## **Лабораторная работа № 5 Работа с представлениями**

**Теоретические сведения**

Основная структурная единица реляционных БД – таблицы, но язык SQL предоставляет еще один способ организации данных. Представление – это запрос на выборку, которому присваивается уникальное имя и который можно сохранять или удалять из БД как хранимую процедуру. Представления позволяют увидеть результаты сохраненного запроса так, как будто это полноценная таблица. PostgreSQL, встретив в запросе ссылку на представление, ищет его определение в БД. Пользовательский запрос с участием представления преобразуется в эквивалентный запрос к исходным таблицам. Если определение представления простое, то каждая строка представления формируется «на лету». Если определение сложное, PostgreSQL материализует представление в виде временной таблицы. Клиент, обращаясь к представлению, будет видеть только столбцы результирующей таблицы. Не имеет значения, сколько столбцов в исходной таблице и является ли запрос, лежащий в основе представления, одно- или многотабличным. Клиенту можно запретить обращаться к исходным таблицам, но снабдить привилегиями обращения к представлениям. На одном наборе таблиц можно создать гибкие системы доступа.

Преимущества представлений:

* *безопасность* – каждый пользователь имеет доступ к небольшому числу представлений, содержащих только ту информацию, которую ему позволе­но знать;
* *простота запросов* – с помощью представления можно извлечь данные из не­скольких таблиц и представить их как одну таблицу (запрос ко многим таблицам заменяется однотабличным запросом к представлению);
* *простота структуры* – представления позволяют создать для каждого пользовате­ля собственную структуру БД (отображаются данные, которые ему нужны);
* *защита от изменений* – таблицы и их структура могут постоянно изменяться и переименовываться; представления позволяют создавать виртуальные таблицы со старыми именами и структурой, позволяя избежать модификации приложений.

Недостатки представлений:

* *производительность* – представления создают видимость существования таблицы, и PostgreSQL приходится преобразовывать запрос к представлению в запрос к исходным таблицам; если представление отображает многотабличный запрос, то простой запрос к представлению становится сложным объединением;
* *ограничение на обновление* – когда пользователь пытается обновить строки представления, PostgreSQL необходимо обновить строки в исходных таблицах; это возможно только для простых представлений, сложные представления доступны только для выборки.

Поэтому не стоит везде применять представления вместо исходных таблиц.

**Создание представлений**. Осуществляется при помощи оператора

*CRFATE [OR REPLACE] [ALGORITHM = {UNDEFINED / MERGE / TEMPTABLE}] VIEW view\_name [(column\_list)] AS select\_statement*

*[WITH [CASCADED / LOCAL] CHECK OPTION];*

Оператор создает представление *view\_name* со столбцами, перечисленными в *column\_list*, на основании запроса *select\_statement*.

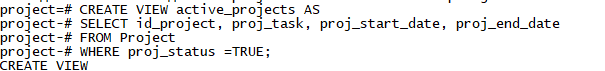
Рассмотрим создание представления: у нас есть таблица Project, в которой хранятся данные о проектах, и мы хотим создать представление, показывающее только активные проекты:

CREATE VIEW active\_projects AS

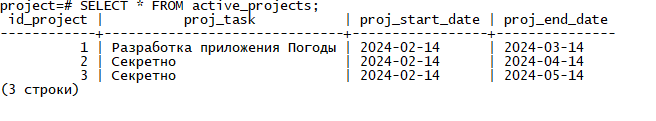
SELECT id\_project, proj\_task, proj\_start\_date, proj\_end\_date

FROM Project

WHERE proj\_status =TRUE;



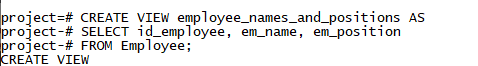
Вызов представления:

SELECT \* FROM active\_projects; Нет данных  


При создании представления можно явно указать список столбцов, например: мы хотим создать представление, которое будет содержать только имена сотрудников и их должности. Мы явно указываем эти столбцы в представлении:  
CREATE VIEW employee\_names\_and\_positions AS

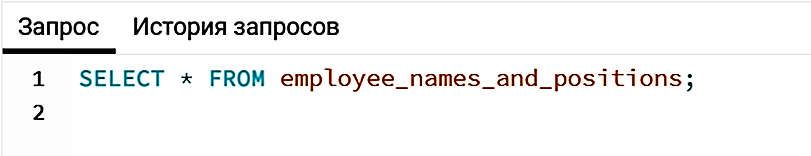
SELECT id\_employee, em\_name, em\_position

FROM Employee;

