Университет ИТМО

Факультет программной инженерии и компьютерной техники

Распределённые системы хранения данных. Лабораторная работа №1.

Группа:	P33131
Студент:	Смирнов Виктор Игоревич
Преподаватель:	Афанасьев Дмитрий Борисович
Вариант:	776

2024

Ключевые слова

База данных, PostgreSQL, системный каталог.

Содержание

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
No.  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.

```sql
drop table person;
create table person (
  id numeric(9, 2) primary key,
  last_name varchar(25) not null,
  first_name varchar(2000) not null,
  patronymic varchar(20),
  birth_date date not null,
  gender char(1) not null,
  foreigner varchar(3) not null,
  created_who varchar(40) not null,
  created_when date not null,
  edited_who varchar(40) not null,
  edited_when date not null,
  death_date date,
  pin varchar(20),
  inn varchar(20),

  check (gender in ('M', 'F')),
  check (gender in ('M', 'F')),
  check (
    length(patronymic) > 10 AND
    length(last_name) > 10 AND
    length(first_name) > 10
  ),
  unique (last_name, first_name, patronymic),
  unique (inn),
  unique (pin),
  exclude (inn WITH =)
);

drop table if exists item;
create table item (
  id1 integer,
  id2 integer,

  id11 integer,
  id12 integer,

  primary key (id1, id2),
  foreign key (id11, id12) references item(id1, id2)
);

comment on column person.id is 'The unique number of the person';
comment on column person.id is 'The unique number of the person';
comment on column person.last_name is 'Last name of the person';
comment on column person.first_name is 'The name of the person';
comment on column person.patronymic is 'The patronymic of the person';
comment on column person.birth_date is 'Date of birth of a person';
comment on column person.death_date is 'Date of death of a person';
```

