ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

(НИУ «БелГУ»)

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

**Отчет по лабораторной работе 5**

**по дисциплине: «Базы данных**

**Тема работы «Представления и функции на языке SQL»**

**“Обслуживание работы конференции”**

студента очного отделения

2 курса 12001801 группы

Капустина Виктора Сергеевича

Проверил(а):

Петров Денис Васильевич

Белгород 2020

**Цель работы:**

1. Приобрести практические навыки выборки данных при помощи языка SQL (оператор SELECT);  
2. Научится создавать представления при помощи языка SQL под управлением СУБД PostgreSQL;  
3. Научиться создавать функции при помощи языка SQL под управлением СУБД PostgreSQL.

**Теоретическая часть.**

Представления (view) – это заранее составленный и хранящийся в БД SQL запрос для выборки данных из одной или нескольких таблиц БД.

Преимущества использования представлений:

1. Упрощение доступа к данным. SQL запрос составлен заранее, уже скомпилирован сервером.

2. Настройка доступа к данным. Каждый пользователь может использовать свой просмотр.

3. Обеспечение независимости от данных. View – защищает пользователей от изменения структуры БД.

4. Защита данных. Просмотры ограничивают доступ к ответственной информации БД.

Существуют модифицируемые (изменяемые) и не модифицируемые.

Считается, что представление является модифицируемым, если оно:

- относится к только одной базовой таблице;

- содержит первичный ключ этой таблицы;

- не имеет никаких полей, которые бы являлись агрегатными функциями;

- не содержит DISTINCT, GROUP BY или HAVING в своем определении;

- не использует подзапросы;

- может быть использовано в другом представлении, но это представление также модифицируемо;

- не использует константы, строки, или выражения среди выбранных полей вывода;

- содержит любые поля основной таблицы, которые имеют ограничение NOT NULL, если другое ограничение по умолчанию не определено.

В PostgreSQL все создаваемые представления по умолчанию являются не модифицируемыми. Чтобы сделать представление модифицируемым, нужно создать правило (RULE), которое обеспечивает выполнение требуемых действий над другими таблицами.

**Функции, написанные на языке SQL**

- SQL-функции - произвольный список операторов SQL, которые возвращают результат последнего запроса в списке. В простом случае будет возвращена только первая строка результата последнего запроса. Если последний запрос вообще не вернёт строки, будет возвращено значение NULL.

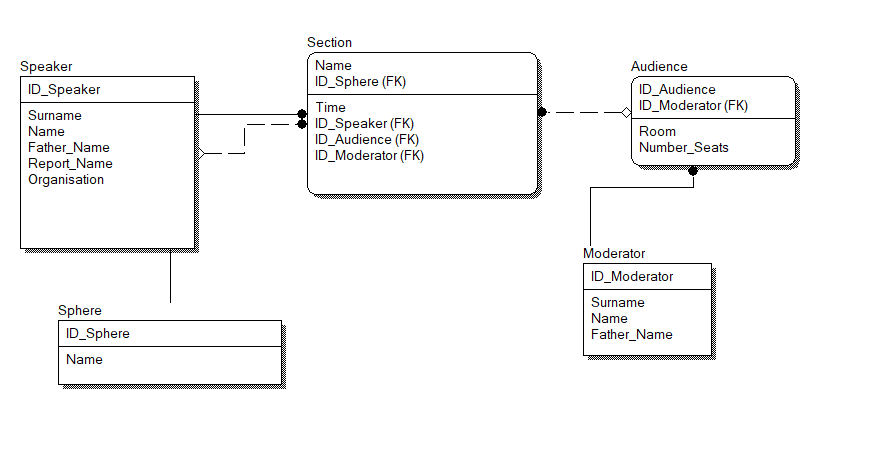
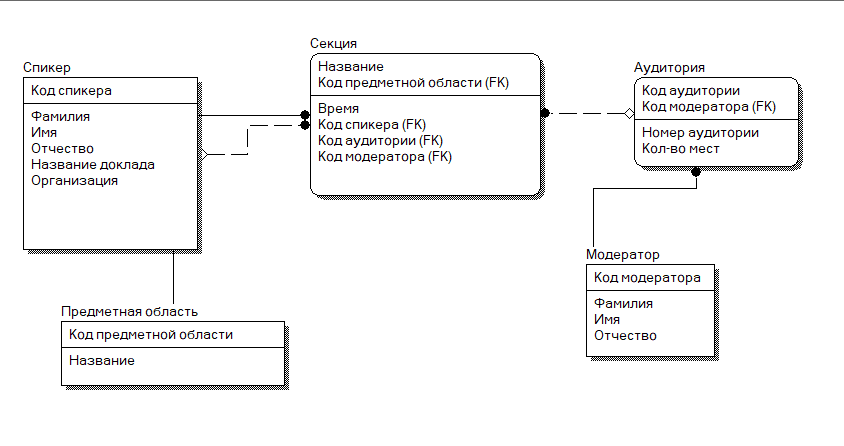
- можно объявить SQL-функцию как возвращающую множество, указав в качестве возвращаемого типа функции SETOF \_тип, либо объявив её с указанием RETURNS TABLE. В этом случае будут возвращены все строки результата последнего запроса

