Университет ИТМО

Факультет ФПИ и КТ P33131

**Отчет по лабраторной работе №1**

**«Распределенные системы хранения данных»**

Вариант 816

Студент:

Чжоу Хунсян

Гр.P33131

Преподаватель:

# Оглавление

**ЗАДАНИЕ ........................................................................................................................................ 2** **РЕШЕНИЕ ....................................................................................................................................... 2** **СТРУКТУРА ТАБЛИЦЫ .................................................................................................................. 3** **ПРИМЕР .......................................................................................................................................... 5** **ВЫВОД ...........................................................................................................................................**

# Задание

Используя сведения из системных каталогов получить информацию о первичных и внешних ключах схемы: Номер по порядку, Имя ограничения целостности, Тип, Имя столбца, Имя таблицы. Кроме того, для внешних ключей указать Имя таблицы и Имя столбца на которые ссылаются эти ключи. Тип ограничения: R - внешний ключ, P - первичный ключ,

Имя ограничения Тип Имя столбца Имя таблицы Имя таблицы Имя столбц

--------------------- --- ------------------- ----------- ----------- ----------

ПЛАН\_PK P ИД Н\_ПЛАНЫ

ПЛАН\_КАФ\_FK R ОТД\_ИД\_ЗАКРЕПЛЕН\_ЗА Н\_ПЛАНЫ Н\_ОТДЕЛЫ ИД

ПЛАН\_НАПС\_FK R НАПС\_ИД Н\_ПЛАНЫ Н\_НАПРАВЛЕН ИД

ПЛАН\_ПЛАН\_FK R ПЛАН\_ИД Н\_ПЛАНЫ Н\_ПЛАНЫ ИД

ПЛАН\_ПЛАН\_ОСНОВ\_НА\_FK R ПЛАН\_ИД\_ОСНОВ\_НА Н\_ПЛАНЫ Н\_ПЛАНЫ ИД

ПЛАН\_ТЛП\_FK R ТПЛ\_ИД Н\_ПЛАНЫ Н\_ТИПЫ\_ПЛАН ИД

ПЛАН\_ФАК\_FK R ОТД\_ИД Н\_ПЛАНЫ Н\_ОТДЕЛЫ ИД

ПЛАН\_ФО\_FK R ФО\_ИД Н\_ПЛАНЫ Н\_ФОРМЫ\_ОБУ ИД

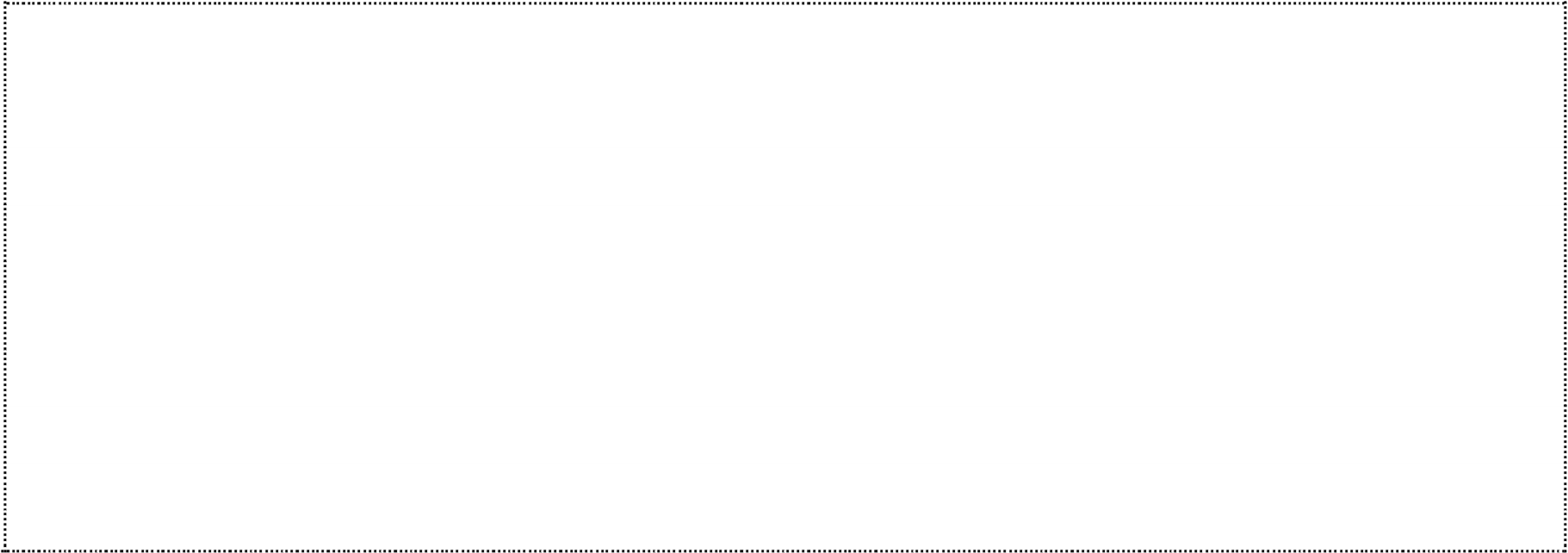
УЧЕН\_PK P ИД Н\_УЧЕНИКИ

УЧЕН\_ОБУЧ\_FK R ВИД\_ОБУЧ\_ИД Н\_УЧЕНИКИ Н\_ОБУЧЕНИЯ ИД\_ОБУЧ\_ИД

УЧЕН\_ОБУЧ\_FK R ЧЛВК\_ИД Н\_УЧЕНИКИ Н\_ОБУЧЕНИЯ ЧЛВК\_ИД

УЧЕН\_ПЛАН\_FK R ПЛАН\_ИД Н\_УЧЕНИКИ Н\_ПЛАНЫ ИД

УЧЕН\_ПЛАН\_ГРУППА\_FK R ГРУППА Н\_УЧЕНИКИ Н\_ГРУППЫ\_ПЛ ГРУППА

УЧЕН\_ПЛАН\_ГРУППА\_FK R ПЛАН\_ИД Н\_УЧЕНИКИ Н\_ГРУППЫ\_ПЛ ПЛАН\_ИД ...

