

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	10

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     NULLIF(pg_attribute.atttypmod, -1) AS type_data,
24     (NOT pg_attribute.attnotnull) AS is_nullable
25 FROM pg_attribute;
26
27 DROP VIEW IF EXISTS meta_comment CASCADE;
28 CREATE VIEW meta_comment AS
29 SELECT
30     pg_description.objoid        AS owner_id,
31     pg_description.objsubid      AS child_id,
32     pg_description.description   AS content
33 FROM pg_description;
34
35 DROP VIEW IF EXISTS meta_type CASCADE;
36 CREATE VIEW meta_type AS
37 SELECT
38     pg_type.oid        AS id,
39     pg_type.typname    AS name
40 FROM pg_type;
41
42 DROP VIEW IF EXISTS meta_operator CASCADE;
43 CREATE VIEW meta_operator AS
44 SELECT
45     pg_operator.oid        AS id,
46     pg_operator.oprname    AS name
47 FROM pg_operator;
48
49 DROP VIEW IF EXISTS meta_constraint_check CASCADE;
50 CREATE VIEW meta_constraint_check AS
51 SELECT
52     pg_constraint.oid        AS id,
53     pg_constraint.conname    AS name,
54     pg_constraint.connamespace AS namespace_id,
55     pg_constraint.conrelid    AS constrained_table_id,
56     ,
57     pg_constraint.conkey      AS
58     constrained_column_numbers,
59     pg_get_expr(pg_constraint.conbin, COALESCE(pg_class.oid, 0)) AS clause
60 FROM pg_constraint
61 LEFT JOIN pg_class ON pg_class.oid = pg_constraint.conrelid
62 WHERE pg_constraint.contype = 'c';
63
64 DROP VIEW IF EXISTS meta_constraint_foreign_key CASCADE;
65 CREATE VIEW meta_constraint_foreign_key AS
66 SELECT
67     pg_constraint.oid        AS id,
68     pg_constraint.conname    AS name,
69     pg_constraint.connamespace AS namespace_id,
70     pg_constraint.conrelid    AS constrained_table_id,
71     pg_constraint.conkey      AS constrained_column_numbers,
72     pg_constraint.confrelid    AS referenced_table_id,
73     pg_constraint.confkey      AS referenced_column_numbers
74 FROM pg_constraint
75 WHERE pg_constraint.contype = 'f';
76
77 DROP VIEW IF EXISTS meta_constraint_primary_key CASCADE;
78 CREATE VIEW meta_constraint_primary_key AS
79 SELECT
80     pg_constraint.oid        AS id,
81     pg_constraint.conname    AS name,
82     pg_constraint.connamespace AS namespace_id,
83     pg_constraint.conrelid    AS constrained_table_id,
84     pg_constraint.conkey      AS constrained_column_numbers
85 FROM pg_constraint
86 WHERE pg_constraint.contype = 'p';
87
88 DROP VIEW IF EXISTS meta_constraint_unique CASCADE;
89 CREATE VIEW meta_constraint_unique AS
90 SELECT
91     pg_constraint.oid        AS id,
92     pg_constraint.conname    AS name,

```

```

91     pg_constraint.connamespace AS namespace_id,
92     pg_constraint.conrelid     AS constrained_table_id,
93     pg_constraint.conkey       AS constrained_column_numbers
94 FROM pg_constraint
95 WHERE pg_constraint.contype = 'u';
96
97 DROP VIEW IF EXISTS meta_constraint_exclusion CASCADE;
98 CREATE VIEW meta_constraint_exclusion AS
99 SELECT
100     pg_constraint.oid           AS id,
101     pg_constraint.conname       AS name,
102     pg_constraint.connamespace AS namespace_id,
103     pg_constraint.conrelid      AS constrained_table_id,
104     pg_constraint.conkey        AS constrained_column_numbers,
105     pg_constraint.conexclp      AS per_column_operator_ids
106 FROM pg_constraint
107 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
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 (

```