3 Реализация скрипта

```sql
DROP VIEW IF EXISTS meta_namespace CASCADE;
CREATE VIEW meta_namespace AS
  SELECT
    pg_namespace.oid     AS id,
    pg_namespace.nspname AS name
  FROM pg_namespace;

DROP VIEW IF EXISTS meta_table CASCADE;
CREATE VIEW meta_table AS
  SELECT
    pg_class.oid          AS id,
    pg_class.relname      AS name,
    pg_class.relnamespace AS namespace_id
  FROM pg_class;

DROP VIEW IF EXISTS meta_table_column CASCADE;
CREATE VIEW meta_table_column AS
  SELECT
    pg_attribute.attrelid         AS table_id,
    pg_attribute.attnum           AS number,
    pg_attribute.attname          AS name,
    pg_attribute.atttypid         AS type_id,
    (NOT pg_attribute.attnotnull) AS is_nullable
  FROM pg_attribute;

DROP VIEW IF EXISTS meta_comment CASCADE;
CREATE VIEW meta_comment AS
  SELECT
    pg_description.objoid       AS owner_id,
    pg_description.objsubid     AS child_id,
    pg_description.description  AS content
  FROM pg_description;

DROP VIEW IF EXISTS meta_type CASCADE;
CREATE VIEW meta_type AS
  SELECT
    pg_type.oid     AS id,
    pg_type.typname AS name
  FROM pg_type;

DROP VIEW IF EXISTS meta_operator CASCADE;
CREATE VIEW meta_operator AS
  SELECT
    pg_operator.oid       AS id,
    pg_operator.oprname   AS name
  FROM pg_operator;

DROP VIEW IF EXISTS meta_constraint_check CASCADE;
CREATE VIEW meta_constraint_check AS
  SELECT
    pg_constraint.oid                                            AS id,
    pg_constraint.conname                                        AS name,
    pg_constraint.connamespace                                   AS namespace_id,
    pg_constraint.conrelid                                       AS constrained_table_id
    ,
    pg_constraint.conkey                                         AS
    constrained_column_numbers,
    pg_get_expr(pg_constraint.conbin, COALESCE(pg_class.oid, 0)) AS clause
  FROM pg_constraint
  LEFT JOIN pg_class ON pg_class.oid = pg_constraint.conrelid
  WHERE pg_constraint.contype = 'c';

DROP VIEW IF EXISTS meta_constraint_foreign_key CASCADE;
CREATE VIEW meta_constraint_foreign_key AS
  SELECT
    pg_constraint.oid          AS id,
    pg_constraint.conname      AS name,
    pg_constraint.connamespace AS namespace_id,
    pg_constraint.conrelid     AS constrained_table_id,
    pg_constraint.conkey       AS constrained_column_numbers,
    pg_constraint.confrelid    AS referenced_table_id,
    pg_constraint.confkey      AS referenced_column_numbers
  FROM pg_constraint
  WHERE pg_constraint.contype = 'f';

DROP VIEW IF EXISTS meta_constraint_primary_key CASCADE;
CREATE VIEW meta_constraint_primary_key AS
  SELECT
    pg_constraint.oid          AS id,
    pg_constraint.conname      AS name,
    pg_constraint.connamespace AS namespace_id,
    pg_constraint.conrelid     AS constrained_table_id,
    pg_constraint.conkey       AS constrained_column_numbers
  FROM pg_constraint
  WHERE pg_constraint.contype = 'p';

DROP VIEW IF EXISTS meta_constraint_unique CASCADE;
CREATE VIEW meta_constraint_unique AS
  SELECT
    pg_constraint.oid          AS id,
    pg_constraint.conname      AS name,
    pg_constraint.connamespace AS namespace_id,
    pg_constraint.conrelid     AS constrained_table_id,
    pg_constraint.conkey       AS constrained_column_numbers
  FROM pg_constraint
  WHERE pg_constraint.contype = 'u';

DROP VIEW IF EXISTS meta_constraint_exclusion CASCADE;
CREATE VIEW meta_constraint_exclusion AS
  SELECT
    pg_constraint.oid          AS id,
    pg_constraint.conname      AS name,
    pg_constraint.connamespace AS namespace_id,
    pg_constraint.conrelid     AS constrained_table_id,
    pg_constraint.conkey       AS constrained_column_numbers,
    pg_constraint.conexclop    AS per_column_operator_ids
  FROM pg_constraint
  WHERE pg_constraint.contype = 'x';
```

```
DROP VIEW IF EXISTS meta_display_constraint_check CASCADE;
CREATE VIEW meta_display_constraint_check AS
  SELECT
    meta_constraint_check.id                         AS id,
    meta_constraint_check.name                       AS name,
    meta_constraint_check.namespace_id               AS namespace_id,
    meta_constraint_check.constrained_table_id       AS constrained_table_id,
    meta_constraint_check.constrained_column_numbers AS constrained_column_numbers,
    meta_constraint_check.clause                     AS clause
  FROM meta_constraint_check;

DROP VIEW IF EXISTS meta_display_constraint_check_single CASCADE;
CREATE VIEW meta_display_constraint_check_single AS
  SELECT
    meta_display_constraint_check.id                            AS id,
    meta_display_constraint_check.name                          AS name,
    meta_display_constraint_check.namespace_id                  AS namespace_id,
    meta_display_constraint_check.constrained_table_id          AS constrained_table_id,
    meta_display_constraint_check.constrained_column_numbers[1] AS
    constrained_column_number,
    meta_display_constraint_check.clause                        AS clause
  FROM meta_display_constraint_check