# Решение

CREATE OR REPLACE PROCEDURE *get\_pf\_constraint\_info*(schema TEXT)  
 LANGUAGE plpgsql  
AS  
$$  
DECLARE  
 constraint\_record RECORD;  
BEGIN  
 RAISE INFO '% % % % % %',  
 *format*('%-40s', 'Имя ограничения'),  
 *format*('%-3s', 'Тип'),  
 *format*('%-20s', 'Имя таблицы'),  
 *format*('%-30s', 'Имя столбцов'),  
 *format*('%-20s', 'Имя внешней таблицы'),  
 *format*('%-30s', 'Имя внешних столбцов');  
 RAISE INFO '% % % % % %',  
 *repeat*('-', 40),  
 *repeat*('-', 3),  
 *repeat*('-', 20),  
 *repeat*('-', 30),  
 *repeat*('-', 20),  
 *repeat*('-', 30);  
 FOR constraint\_record IN  
 SELECT  
 conname AS constraint\_name,  
 contype AS constraint\_type,  
 conrelid::regclass::text AS table\_name,  
 *array\_agg*(DISTINCT a.attname) AS column\_names,  
 f.relname AS foreign\_table,  
 *array\_agg*(DISTINCT af.attname) AS foreign\_column\_names  
 FROM  
 pg\_constraint c  
 JOIN pg\_namespace namespace ON c.connamespace = namespace.oid  
 JOIN pg\_class t ON c.conrelid = t.oid  
 JOIN pg\_attribute a ON a.attrelid = t.oid AND a.attnum = ANY(c.conkey)  
 LEFT JOIN pg\_class f ON c.confrelid = f.oid  
 LEFT JOIN pg\_attribute af ON af.attrelid = f.oid AND af.attnum = ANY(c.confkey)  
 WHERE namespace.nspname = schema AND contype IN ('p', 'f')  
 GROUP BY conname, contype, table\_name, foreign\_table  
 LOOP  
 IF constraint\_record.constraint\_type = 'p' THEN  
 RAISE INFO '% % % %',  
 *format*('%-40s', constraint\_record.constraint\_name),  
 *format*('%-3s', 'P'),  
 *format*('%-20s', constraint\_record.table\_name),  
 *format*('%-30s', *array\_to\_string*(constraint\_record.column\_names, ', '));  
 ELSIF constraint\_record.constraint\_type = 'f' THEN  
 RAISE INFO '% % % % % %',  
 *format*('%-40s', constraint\_record.constraint\_name),  
 *format*('%-3s', 'R'),  
 *format*('%-20s', constraint\_record.table\_name),  
 *format*('%-30s', *array\_to\_string*(constraint\_record.column\_names, ', ')),  
 *format*('%-20s', constraint\_record.foreign\_table),  
 *format*('%-30s', *array\_to\_string*(constraint\_record.foreign\_column\_names, ', '));  
 END IF;  
 END LOOP;  
END  
$$;  
  
call *get\_pf\_constraint\_info*('s336184');

# Структура таблицы

-- Create TYPE ENUM

CREATE TYPE PROBLEM\_TYPE AS ENUM ('UI', 'BUGS', 'SCRIPT');