```

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

```

```

121     constrained_column_numbers,
122     (
123         'PRIMARY KEY ' || (
124             SELECT string_agg(meta_display_column_name(constrained_table_id,
125                 constrained_column_number), ', ')
126             FROM unnest(meta_constraint_primary_key.constrained_column_numbers)
127             AS constrained_column_number
128         )
129         AS clause
130     )
131 FROM meta_constraint_primary_key
132 WHERE cardinality(meta_constraint_primary_key.constrained_column_numbers) != 1;
133
134 DROP VIEW IF EXISTS meta_display_constraint_unique_single CASCADE;
135 CREATE VIEW meta_display_constraint_unique_single AS
136 SELECT
137     meta_constraint_unique.id AS id,
138     meta_constraint_unique.name AS name,
139     meta_constraint_unique.namespace_id AS namespace_id,
140     meta_constraint_unique.constrained_table_id AS constrained_table_id,
141     meta_constraint_unique.constrained_column_numbers[1] AS constrained_column_number,
142     'UNIQUE' AS clause
143 FROM meta_constraint_unique
144 WHERE cardinality(meta_constraint_unique.constrained_column_numbers) = 1;
145
146 DROP VIEW IF EXISTS meta_display_constraint_unique_multiple CASCADE;
147 CREATE VIEW meta_display_constraint_unique_multiple AS
148 SELECT
149     meta_constraint_unique.id AS id,
150     meta_constraint_unique.name AS name,
151     meta_constraint_unique.namespace_id AS namespace_id,
152     meta_constraint_unique.constrained_table_id AS constrained_table_id,
153     meta_constraint_unique.constrained_column_numbers AS constrained_column_numbers,
154     (
155         'UNIQUE ' || (
156             SELECT string_agg(meta_display_column_name(constrained_table_id,
157                 constrained_column_number), ', ')
158             FROM unnest(meta_constraint_unique.constrained_column_numbers)
159             AS constrained_column_number
160         )
161         AS clause
162     )
163 FROM meta_constraint_unique
164 WHERE cardinality(meta_constraint_unique.constrained_column_numbers) != 1;
165
166 DROP VIEW IF EXISTS meta_display_constraint_exclusion CASCADE;
167 CREATE VIEW meta_display_constraint_exclusion_multiple AS
168 SELECT
169     meta_constraint_exclusion.id AS id,
170     meta_constraint_exclusion.name AS name,
171     meta_constraint_exclusion.namespace_id AS namespace_id,
172     meta_constraint_exclusion.constrained_table_id AS constrained_table_id,
173     meta_constraint_exclusion.constrained_column_numbers AS constrained_column_numbers,
174     (
175         'EXCLUDE ' || (
176             SELECT
177                 string_agg((
178                     '(',
179                     || meta_display_column_name(constrained_table_id, column_number)
180                     || ' WITH '
181                     || meta_operator.name ||
182                     ')',
183                     ', ')
184             FROM unnest(
185                 meta_constraint_exclusion.constrained_column_numbers,
186                 meta_constraint_exclusion.per_column_operator_ids
187             ) WITH ORDINALITY AS column_operator(column_number, operator_id)
188             JOIN meta_operator ON meta_operator.id = column_operator.operator_id
189         )
190         AS clause
191     )
192 FROM meta_constraint_exclusion;
193
194 DROP VIEW IF EXISTS meta_display_constraint_single CASCADE;
195 CREATE VIEW meta_display_constraint_single AS
196 (

```

