EXISTS meta_display_constraint_primary_key_single CASCADE;
105 CREATE VIEW meta_display_constraint_primary_key_single AS
106     SELECT
107         meta_constraint_primary_key.id AS id,
108         meta_constraint_primary_key.name AS name,
109         meta_constraint_primary_key.namespace_id AS namespace_id,
110         meta_constraint_primary_key.constrained_table_id AS constrained_table_id,
111         meta_constraint_primary_key.constrained_column_numbers[1] AS
112         constrained_column_number,
113         'PRIMARY KEY' AS clause
114     FROM meta_constraint_primary_key
115     WHERE cardinality(meta_constraint_primary_key.constrained_column_numbers) = 1;
116
117 DROP VIEW IF EXISTS meta_display_constraint_primary_key_multiple CASCADE;
118 CREATE VIEW meta_display_constraint_primary_key_multiple AS
119     SELECT
120         meta_constraint_primary_key.id AS id,
121         meta_constraint_primary_key.name AS name,
122         meta_constraint_primary_key.namespace_id AS namespace_id,
123         meta_constraint_primary_key.constrained_table_id AS constrained_table_id,
124         meta_constraint_primary_key.constrained_column_numbers AS
125         constrained_column_numbers,

```

```

121 (
122     'PRIMARY KEY ' || (
123         SELECT string_agg(meta_display_column_name(constrained_table_id,
124             constrained_column_number), ', ')
125         FROM unnest(meta_constraint_primary_key.constrained_column_numbers)
126         AS constrained_column_number
127     )
128     ) AS clause
129 FROM meta_constraint_primary_key
130 WHERE cardinality(meta_constraint_primary_key.constrained_column_numbers) != 1;
131 DROP VIEW IF EXISTS meta_display_constraint_unique_single CASCADE;
132 CREATE VIEW meta_display_constraint_unique_single AS
133 SELECT
134     meta_constraint_unique.id AS id,
135     meta_constraint_unique.name AS name,
136     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_exclusion CASCADE;
163 CREATE VIEW meta_display_constraint_exclusion_multiple AS
164 SELECT
165     meta_constraint_exclusion.id AS id,
166     meta_constraint_exclusion.name AS name,
167     meta_constraint_exclusion.namespace_id AS namespace_id,
168     meta_constraint_exclusion.constrained_table_id AS constrained_table_id,
169     meta_constraint_exclusion.constrained_column_numbers AS constrained_column_numbers,
170     (
171         'EXCLUDE ' || (
172             SELECT
173                 string_agg((
174                     meta_display_column_name(constrained_table_id, column_number)
175                     || ' WITH ' || meta_operator.name
176                 ), ', ')
177             FROM unnest(
178                 meta_constraint_exclusion.constrained_column_numbers,
179                 meta_constraint_exclusion.per_column_operator_ids
180             ) WITH ORDINALITY AS column_operator(column_number, operator_id)
181             JOIN meta_operator ON meta_operator.id = column_operator.operator_id
182         )
183     ) AS clause
184 FROM meta_constraint_exclusion;
185
186 DROP VIEW IF EXISTS meta_display_constraint_single CASCADE;
187 CREATE VIEW meta_display_constraint_single AS
188 (
189     SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
190     clause
191     FROM meta_display_constraint_check_single
192 ) UNION ALL (

```