CREATE TYPE HOUSE\_TYPE AS ENUM ('APARTMENTS', 'VILLAS', 'HIGH-END','ORDINARY');

CREATE TYPE DEVICE\_TYPE AS ENUM ('AIR\_CONDITION','LIGHT', 'HUMIDIFIER', 'BATHTUB', 'OUTLET','CURTAINS', 'FAN', 'CAMERA', 'WATER\_HEATER');

CREATE TYPE SENSOR\_TYPE AS ENUM ('TEMPERATURE', 'HUMIDITY', 'SMOKE');

CREATE TYPE ACTION\_TYPE AS ENUM ('CLOSE','OPEN','SWITCH\_OFF','SWITCH\_ON','ADJUST\_VALUE','TURN\_ON','TURN\_OFF');

CREATE TYPE SCRIPT\_TYPE AS ENUM ('CONDITIONAL','SCHEDULE');

CREATE TYPE COUNTRY AS ENUM('US','UK','RUSSIAN','CHINA','FRANCE');

CREATE TYPE CITY AS ENUM('Shanghai', 'Beijing', 'Shenzhen', 'Guangzhou', 'Chengdu','Paris', 'Marseille', 'Lyon',

'Toulouse','Cambridge', 'Edinburgh', 'London', 'Liverpool','New York', 'Los Angeles', 'Chicago', 'Boston');

CREATE TYPE ROOM\_TYPE AS ENUM('KITCHEN','BEDROOM','BATHROOM','LIVING');

CREATE TYPE GENDER AS ENUM ('MALE','FEMALE');

-- Create Table

CREATE TABLE IF NOT EXISTS family( id SERIAL PRIMARY KEY NOT NULL, name VARCHAR(64) NOT NULL, info TEXT NOT NULL

);

CREATE TABLE IF NOT EXISTS "user"( id SERIAL PRIMARY KEY NOT NULL, password VARCHAR(256) NOT NULL, gender GENDER NOT NULL, username VARCHAR(64) NOT NULL, age INT NOT NULL CHECK (age > 0)

);

CREATE TABLE IF NOT EXISTS supporter( id SERIAL PRIMARY KEY NOT NULL, password VARCHAR(256) NOT NULL, username VARCHAR(64) NOT NULL, is\_free BOOLEAN NOT NULL DEFAULT TRUE

);

CREATE TABLE IF NOT EXISTS issue( id SERIAL PRIMARY KEY NOT NULL, user\_id INT NOT NULL REFERENCES "user"(id), supporter\_id INT NOT NULL REFERENCES supporter(id), is\_finished BOOLEAN NOT NULL DEFAULT FALSE, description TEXT NOT NULL, issue\_type PROBLEM\_TYPE NOT NULL, create\_time DATE DEFAULT CURRENT\_DATE

);

CREATE TABLE IF NOT EXISTS address( id SERIAL NOT NULL PRIMARY KEY, country COUNTRY NOT NULL, city CITY NOT NULL, street VARCHAR(128) NOT NULL

);

CREATE TABLE IF NOT EXISTS house( id SERIAL PRIMARY KEY NOT NULL, address\_id INT REFERENCES address(id) NOT NULL, house\_type HOUSE\_TYPE NOT NULL

);

CREATE TABLE IF NOT EXISTS room( id SERIAL PRIMARY KEY NOT NULL, house\_id INT NOT NULL REFERENCES house(id), area\_size FLOAT NOT NULL, height FLOAT NOT NULL, room\_type ROOM\_TYPE NOT NULL, is\_filled BOOLEAN NOT NULL DEFAULT FALSE

);

CREATE TABLE IF NOT EXISTS device( id SERIAL PRIMARY KEY NOT NULL, room\_id INT REFERENCES room(id) NOT NULL, manufacture VARCHAR(256) NOT NULL, available BOOLEAN NOT NULL DEFAULT TRUE, device\_type DEVICE\_TYPE NOT NULL

);

CREATE TABLE IF NOT EXISTS sensor( id SERIAL PRIMARY KEY NOT NULL, room\_id INT REFERENCES room(id) NOT NULL, manufacture VARCHAR(256) NOT NULL, available BOOLEAN NOT NULL DEFAULT TRUE, sensor\_type SENSOR\_TYPE NOT NULL

);

CREATE TABLE IF NOT EXISTS device\_action( id SERIAL PRIMARY KEY NOT NULL, device\_type DEVICE\_TYPE NOT NULL, action\_type ACTION\_TYPE NOT NULL, description TEXT

);

CREATE TABLE IF NOT EXISTS script( id SERIAL PRIMARY KEY NOT NULL, creator INT REFERENCES "user"(id) NOT NULL, script\_type SCRIPT\_TYPE NOT NULL