```

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_foreign_key_single
197   ) UNION ALL (
198     SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
199           clause
200   FROM meta_display_constraint_primary_key_single
201   ) UNION ALL (
202     SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
203           clause
204   FROM meta_display_constraint_unique_single
205   );
206
207 DROP VIEW IF EXISTS meta_display_constraint_multiple CASCADE;
208 CREATE VIEW meta_display_constraint_multiple AS
209 (
210     SELECT id, name, namespace_id, constrained_table_id, clause
211   FROM meta_display_constraint_check_multiple
212   ) UNION ALL (
213     SELECT id, name, namespace_id, constrained_table_id, clause
214   FROM meta_display_constraint_foreign_key_multiple
215   ) UNION ALL (
216     SELECT id, name, namespace_id, constrained_table_id, clause
217   FROM meta_display_constraint_primary_key_multiple
218   ) UNION ALL (
219     SELECT id, name, namespace_id, constrained_table_id, clause
220   FROM meta_display_constraint_unique_multiple
221   );
222
223
224 1 DROP VIEW IF EXISTS main_table_column_constraint CASCADE;
225 2 CREATE VIEW main_table_column_constraint AS
226 3     SELECT
227 4         meta_namespace.name                AS schema_name,
228 5         meta_table.name                    AS table_name,
229 6         meta_table_column.name             AS column_name,
230 7         meta_display_constraint_single.name AS constraint_name,
231 8         meta_display_constraint_single.clause AS constraint_clause
232 9     FROM meta_table
23310     JOIN meta_namespace ON meta_table.namespace_id = meta_namespace.id
23411     JOIN meta_table_column
23512         ON meta_table_column.table_id = meta_table.id
23613     LEFT JOIN meta_display_constraint_single ON (
23714         meta_display_constraint_single.constrained_table_id = meta_table.id AND
23815         meta_display_constraint_single.constrained_column_number = meta_table_column.number
23916     );
240
241 17
242 18 DROP VIEW IF EXISTS main_table_constraint CASCADE;
243 19 CREATE VIEW main_table_constraint AS
244 20     SELECT
245 21         meta_namespace.name                AS schema_name,
246 22         meta_table.name                    AS table_name,
247 23         meta_display_constraint_multiple.name AS constraint_name,
248 24         meta_display_constraint_multiple.clause AS constraint_clause
249 25     FROM meta_table
25026     JOIN meta_namespace ON meta_table.namespace_id = meta_namespace.id
25127     LEFT JOIN meta_display_constraint_multiple ON (
25228         meta_display_constraint_multiple.constrained_table_id = meta_table.id
25329     );
254
255 30
256 31 DROP PROCEDURE IF EXISTS main_table_print_pretty;
257 32 CREATE PROCEDURE main_table_print_pretty (
258 33     table_schema text,
259 34     table_name text
260 35 ) AS $$
261 36 DECLARE
262 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.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_column.type_data AS type_data,
80 meta_table.id AS table_id
81 FROM meta_table
82 JOIN meta_namespace ON meta_namespace.id = meta_table.namespace_id
83 JOIN meta_table_column ON meta_table.id = meta_table_column.table_id
84 JOIN meta_type ON meta_type.id = meta_table_column.type_id
85 WHERE meta_namespace.name = main_table_print_pretty.table_schema
86 AND meta_table.name = main_table_print_pretty.table_name
87 AND meta_table_column.number > 0
88 LOOP
89 RAISE INFO
90 '| % | % | % |',
91 rpad(col.column_number::text, C1W, ' '),
92 rpad(col.column_name, C2W, ' '),
93 (
94 rpad('Type', C31W, ' ') || ': ' ||
95 rpad(col.type_name || (
96 CASE WHEN col.type_name = 'varchar'
97 THEN '(' || col.type_data - 4 || ')'
98 ELSE '' END
99 ) || ' ' || (
100 CASE WHEN col.is_nullable THEN 'NULLABLE' ELSE 'NOT NULL' END
101 ), C32W, ' ');
102 );
103 FOR col_constr IN
104 SELECT *
105 FROM meta_comment
106 WHERE meta_comment.owner_id = col.table_id
107 AND meta_comment.child_id = col.column_number
108 LOOP
109 IF NOT col_constr IS NULL THEN
110 RAISE INFO

```