```

191     SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
192           clause
193   ) UNION ALL (
194     SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
195           clause
196   FROM meta_display_constraint_primary_key_single
197   ) UNION ALL (
198     SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
199           clause
200   FROM meta_display_constraint_unique_single
201   );
202
203 DROP VIEW IF EXISTS meta_display_constraint_multiple CASCADE;
204 CREATE VIEW meta_display_constraint_multiple AS
205 (
206   SELECT id, name, namespace_id, constrained_table_id, clause
207   FROM meta_display_constraint_check_multiple
208   ) UNION ALL (
209     SELECT id, name, namespace_id, constrained_table_id, clause
210   FROM meta_display_constraint_foreign_key_multiple
211   ) UNION ALL (
212     SELECT id, name, namespace_id, constrained_table_id, clause
213   FROM meta_display_constraint_primary_key_multiple
214   ) UNION ALL (
215     SELECT id, name, namespace_id, constrained_table_id, clause
216   FROM meta_display_constraint_unique_multiple
217   ) UNION ALL (
218     SELECT id, name, namespace_id, constrained_table_id, clause
219   FROM meta_display_constraint_exclusion_multiple
220   );
221
222
223 1 DROP VIEW IF EXISTS main_table_column_constraint CASCADE;
224 2 CREATE VIEW main_table_column_constraint AS
225 3   SELECT
226 4     meta_namespace.name           AS schema_name,
227 5     meta_table.name              AS table_name,
228 6     meta_table_column.name       AS column_name,
229 7     meta_display_constraint_single.name AS constraint_name,
230 8     meta_display_constraint_single.clause AS constraint_clause
231 9   FROM meta_table
23210  JOIN meta_namespace ON meta_table.namespace_id = meta_namespace.id
23311  JOIN meta_table_column
23412    ON meta_table_column.table_id = meta_table.id
23513  LEFT JOIN meta_display_constraint_single ON (
23614    meta_display_constraint_single.constrained_table_id = meta_table.id AND
23715    meta_display_constraint_single.constrained_column_number = meta_table_column.number
23816  );
239
24017
24118 DROP VIEW IF EXISTS main_table_constraint CASCADE;
24219 CREATE VIEW main_table_constraint AS
24320   SELECT
24421     meta_namespace.name           AS schema_name,
24522     meta_table.name              AS table_name,
24623     meta_display_constraint_multiple.name AS constraint_name,
24724     meta_display_constraint_multiple.clause AS constraint_clause
24825  FROM meta_table
24926  JOIN meta_namespace ON meta_table.namespace_id = meta_namespace.id
25027  LEFT JOIN meta_display_constraint_multiple ON (
25128    meta_display_constraint_multiple.constrained_table_id = meta_table.id
25229  );
253
25430
25531 DROP PROCEDURE IF EXISTS main_table_print_pretty;
25632 CREATE PROCEDURE main_table_print_pretty (
25733   table_schema text,
25834   table_name text
25935 ) AS $$
26036 DECLARE
26137   col record;
26238   col_constr record;
26339
26440   C1W integer;
26541   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.number AS column_number,
76 meta_table_column.name AS column_name,
77 meta_type.name AS type_name,
78 meta_table_column.is_nullable AS is_nullable,
79 meta_table.id AS table_id
80 FROM meta_table
81 JOIN meta_namespace ON meta_namespace.id = meta_table.namespace_id
82 JOIN meta_table_column ON meta_table.id = meta_table_column.table_id
83 JOIN meta_type ON meta_type.id = meta_table_column.type_id
84 WHERE meta_namespace.name = main_table_print_pretty.table_schema
85 AND meta_table.name = main_table_print_pretty.table_name
86 AND meta_table_column.number > 0
87 LOOP
88 RAISE INFO
89 '| % | % | % |',
90 rpad(col.column_number::text, C1W, ' '),
91 rpad(col.column_name, C2W, ' '),
92 (rpad('Type', C31W, ' ') || ': ' || rpad(col.type_name, C32W, ' '));
93 RAISE INFO
94 '| % | % | % |',
95 rpad('', C1W, ' '),
96 rpad('', C2W, ' '),
97 rpad('Null', C31W, ' ') || ': ' || rpad((
98 CASE WHEN col.is_nullable THEN 'NULLABLE' ELSE 'NOT NULL' END
99 ), C32W, ' ');
100
101 FOR col_constr IN
102 SELECT *
103 FROM meta_comment
104 WHERE meta_comment.owner_id = col.table_id
105 AND meta_comment.child_id = col.column_number
106 LOOP
107 IF NOT col_constr IS NULL THEN
108 RAISE INFO
109 '| % | % | % |',
110 rpad('', C1W, ' '),
111 rpad('', C2W, ' '),
112 rpad('Comment', C31W, ' ') || ': ' || rpad(
113 col_constr.content, C32W, ' ');
114 END IF;

```