- Тело SQL-функции должно представлять собой список SQL операторов, разделенных точкой с запятой. Точка с запятой после последнего оператора может отсутствовать.

- Если только функция не объявлена как возвращающая void, последним оператором должен быть SELECT, либо INSERT, UPDATE или DELETE с предложением RETURNING

- Синтаксис команды CREATE FUNCTION требует, чтобы тело функции было записано как строковая константа. Обычно для этого удобнее всего заключать строковую константу в доллары.

**Ход работы**

**Физическая ER модель**  
 **Логическая ER модель**

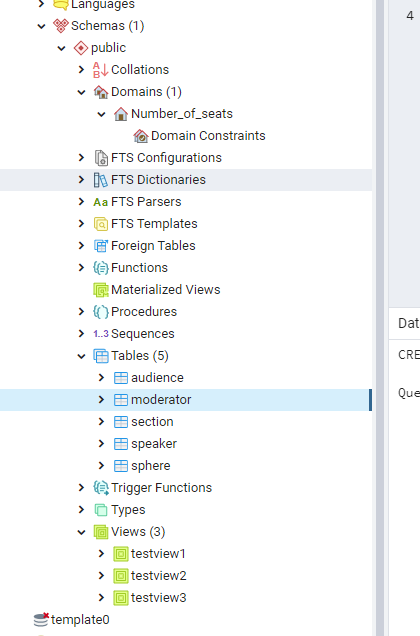
**Представления.**

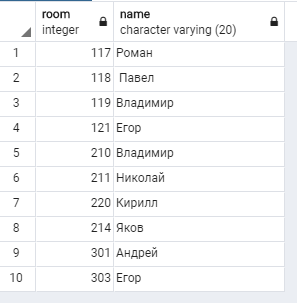
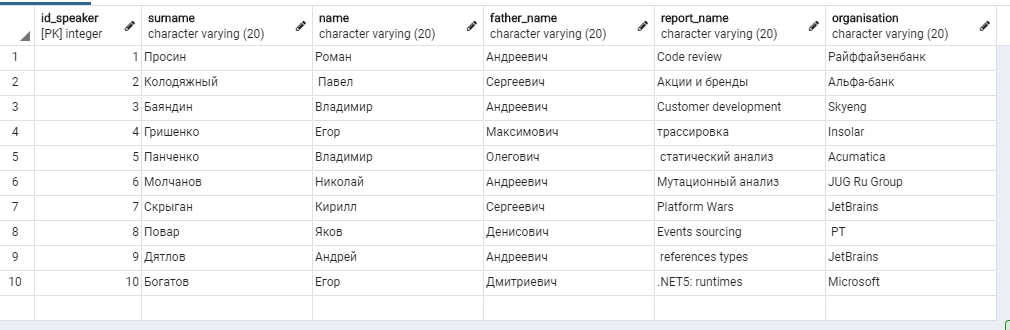
**Создание не модифицируемых представлений.**