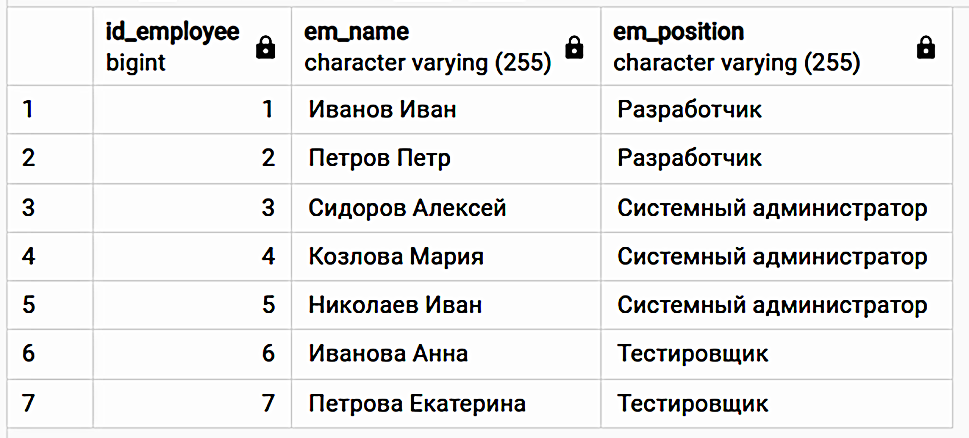


Пример запроса в PgAdmin4(аналогично в консоли):

SELECT \* FROM employee\_names\_and\_positions;



Результат:



Представления, не содержащие дополнитель­ных столбцов, называются *вертикальными представлениями.* Они применяются для ограничения доступа пользователей к столбцам таблицы.

Кроме вертикальных представлений используются *горизонтальные представ­ления,* которые делают види­мыми только те строки, с которыми работают пользователи. Например, чтобы в электронном магазине каждый менеджер видел только те товарные позиции, за которые отвечает, можно создать представления для менеджеров. Учетные записи менеджеров следует лишить привилегии доступа к таблице и разрешить просматривать только свои представления.

Создадим представление *project\_details,* содержищий информацию о проектах, включая задачи проекта, даты начала и окончания, статус проекта, название компании клиента (cl\_company) и задачу отдела (dep\_task), если такая информация доступна:

CREATE VIEW project\_details AS

SELECT p.id\_project,

p.proj\_task,

p.proj\_start\_date,

p.proj\_end\_date,

p.proj\_status,

c.cl\_company AS client\_company,

d.dep\_task AS department\_task

FROM

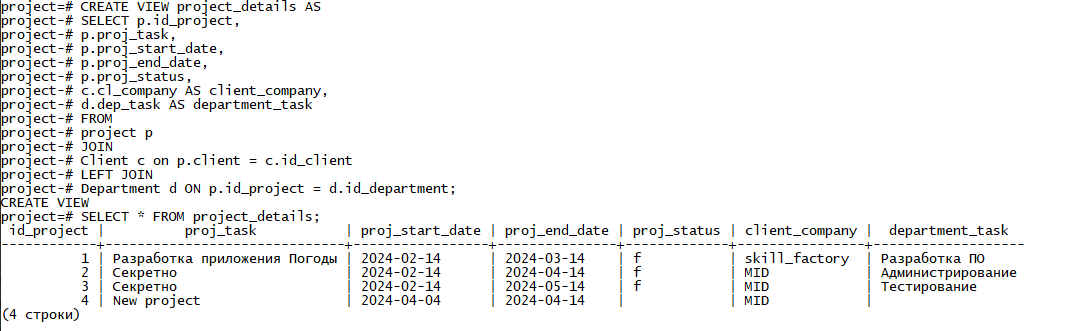
project p

JOIN  
Client c on p.client = c.id\_client

LEFT JOIN

Department d ON p.id\_project = d.id\_department;

SELECT \* FROM project\_details;



Такое представление может быть полезно для быстрого получения общей информации о проектах, включая информацию о клиентах и отделах, без необходимости писать сложные JOIN-запросы каждый раз.

Наиболее удобно использовать представления для формирования сгруппированных таблиц. При работе с такими таблицами PosgreSQL самостоятельно формирует временную таблицу – см. пункт «Пример выполнения работы» (пример 2).

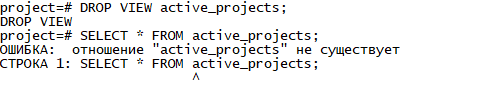
**Удаление представлений.** Выполняется с помощью оператора:

*DROP VIEW [IF EXISTS] view\_name [, view\_name] … ;*

Оператор позволяет уничтожить одно или несколько представлений, например:

DROP VIEW active\_projects;

SELECT \* FROM active\_projects;



**Обновление представлений.** Некоторые представления могут быть обновляемыми, если они соответствуют определенным критериям. Однако большинство представлений в PostgreSQL являются только для чтения. Если представление обновляемо, вы можете выполнять операции INSERT, UPDATE и DELETE через него.