  WHERE cardinality(meta_display_constraint_check.constrained_column_numbers) = 1;

DROP VIEW IF EXISTS meta_display_constraint_check_multiple CASCADE;
CREATE VIEW meta_display_constraint_check_multiple AS
  SELECT
    meta_display_constraint_check.id                         AS id,
    meta_display_constraint_check.name                       AS name,
    meta_display_constraint_check.namespace_id               AS namespace_id,
    meta_display_constraint_check.constrained_table_id       AS constrained_table_id,
    meta_display_constraint_check.constrained_column_numbers AS
    constrained_column_numbers,
    meta_display_constraint_check.clause                     AS clause
  FROM meta_display_constraint_check
  WHERE cardinality(meta_display_constraint_check.constrained_column_numbers) != 1;

DROP VIEW IF EXISTS meta_display_constraint_foreign_key_single CASCADE;
CREATE VIEW meta_display_constraint_foreign_key_single AS
  SELECT
    meta_constraint_foreign_key.id                            AS id,
    meta_constraint_foreign_key.name                          AS name,
    meta_constraint_foreign_key.namespace_id                  AS namespace_id,
    meta_constraint_foreign_key.constrained_table_id          AS constrained_table_id,
    meta_constraint_foreign_key.constrained_column_numbers[1] AS
    constrained_column_number,
    ('REFERENCES ' || meta_table_column.name::text)           AS clause
  FROM meta_constraint_foreign_key
  JOIN meta_table        ON meta_table.id = meta_constraint_foreign_key.
    referenced_table_id
  JOIN meta_table_column ON (
    meta_table_column.table_id = meta_table.id AND
    meta_table_column.number = meta_constraint_foreign_key.referenced_column_numbers[1]
  )
  WHERE (
   cardinality(meta_constraint_foreign_key.constrained_column_numbers) = 1 AND
```

```sql
   cardinality(meta_constraint_foreign_key.referenced_column_numbers) = 1
  );

DROP FUNCTION IF EXISTS meta_display_column_name CASCADE;
CREATE FUNCTION meta_display_column_name(
  table_id      oid,
  column_number integer
) RETURNS text AS $$
DECLARE
  column_name text;
BEGIN
  SELECT meta_table_column.name INTO column_name
  FROM meta_table
  JOIN meta_table_column ON meta_table_column.table_id = meta_table.id
  WHERE meta_table.id = meta_display_column_name.table_id
    AND meta_table_column.number = meta_display_column_name.column_number;

  RETURN column_name;
END;
$$ LANGUAGE plpgsql;

DROP VIEW IF EXISTS meta_display_constraint_foreign_key_multiple CASCADE;
CREATE VIEW meta_display_constraint_foreign_key_multiple AS
  SELECT
    meta_constraint_foreign_key.id                        AS id,
    meta_constraint_foreign_key.name                      AS name,
    meta_constraint_foreign_key.namespace_id              AS namespace_id,
    meta_constraint_foreign_key.constrained_table_id      AS constrained_table_id,
    meta_constraint_foreign_key.constrained_column_numbers AS constrained_column_numbers
    ,
    meta_constraint_foreign_key.referenced_table_id       AS referenced_table_id,
    meta_constraint_foreign_key.referenced_column_numbers  AS referenced_column_numbers,
    (
      (
        SELECT string_agg(meta_display_column_name(constrained_table_id,
    constrained_column_number), ', ')
        FROM unnest(meta_constraint_foreign_key.constrained_column_numbers)
        AS constrained_column_number
      ) || ' REFERENCES ' || (
        SELECT string_agg(meta_display_column_name(referenced_table_id,
    referenced_column_number), ', ')
        FROM unnest(meta_constraint_foreign_key.referenced_column_numbers)
        AS referenced_column_number
      )
    )                                                     AS clause
  FROM meta_constraint_foreign_key
  WHERE (
   cardinality(meta_constraint_foreign_key.constrained_column_numbers) != 1 AND
   cardinality(meta_constraint_foreign_key.referenced_column_numbers) != 1
  );

DROP VIEW IF EXISTS meta_display_constraint_primary_key_single CASCADE;
CREATE VIEW meta_display_constraint_primary_key_single AS
  SELECT
    meta_constraint_primary_key.id                           AS id,
    meta_constraint_primary_key.name                         AS name,
    meta_constraint_primary_key.namespace_id                 AS namespace_id,
    meta_constraint_primary_key.constrained_table_id         AS constrained_table_id,
    meta_constraint_primary_key.constrained_column_numbers[1] AS
    constrained_column_number,
    'PRIMARY KEY'                                            AS clause
  FROM meta_constraint_primary_key
  WHERE cardinality(meta_constraint_primary_key.constrained_column_numbers) = 1;

DROP VIEW IF EXISTS meta_display_constraint_primary_key_multiple CASCADE;
CREATE VIEW meta_display_constraint_primary_key_multiple AS
  SELECT
    meta_constraint_primary_key.id                         AS id,
    meta_constraint_primary_key.name                       AS name,
    meta_constraint_primary_key.namespace_id               AS namespace_id,
    meta_constraint_primary_key.constrained_table_id       AS constrained_table_id,