);

CREATE TABLE IF NOT EXISTS schedule\_script( id SERIAL PRIMARY KEY NOT NULL, script\_id INT REFERENCES script(id) NOT NULL, action\_time TIME NOT NULL, repeat\_on\_monday BOOLEAN NOT NULL DEFAULT FALSE, repeat\_on\_tuesday BOOLEAN NOT NULL DEFAULT FALSE, repeat\_on\_wednesday BOOLEAN NOT NULL DEFAULT FALSE, repeat\_on\_thursday BOOLEAN NOT NULL DEFAULT FALSE, repeat\_on\_friday BOOLEAN NOT NULL DEFAULT FALSE, repeat\_on\_saturday BOOLEAN NOT NULL DEFAULT FALSE, repeat\_on\_sunday BOOLEAN NOT NULL DEFAULT FALSE

);

CREATE TABLE IF NOT EXISTS condition\_script( id SERIAL PRIMARY KEY NOT NULL, script\_id INT REFERENCES script(id) NOT NULL, condition\_text TEXT NOT NULL

);

CREATE TABLE IF NOT EXISTS contact( user\_id INT REFERENCES "user"(id) NOT NULL UNIQUE, email VARCHAR(128) NOT NULL UNIQUE, phone\_num VARCHAR(64) NOT NULL UNIQUE

);

CREATE TABLE IF NOT EXISTS list\_action\_script( id SERIAL NOT NULL PRIMARY KEY, script\_id INT REFERENCES script(id), action\_id INT REFERENCES device\_action(id) );

CREATE TABLE IF NOT EXISTS list\_script\_user( id SERIAL NOT NULL PRIMARY KEY, script\_id INT REFERENCES script(id), user\_id INT REFERENCES "user"(id),

UNIQUE (script\_id, user\_id)

);

CREATE TABLE IF NOT EXISTS list\_user\_house( id SERIAL NOT NULL PRIMARY KEY, user\_id INT REFERENCES "user"(id), house\_id INT REFERENCES house(id),

UNIQUE (user\_id, house\_id)

);

CREATE TABLE IF NOT EXISTS list\_user\_family( id SERIAL NOT NULL PRIMARY KEY, user\_id INT REFERENCES "user"(id), family\_id INT REFERENCES house(id),

UNIQUE (user\_id, family\_id) );

Скрипт:

psql -h pg -d studs -c call get\_pf\_constraint\_info(‘s336184’);

# Пример

studs.s336184> CALL get\_pf\_constraint\_info('s336184') Имя ограничения Тип Имя столбца Имя таблицы Имя таблицы Имя столбца ---------------------------------------- --- --------------- -------------------- --------------------

--------------- address\_pkey P id address condition\_script\_pkey P id condition\_script condition\_script\_script\_id\_fkey R script\_id condition\_script script id contact\_user\_id\_fkey R user\_id contact user id device\_action\_pkey P id device\_action device\_pkey P id device device\_room\_id\_fkey R room\_id device room id family\_pkey P id family house\_address\_id\_fkey R address\_id house address id house\_pkey P id house issue\_pkey P id issue issue\_supporter\_id\_fkey R supporter\_id issue supporter id issue\_user\_id\_fkey R user\_id issue user id list\_action\_script\_action\_id\_fkey R action\_id list\_action\_script device\_action id list\_action\_script\_pkey P id list\_action\_script list\_action\_script\_script\_id\_fkey R script\_id list\_action\_script script id list\_script\_user\_pkey P id list\_script\_user list\_script\_user\_script\_id\_fkey R script\_id list\_script\_user script id list\_script\_user\_user\_id\_fkey R user\_id list\_script\_user user id list\_user\_family\_family\_id\_fkey R family\_id list\_user\_family house id list\_user\_family\_pkey P id list\_user\_family list\_user\_family\_user\_id\_fkey R user\_id list\_user\_family user id list\_user\_house\_house\_id\_fkey R house\_id list\_user\_house house id list\_user\_house\_pkey P id list\_user\_house list\_user\_house\_user\_id\_fkey R user\_id list\_user\_house user id room\_house\_id\_fkey R house\_id room house id room\_pkey P id room schedule\_script\_pkey P id schedule\_script schedule\_script\_script\_id\_fkey R script\_id schedule\_script script id script\_creator\_fkey R creator script user id script\_pkey P id script sensor\_pkey P id sensor sensor\_room\_id\_fkey R room\_id sensor room id supporter\_pkey P id supporter

# Вывод

В ходе выполнения работы изучал как использовать системный каталог чтобы читать информацию таблиц в БД.