```

115 END LOOP;
116 FOR col_constr IN
117     SELECT
118         constraint_name AS name,
119         constraint_clause AS clause
120     FROM main_table_column_constraint
121     WHERE
122         main_table_column_constraint.schema_name = main_table_print_pretty.table_schema
123     AND
124         main_table_column_constraint.table_name = main_table_print_pretty.table_name AND
125         main_table_column_constraint.column_name = col.column_name
126 LOOP
127     IF NOT col_constr.name IS NULL THEN
128         RAISE INFO
129             ' | % | % | % | ',
130             rpad('', C1W, ' '),
131             rpad('', C2W, ' '),
132             (
133                 rpad('Constr', C31W, ' ') || ': ' || rpad(
134                     (col_constr.name || ' ' || col_constr.clause), C32W, ' '
135                 )
136             );
137     END IF;
138 END LOOP;
139
140 FOR col IN
141     SELECT
142         main_table_constraint.constraint_name AS constraint_name,
143         main_table_constraint.constraint_clause AS constraint_clause
144     FROM main_table_constraint
145     WHERE
146         main_table_constraint.schema_name = main_table_print_pretty.table_schema AND
147         main_table_constraint.table_name = main_table_print_pretty.table_name
148 LOOP
149     RAISE INFO
150         ' | % | ',
151         (rpad('Constr', C31W, ' ') || ': ' ||
152         (col.constraint_name || ' ' || col.constraint_clause))
153     ;
154 END LOOP;
155 END;
156 $$ language plpgsql;
157
158 drop procedure IF EXISTS solution;
159 create or replace procedure solution(
160     table_name text
161 ) as $$
162 declare
163     table_schema text;
164 begin
165     select information_schema.tables.table_schema into table_schema
166     from information_schema.tables
167     where information_schema.tables.table_name = solution.table_name
168     limit 1;
169
170     call main_table_print_pretty(table_schema, table_name);
171 end;
172 $$ language plpgsql;
173
174 call solution('person');
```

4 Таблица

```

1 |--- Table "public.person" Information
  |-----|
2 | N | Name          | Attributes
  |-----|
3 |---|
4 | 1 | id            | Type      : numeric
  |-----|
```

5			Null	: NOT NULL
6			Comment	: The unique number of the person
7			Constr	: person_pkey PRIMARY KEY
8	2	last_name	Type	: varchar
9			Null	: NOT NULL
10			Comment	: Last name of the person
11	3	first_name	Type	: varchar
12			Null	: NOT NULL
13			Comment	: The name of the person
14	4	patronymic	Type	: varchar
15			Null	: NULLABLE
16			Comment	: The patronymic of the person
17	5	birth_date	Type	: date
18			Null	: NOT NULL
19			Comment	: Date of birth of a person
20	6	gender	Type	: bpchar
21			Null	: NOT NULL
22			Constr	: person_gender_check (gender = ANY (ARRAY['M'::bpchar, 'F'::bpchar]))
23			Constr	: person_gender_check1 (gender = ANY (ARRAY['M'::bpchar, 'F'::bpchar]))
24	7	foreigner	Type	: varchar
25			Null	: NOT NULL
26	8	created_who	Type	: varchar
27			Null	: NOT NULL
28	9	created_when	Type	: date
29			Null	: NOT NULL
30	10	edited_who	Type	: varchar
31			Null	: NOT NULL
32	11	edited_when	Type	: date
33			Null	: NOT NULL
34	12	death_date	Type	: date
35			Null	: NULLABLE
36			Comment	: Date of death of a person
37	13	pin	Type	: varchar
38			Null	: NULLABLE
39			Constr	: person_pin_key UNIQUE
40	14	inn	Type	: varchar
41			Null	: NULLABLE

```

42 |          |          | Constr : person_inn_key UNIQUE
    |          |          |
43 | Constr : person_check ((length((patronymic)::text) > 10) AND (length((last_name)::
    |          |          | text) > 10) AND (length((first_name)::text) > 10)) |
44 | Constr : person_last_name_first_name_patronymic_key UNIQUE last_name, first_name,
    |          |          | patronymic |
45 | Constr : person_inn_excl EXCLUDE inn WITH = |

```

5 Вывод

Данная лабораторная работа помогла мне изучить системный каталог PostgreSQL.

Список литературы