    meta_constraint_primary_key.constrained_column_numbers  AS
    constrained_column_numbers,
    (
      'PRIMARY KEY ' || (
        SELECT string_agg(meta_display_column_name(constrained_table_id,
    constrained_column_number), ', ')
        FROM unnest(meta_constraint_primary_key.constrained_column_numbers)
        AS constrained_column_number
      )
    )                                                     AS clause
  FROM meta_constraint_primary_key
  WHERE cardinality(meta_constraint_primary_key.constrained_column_numbers) != 1;

DROP VIEW IF EXISTS meta_display_constraint_unique_single CASCADE;
CREATE VIEW meta_display_constraint_unique_single AS
  SELECT
    meta_constraint_unique.id                            AS id,
    meta_constraint_unique.name                          AS name,
    meta_constraint_unique.namespace_id                  AS namespace_id,
    meta_constraint_unique.constrained_table_id          AS constrained_table_id,
    meta_constraint_unique.constrained_column_numbers[1] AS constrained_column_number,
    'UNIQUE'                                             AS clause
  FROM meta_constraint_unique
  WHERE cardinality(meta_constraint_unique.constrained_column_numbers) = 1;

DROP VIEW IF EXISTS meta_display_constraint_unique_multiple CASCADE;
CREATE VIEW meta_display_constraint_unique_multiple AS
  SELECT
    meta_constraint_unique.id                          AS id,
    meta_constraint_unique.name                        AS name,
    meta_constraint_unique.namespace_id                AS namespace_id,
    meta_constraint_unique.constrained_table_id        AS constrained_table_id,
    meta_constraint_unique.constrained_column_numbers  AS constrained_column_numbers,
    (
      'UNIQUE ' || (
        SELECT string_agg(meta_display_column_name(constrained_table_id,
    constrained_column_number), ', ')
        FROM unnest(meta_constraint_unique.constrained_column_numbers)
        AS constrained_column_number
      )
    )                                                  AS clause
  FROM meta_constraint_unique
  WHERE cardinality(meta_constraint_unique.constrained_column_numbers) != 1;

DROP VIEW IF EXISTS meta_display_constraint_exclusion CASCADE;
CREATE VIEW meta_display_constraint_exclusion_multiple AS
  SELECT
    meta_constraint_exclusion.id                         AS id,
    meta_constraint_exclusion.name                       AS name,
    meta_constraint_exclusion.namespace_id               AS namespace_id,
    meta_constraint_exclusion.constrained_table_id       AS constrained_table_id,
    meta_constraint_exclusion.constrained_column_numbers AS constrained_column_numbers,
    (
      'EXCLUDE ' || (
        SELECT
          string_agg((
            meta_display_column_name(constrained_table_id, column_number)
            || ' WITH ' || meta_operator.name
          ), ', ')
        FROM unnest(
          meta_constraint_exclusion.constrained_column_numbers,
          meta_constraint_exclusion.per_column_operator_ids
        ) WITH ORDINALITY AS column_operator(column_number, operator_id)
        JOIN meta_operator ON meta_operator.id = column_operator.operator_id
      )
    )                                                  AS clause
  FROM meta_constraint_exclusion;

DROP VIEW IF EXISTS meta_display_contraint_single CASCADE;
CREATE VIEW meta_display_contraint_single AS
  (
    SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
    clause
    FROM meta_display_constraint_check_single
  ) UNION ALL (
```

```sql
    SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
    clause
    FROM meta_display_constraint_foreign_key_single
  ) UNION ALL (
    SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
    clause
    FROM meta_display_constraint_primary_key_single
  ) UNION ALL (
    SELECT id, name, namespace_id, constrained_table_id, constrained_column_number,
    clause
    FROM meta_display_constraint_unique_single
  );

DROP VIEW IF EXISTS meta_display_contraint_multiple CASCADE;
CREATE VIEW meta_display_contraint_multiple AS
  (
    SELECT id, name, namespace_id, constrained_table_id, clause
    FROM meta_display_constraint_check_multiple
  ) UNION ALL (
    SELECT id, name, namespace_id, constrained_table_id, clause
    FROM meta_display_constraint_foreign_key_multiple
  ) UNION ALL (
    SELECT id, name, namespace_id, constrained_table_id, clause
    FROM meta_display_constraint_primary_key_multiple
  ) UNION ALL (
    SELECT id, name, namespace_id, constrained_table_id, clause
    FROM meta_display_constraint_unique_multiple
  ) UNION ALL (
    SELECT id, name, namespace_id, constrained_table_id, clause
    FROM meta_display_constraint_exclusion_multiple
  );
```

```sql
DROP VIEW IF EXISTS main_table_column_constraint CASCADE;
CREATE VIEW main_table_column_constraint AS
  SELECT
    meta_namespace.name                    AS schema_name,
    meta_table.name                        AS table_name,
    meta_table_column.name                 AS column_name,
    meta_display_contraint_single.name     AS contraint_name,