Для создания представления используется команда CREATE VIEW

1. CREATE VIEW testview1 AS SELECT moderator.id\_moderator, moderator.Surname, moderator.Name from moderator;
2. CREATE VIEW testview2 AS SELECT Audience.Room, Speaker.Name from Audience INNER JOIN Speaker ON Audience.ID\_Audience = Speaker.ID\_Speaker;
3. CREATE VIEW testview3 AS SELECT \* FROM Speaker ORDER BY speaker.ID\_Speaker;

В первом представлении использован обычный select для вывода имен, фамилий и номера модераторов. Во-втором комбинированный запрос, где мы получим в какой аудитории будет выступать каждый спикер. И в третьем представлении мы получим всю информацию о спикерах. После успешного создания они отобразятся в PgAdmin’e (рис.1)

  
**Рис. 1** Созданные представления

**Результаты представлений:**  
Первого  
  
Второго  
  
Третьего  


**Создание модифицируемых представлений.**

Для создания модифицируемых представлений уже нужно создавать правила.

**1** 1) CREATE VIEW test\_v1 AS SELECT \* FROM Audience;

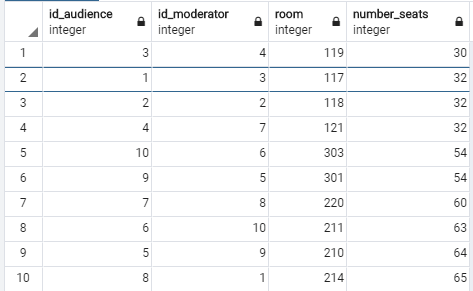
2)CREATE RULE test\_r1 AS ON UPDATE TO test\_v1 DO INSTEAD (

UPDATE Audience SET Number\_Seats = new.Number\_Seats WHERE Number\_Seats = old.Number\_Seats; );

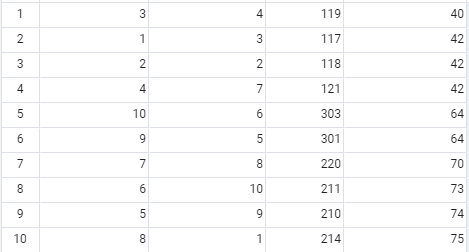
3) UPDATE test\_v1 SET Number\_Seats = Numbers\_Seats + 10;

1 Представление, которое выводит все данные из таблицы Audience.  
2 Создаём правило для представления test\_v1, которое будет обновлять колонку Number\_Seats  
3 Модифицируем, прибавляя к колонке Number\_Seats 10 (увеличиваем кол-во мест на 10).

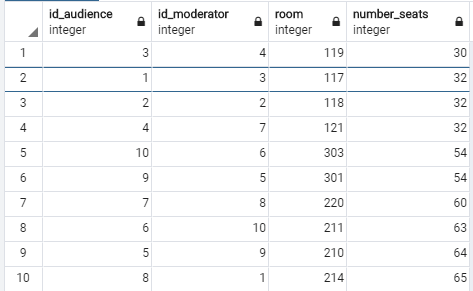
До:

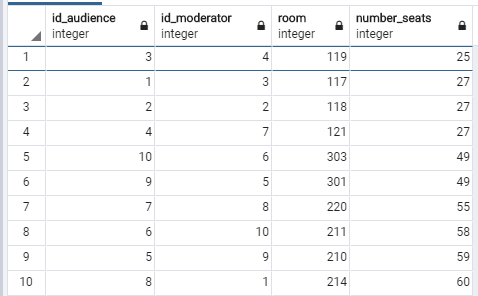


После:

  
  