```

111         '| % | % | % |',
112         rpad('', C1W, ' '),
113         rpad('', C2W, ' '),
114         rpad('Comment', C31W, ' ') || ': ' || rpad(
115             col_constr.content, C32W, ' ');
116     END IF;
117 END LOOP;
118 FOR col_constr IN
119     SELECT
120         constraint_name AS name,
121         constraint_clause AS clause
122     FROM main_table_column_constraint
123     WHERE
124         main_table_column_constraint.schema_name = main_table_print_pretty.table_schema
125     AND
126         main_table_column_constraint.table_name = main_table_print_pretty.table_name AND
127         main_table_column_constraint.column_name = col.column_name
128 LOOP
129     IF NOT col_constr.name IS NULL THEN
130         RAISE INFO
131             '| % | % | % |',
132             rpad('', C1W, ' '),
133             rpad('', C2W, ' '),
134             (
135                 rpad('Constr', C31W, ' ') || ': ' || rpad(
136                     (col_constr.name || ' ' || col_constr.clause), C32W, ' '
137             );
138     END IF;
139 END LOOP;
140 END LOOP;
141
142 FOR col IN
143     SELECT
144         main_table_constraint.constraint_name AS constraint_name,
145         main_table_constraint.constraint_clause AS constraint_clause
146     FROM main_table_constraint
147     WHERE
148         main_table_constraint.schema_name = main_table_print_pretty.table_schema AND
149         main_table_constraint.table_name = main_table_print_pretty.table_name
150 LOOP
151     RAISE INFO
152         '| % |',
153         (rpad('Constr', C31W, ' ') || ': ' ||
154         (col.constraint_name || ' ' || col.constraint_clause))
155     ;
156 END LOOP;
157 END;
158 $$ language plpgsql;
159
160 drop procedure IF EXISTS solution;
161 create or replace procedure solution(
162     table_name text
163 ) as $$
164 declare
165     table_schema text;
166 begin
167     select information_schema.tables.table_schema into table_schema
168     from information_schema.tables
169     where information_schema.tables.table_name = solution.table_name
170     limit 1;
171
172     call main_table_print_pretty(table_schema, table_name);
173 end;
174 $$ language plpgsql;
175
176 call solution('person');
```

4 Таблица

```

1 |--- Table "public.person" Information
-----|
```

2	N	Name	Attributes
3			
4	1	id	Type : numeric NOT NULL
5			Comment : The unique number of the person
6			Constr : person_pkey PRIMARY KEY
7	2	last_name	Type : varchar(25) NOT NULL
8			Comment : Last name of the person
9	3	first_name	Type : varchar(2000) NOT NULL
10			Comment : The name of the person
11	4	patronymic	Type : varchar(20) NULLABLE
12			Comment : The patronymic of the person
13	5	birth_date	Type : date NOT NULL
14			Comment : Date of birth of a person
15	6	gender	Type : bpchar NOT NULL
16			Constr : person_gender_check (gender = ANY (ARRAY['M'::bpchar, 'F'::bpchar]))
17			Constr : person_gender_check1 (gender = ANY (ARRAY['M'::bpchar, 'F'::bpchar]))
18	7	foreigner	Type : varchar(3) NOT NULL
19	8	created_who	Type : varchar(40) NOT NULL
20	9	created_when	Type : date NOT NULL
21	10	edited_who	Type : varchar(40) NOT NULL
22	11	edited_when	Type : date NOT NULL
23	12	death_date	Type : date NULLABLE
24			Comment : Date of death of a person
25	13	pin	Type : varchar(20) NULLABLE
26			Constr : person_pin_key UNIQUE
27	14	inn	Type : varchar(20) NULLABLE
28			Constr : person_inn_key UNIQUE
29			Constr : person_check ((length((patronymic)::text) > 10) AND (length((last_name)::text) > 10) AND (length((first_name)::text) > 10))
30			Constr : person_last_name_first_name_patronymic_key UNIQUE last_name, first_name, patronymic
31			Constr : person_inn_excl EXCLUDE (inn WITH =)

5 Вывод

Данная лабораторная работа помогла мне изучить системный каталог PostgreSQL.

Список литературы