    meta_display_contraint_single.clause   AS contraint_clause
  FROM meta_table
  JOIN meta_namespace ON meta_table.namespace_id = meta_namespace.id
  JOIN meta_table_column
    ON meta_table_column.table_id = meta_table.id
  LEFT JOIN meta_display_contraint_single ON (
    meta_display_contraint_single.constrained_table_id = meta_table.id AND
    meta_display_contraint_single.constrained_column_number = meta_table_column.number
  );

DROP VIEW IF EXISTS main_table_constraint CASCADE;
CREATE VIEW main_table_constraint AS
  SELECT
    meta_namespace.name                      AS schema_name,
    meta_table.name                          AS table_name,
    meta_display_contraint_multiple.name     AS constraint_name,
    meta_display_contraint_multiple.clause   AS constraint_clause
  FROM meta_table
  JOIN meta_namespace ON meta_table.namespace_id = meta_namespace.id
  LEFT JOIN meta_display_contraint_multiple ON (
    meta_display_contraint_multiple.constrained_table_id = meta_table.id
  );

DROP PROCEDURE IF EXISTS main_table_print_pretty;
CREATE PROCEDURE main_table_print_pretty (
  table_schema  text,
  table_name    text
) AS $$
DECLARE
  col        record;
  col_constr record;

  C1W  integer;
  C2W  integer;
  C31W integer;
  C32W integer;
  REM  integer;
BEGIN
  C1W := 2;
  C2W := 12;
  C31W := 8;
  C32W := 64 + 8;
  REM := 11;

  ----- HEADER -----
  RAISE INFO
    '%',
    rpad(
      '|--- Table "' || table_schema || '.' || table_name || '" Information ',
      C1W + C2W + C31W + C32W + REM,
      '-'
    ) || '|';

  RAISE INFO
    '| % | % | % |',
    rpad('N', C1W, ' '),
    rpad('Name', C2W, ' '),
    rpad('Attributes', C31W + C32W + 2, ' ');

  RAISE INFO
    '%',
    rpad('|', C1W + C2W + C31W + C32W + REM, '-') || '|';


  ----- ROWS -----
  FOR col IN
    SELECT
      meta_table_column.number      AS column_number,
      meta_table_column.name        AS column_name,
      meta_type.name                AS type_name,
      meta_table_column.is_nullable AS is_nullable,
      meta_table.id                 AS table_id
    FROM meta_table
    JOIN meta_namespace ON meta_namespace.id = meta_table.namespace_id
    JOIN meta_table_column ON meta_table.id = meta_table_column.table_id
    JOIN meta_type ON meta_type.id = meta_table_column.type_id
    WHERE meta_namespace.name = main_table_print_pretty.table_schema
      AND meta_table.name = main_table_print_pretty.table_name
      AND meta_table_column.number > 0
  LOOP
    RAISE INFO
      '| % | % | % |',
      rpad(col.column_number::text, C1W, ' '),
      rpad(col.column_name, C2W, ' '),
      (rpad('Type', C31W, ' ') || ': ' || rpad(col.type_name, C32W, ' '));
    RAISE INFO
      '| % | % | % |',
      rpad('', C1W, ' '),
      rpad('', C2W, ' '),
      rpad('Null', C31W, ' ') || ': ' || rpad((
        CASE WHEN col.is_nullable THEN 'NULLABLE' ELSE 'NOT NULL' END
      ), C32W, ' ');

    FOR col_constr IN
      SELECT *
      FROM meta_comment
      WHERE meta_comment.owner_id = col.table_id
        AND meta_comment.child_id = col.column_number
    LOOP
      IF NOT col_constr IS NULL THEN
        RAISE INFO
          '| % | % | % |',
          rpad('', C1W, ' '),
          rpad('', C2W, ' '),
          rpad('Comment', C31W, ' ') || ': ' || rpad(
            col_constr.content, C32W, ' ');
      END IF;
    END LOOP;
    FOR col_constr IN
      SELECT
        contraint_name   AS name,
        contraint_clause AS clause
      FROM main_table_column_constraint
      WHERE
        main_table_column_constraint.schema_name = main_table_print_pretty.table_schema AND
        main_table_column_constraint.table_name = main_table_print_pretty.table_name AND
        main_table_column_constraint.column_name = col.column_name
    LOOP
      IF NOT col_constr.name IS NULL THEN
        RAISE INFO
          '| % | % | % |',
          rpad('', C1W, ' '),
          rpad('', C2W, ' '),
          (
            rpad('Constr', C31W, ' ') || ': ' || rpad(
              (col_constr.name || ' ' || col_constr.clause), C32W, ' '
            )
          );
      END IF;
    END LOOP;
   END LOOP;

  FOR col IN
    SELECT
      main_table_constraint.constraint_name    AS constraint_name,
      main_table_constraint.constraint_clause  AS constraint_clause
    FROM main_table_constraint
    WHERE
      main_table_constraint.schema_name = main_table_print_pretty.table_schema AND
      main_table_constraint.table_name = main_table_print_pretty.table_name
  LOOP
    RAISE INFO
      '| % |',
      (rpad('Constr', C31W, ' ') || ': ' ||
      (col.constraint_name || ' ' || col.constraint_clause))
      ;
  END LOOP;
END;
$$ language plpgsql;

drop procedure IF EXISTS solution;
create or replace procedure solution(
  table_name text
) as $$
declare
  table_schema text;
begin
  select information_schema.tables.table_schema into table_schema
  from information_schema.tables
  where information_schema.tables.table_name = solution.table_name
  limit 1;

  call main_table_print_pretty(table_schema, table_name);
end;
$$ language plpgsql;

call solution('person');
```