 Теперь модифицируем, вычтя по 5 из каждого первоначального значения

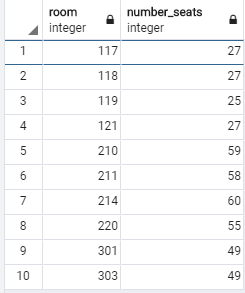
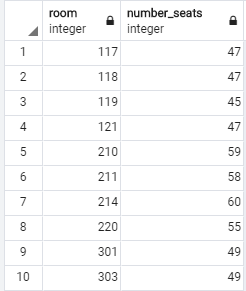
UPDATE test\_v1 SET Number\_Seats = Numbers\_Seats - 5;

.До:  


После:  


**2** 1) CREATE VIEW test\_v2 AS SELECT Room, Number\_Seats FROM Audience ORDER by Room;  
 2)CREATE RULE test\_r2 AS ON UPDATE TO test\_v2 DO INSTEAD (

UPDATE Audience SET Number\_Seats = new.Number\_Seats where Room = old.Room;);  
3)UPDATE test\_v2 SET Number\_Seats = Number\_Seats + 20 where Room < 200;  
1 Представление, которое выводит номера аудитории и кол-во мест в ней  
2 Правило для обновления колонки с кол-вом мест  
3 Модифицируем - Прибавляем 20 мест на первом этаже (Аудитории от 100 до 199)

До:  
  
После:  


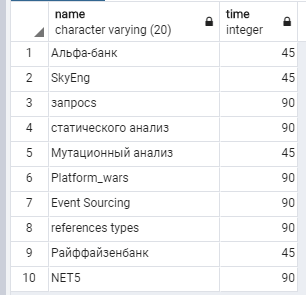
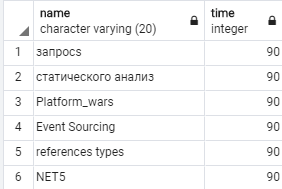
**3** 1)CREATE VIEW test\_v3 AS SELECT Name, Time FROM Section;

2) CREATE RULE test\_r3 AS ON UPDATE TO test\_v3 DO INSTEAD(

SELECT Name, Time FROM Section ;);

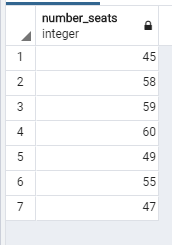
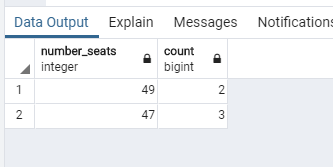
3)SELECT \* FROM test\_v3 where Time > 60;

1 Создадим представление, которое будет выводить ID клиента, его имя и номер.  
2 Создаём правило на вывод.  
3 Модифицируем. Вывод секций, у которых time больше 60 (Идут больше часа)

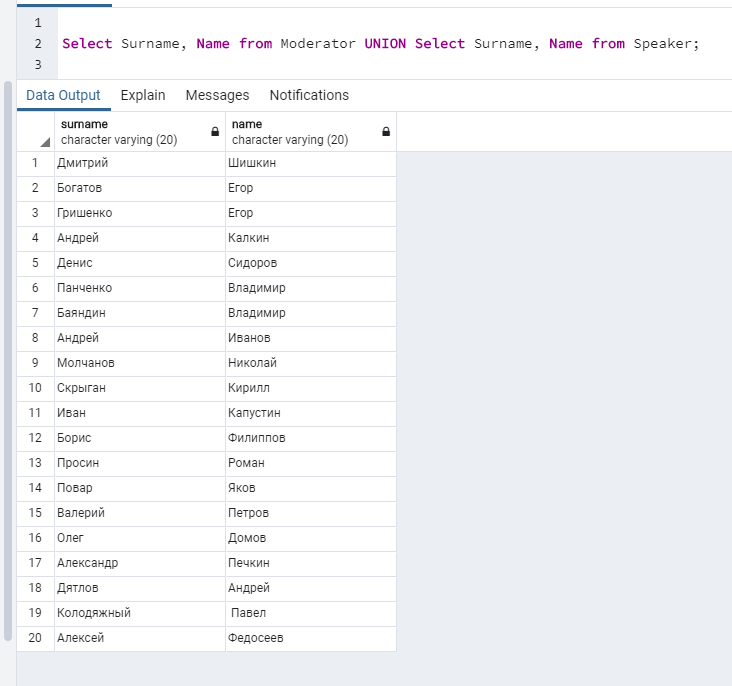
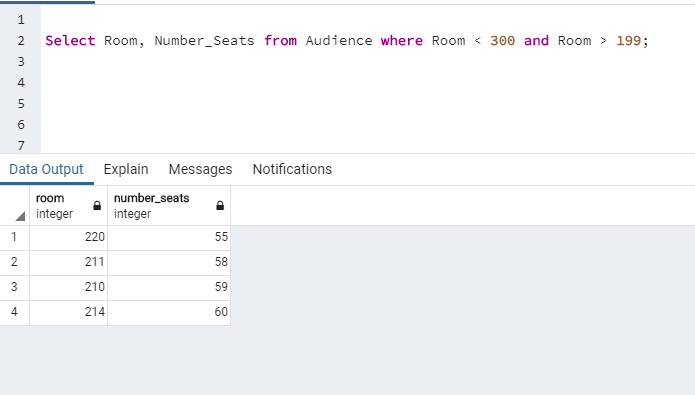
До:  
  