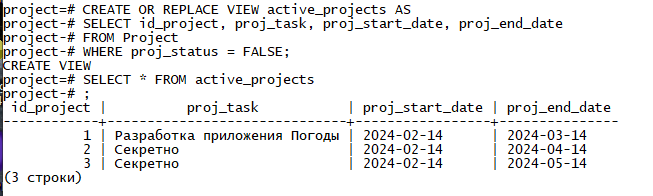
CREATE OR REPLACE VIEW active\_projects AS

SELECT id\_project, proj\_task, proj\_start\_date, proj\_end\_date

FROM Project

WHERE proj\_status = FALSE;

SELECT \* FROM active\_projects;



**Практическая работа**

При выполнении лабораторной работы необходимо:

* для заданной предметной области создать 2 представления в БД, 2 обновления этих представлений;
* сформировать запрос к одному из представлений;
* составить отчет по лабораторной работе.
* \* создать многотабличный запрос , использующий 2 созданных представления.

**Пример выполнения работы**

* + 1. Создадим вертикальное представление *client\_names*, которое будет отобра­жать фамилию и инициалы покупателей, скрывая другие поля.

Перед тем как создать данное представление, нужно изменить значение таблицы client, прописав данную команду:

Update client set cl\_name='Alex Akbulatov' where id\_client= 4;

CREATE VIEW client\_names AS

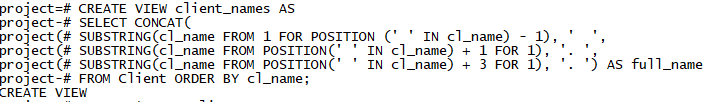
SELECT CONCAT(

SUBSTRING(cl\_name FROM 1 FOR POSITION (' ' IN cl\_name) - 1), ' ',

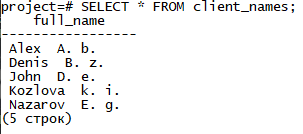
SUBSTRING(cl\_name FROM POSITION(' ' IN cl\_name) + 1 FOR 1),'. ',

SUBSTRING(cl\_name FROM POSITION(' ' IN cl\_name) + 3 FOR 1),'. ') AS full\_name

FROM Client ORDER BY cl\_name;



SELECT \* FROM client\_names;

**

Обновим наше представление, чтобы теперь выводилось не в одном столбце, а в нескольких:

CREATE OR REPLACE VIEW client\_names AS

SELECT

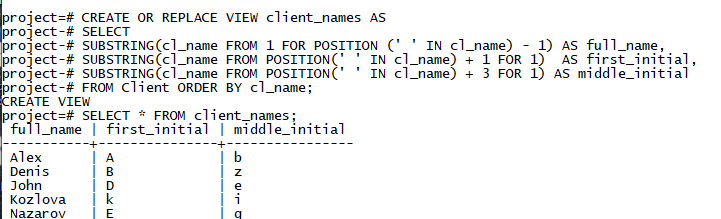
SUBSTRING(cl\_name FROM 1 FOR POSITION (' ' IN cl\_name) - 1) AS full\_name,

SUBSTRING(cl\_name FROM POSITION(' ' IN cl\_name) + 1 FOR 1) AS first\_initial,

SUBSTRING(cl\_name FROM POSITION(' ' IN cl\_name) + 3 FOR 1) AS middle\_initial

FROM Client ORDER BY cl\_name;

SELECT \* FROM client\_names;



Создадим запрос, который будет содержать в себе представление. Нужно вывести задание проекта (proj\_task) клиента и его полное имя из нашего запроса, если проект в еще не сдан.

Для этого нужно сначала обновить представление, добавив в него id клиента.

CREATE OR REPLACE VIEW client\_names AS

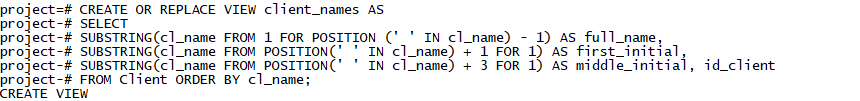
SELECT

SUBSTRING(cl\_name FROM 1 FOR POSITION (' ' IN cl\_name) - 1) AS full\_name,

SUBSTRING(cl\_name FROM POSITION(' ' IN cl\_name) + 1 FOR 1) AS first\_initial,

SUBSTRING(cl\_name FROM POSITION(' ' IN cl\_name) + 3 FOR 1) AS middle\_initial, id\_client

FROM Client ORDER BY cl\_name;

Результат запроса:

SELECT

cn.full\_name,

p.proj\_task

FROM

client\_names cn

JOIN

Project p ON cn.id\_client = p.client

WHERE

p.proj\_status = FALSE;