4 Таблица

```
|--- Table "public.person" Information
    -----------------------------------------------------------------|
| N  | Name         | Attributes
                     |
|----------------------------------------------------------------------------------------------------------

| 1  | id           | Type    : numeric
                     |
|    |               | Null    : NOT NULL |
|    |               | Comment : The unique number of the person |
|    |               | Constr  : person_pkey PRIMARY KEY |
| 2  | last_name     | Type    : varchar |
|    |               | Null    : NOT NULL |
|    |               | Comment : Last name of the person |
| 3  | first_name    | Type    : varchar |
|    |               | Null    : NOT NULL |
|    |               | Comment : The name of the person |
| 4  | patronymic    | Type    : varchar |
|    |               | Null    : NULLABLE |
|    |               | Comment : The patronymic of the person |
| 5  | birth_date    | Type    : date |
|    |               | Null    : NOT NULL |
|    |               | Comment : Date of birth of a person |
| 6  | gender        | Type    : bpchar |
|    |               | Null    : NOT NULL |
|    |               | Constr  : person_gender_check (gender = ANY (ARRAY['M'::bpchar, 'F'::bpchar])) |
|    |               | Constr  : person_gender_check1 (gender = ANY (ARRAY['M'::bpchar, 'F'::bpchar])) |
| 7  | foreigner     | Type    : varchar |
|    |               | Null    : NOT NULL |
| 8  | created_who   | Type    : varchar |
|    |               | Null    : NOT NULL |
| 9  | created_when  | Type    : date |
|    |               | Null    : NOT NULL |
| 10 | edited_who    | Type    : varchar |
|    |               | Null    : NOT NULL |
| 11 | edited_when   | Type    : date |
|    |               | Null    : NOT NULL |
| 12 | death_date    | Type    : date |
|    |               | Null    : NULLABLE |
|    |               | Comment : Date of death of a person |
| 13 | pin           | Type    : varchar |
|    |               | Null    : NULLABLE |
|    |               | Constr  : person_pin_key UNIQUE |
| 14 | inn           | Type    : varchar |
|    |               | Null    : NULLABLE
```

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

5 Вывод

Данная лабораторная работа помогла мне изучить системный каталог PostgreSQL.

Список литературы