После:  


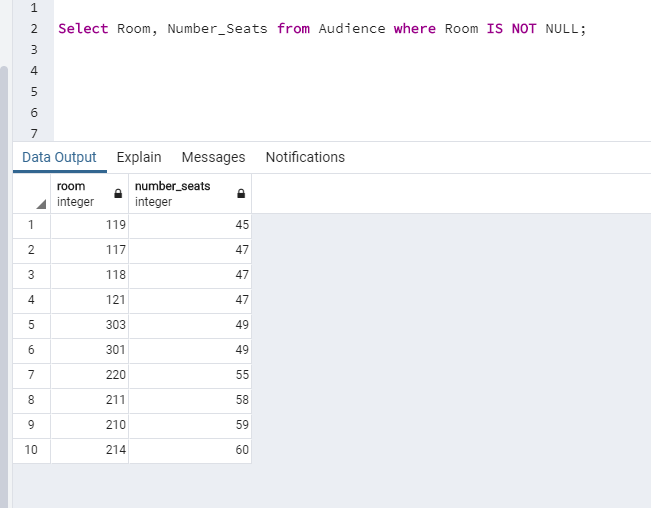
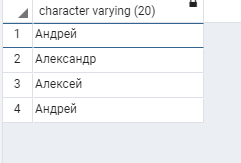
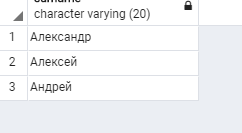
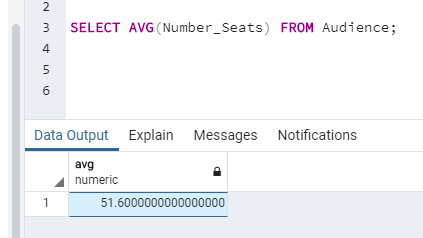
**Ключевые фразы оператора SELECT.**

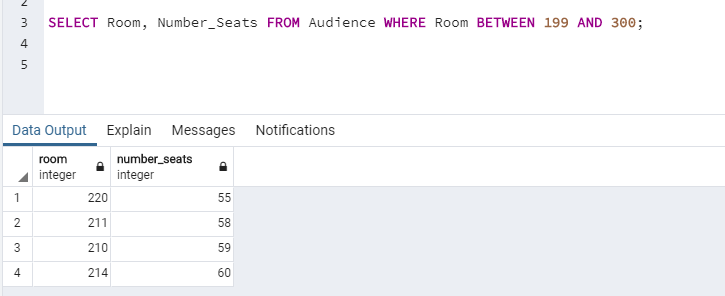
Ключевые фразы использования оператора SELECT (За исключением уже использованных выше)

**Select название\_столбца from название\_таблицы group by критерий –** выводит данные и обеспечивает их группировку по заданному критерию  
  
**Select COUNT() from название\_таблицы group by критерий -**  выводит данные подсчитывая их кол-во. В данном случае выведем значеения, которых больше одного.  


**Select название\_столбца from название\_таблицы union Select название\_столбца from название\_таблицы;**С помощью UNION можно объединить две таблицы в одну. В данном примере выводятся Имена и Фамилии из таблиц Модераторы и Спикеры.

  
  
  
  
  
  
**Select название\_столбца from название\_таблицы условие AND условие -** С помощью AND можно совмещать условия, так в данном примере мы получим номера аудиторий и кол-во мест в них на втором этаже (аудитории 300> и 199<)  


**Select название\_столбца from название\_таблицы WHERE** … **IS NOT NULL -** Выведет все аудитории, где еще есть места  
  
**Select название\_столбца from название\_таблицы LIKE ‘А%’ -** Выведет все что начинается с А. % - означает, что вместо него содержатся символы.  
  
**Select DISTINCT название\_столбца from название\_таблицы условие -** Выведет предыдущий пример, но без повторения результатов  
  
**Select AVG() название\_столбца from название\_таблицы -** Выведет среднее арифметическое кол-ва мест во всех аудиториях.

**Select название\_столбца from название\_табли цы WHERE название\_столбца BETWEEN условие AND условие -** Команда выведет аудитории второго этажа  


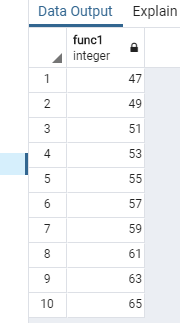
**Функции**

**Func1**

CREATE FUNCTION func1 (Number\_Seats integer)  
RETURNS integer

AS  
$$  
UPDATE Audience set Number\_Seats = Number\_Seats + 2;

SELECT Number\_Seats from Audience;  
$$

LANGUAGE SQL;  
  
  
Результат функции func1

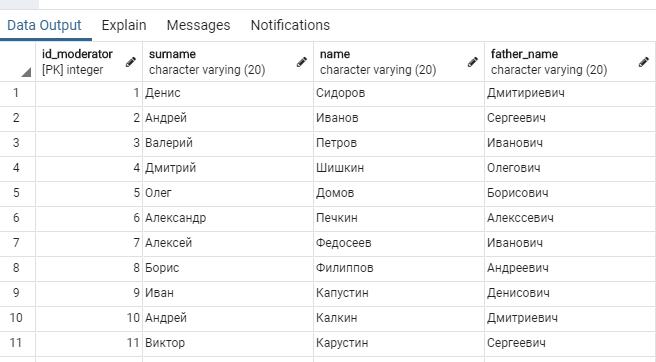
**Func2**

CREATE FUNCTION func2(ID\_Moderator integer, Name character (20), Surname character (20), Father\_Name character (20));  
RETURNS integer

AS  
$$

INSERT INTO PUBLIC.Moderator(ID\_Moderator, Surname, Name, Father\_Name)VALUES(ID\_Moderator, Surname, Name, Father\_Name);

SELECT ID\_Moderator from Moderator;  
$$

LANGUAGE SQL;  
  
В результате работы функции

**Func3**

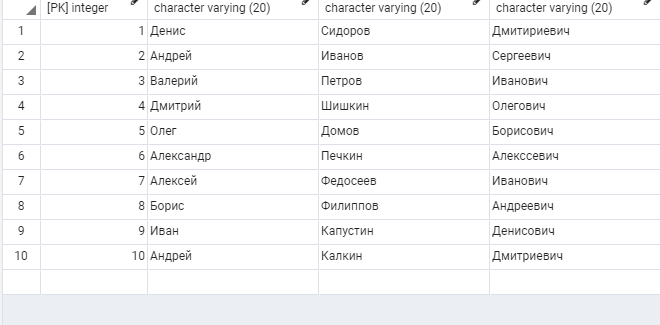
CREATE FUNCTION func3(ID\_Moderator integer);  
RETURNS integer

AS  
$$

DELETE from Moderator

Where ID\_Moderator=11;

SELECT ID\_Moderator from Moderator;  
$$

LANGUAGE SQL;  
  
В результате работы функции