# Тема №5 – Представления

Представление – виртуальный объект в базе данных. Практика показывает, что представления используются, либо слишком часто, либо слишком редко, т.е. очень редко встречаются базы, где бы использование представлений было бы оправданно. Однако представления позволяют достичь следующих целей: сократить кажущуюся сложность базы данных для конечных пользователей, обеспечить доступ пользователей ко всем необходимым данным и вместе с тем запретить доступ к некоторым столбцам, к которым доступ обычных пользователей нежелателен, предусмотреть в базе дополнительные средства индексации, повышающие производительность всей базы. В действительности же представление – это не что иное, как хранимый запрос. Удобство состоит в том, что они обеспечивают выборку данных из одной или нескольких таблиц, обеспечивают согласование данных.

Представление является объектом базы данных, но, в отличие от таблиц, не является физическим хранилищем данных. Его можно представить как хранимое выражение выборки с минимальным набором свойств. Создаваемое представление лишено физической материализации, поэтому указанный запрос будет выполняться при каждом обращении к представлению. Удаление всего представления как объекта никак не повлияет на данные, на основе которых оно построено.

Синтаксическая структура оператора – комбинация рассматриваемых ранее операторов CREATE , а так же оператора SELECT.

CREATE VIEW <название\_представления>

AS

SELECT <описание\_оператора…>

Приведем пример простейшего представления, которое будет выводить все имена, фамилии и отчества всех пользователей:

SELECT

"NAME", "LAST\_NAME", "PATRONYMIC\_NAME"

FROM

users

;

Чтобы создать представление через pgAdmin, в списке сущностей БД найдите пункт «Представления», при помощи ПКМ откройте диалоговое окно и выберите «Создать...» -> «Представление» (рисунок ччч).

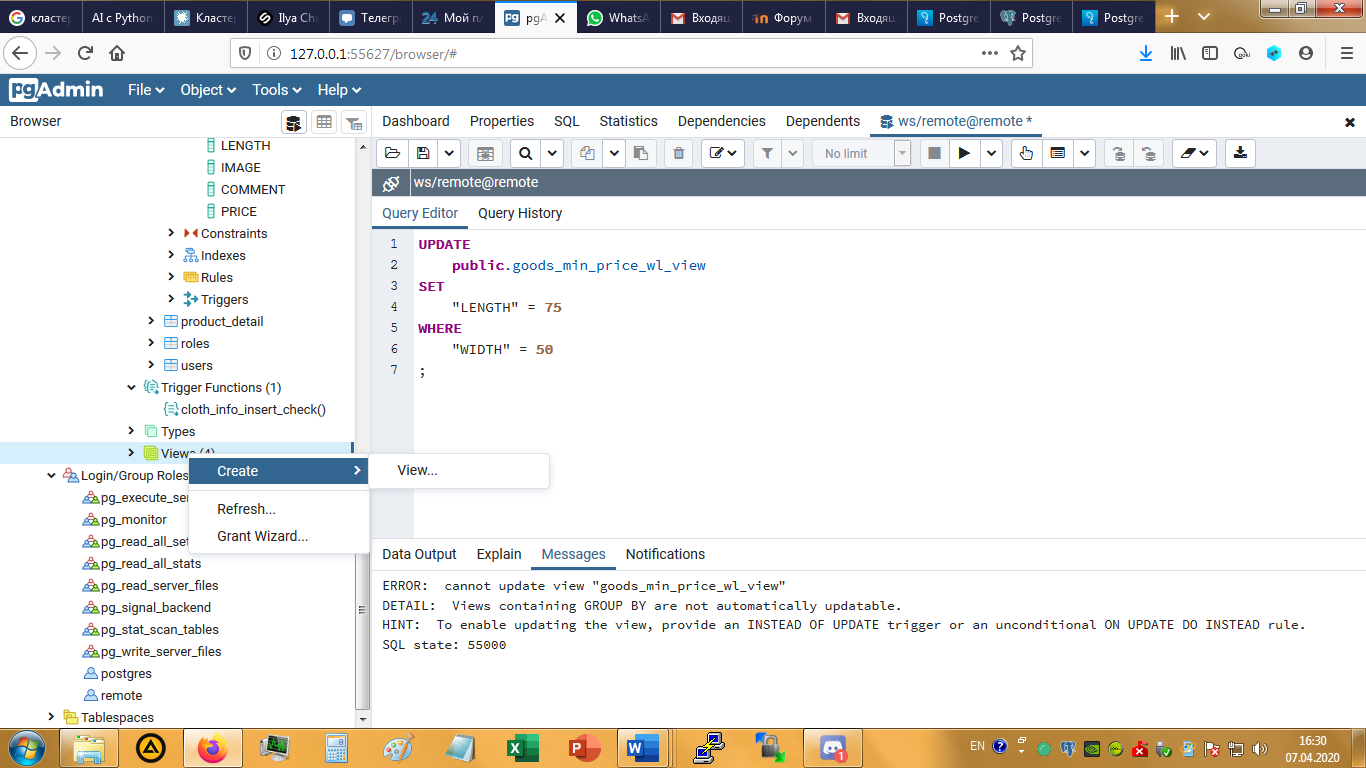


Рисунок ччч – Пункт меню «Добавить представление»

В открывшемся диалоговом окне введём название сущности, как показано на рисунке ппп. Пользуясь возможностью, покажем создание другого представления – выборка товаров с заполненной ценой.

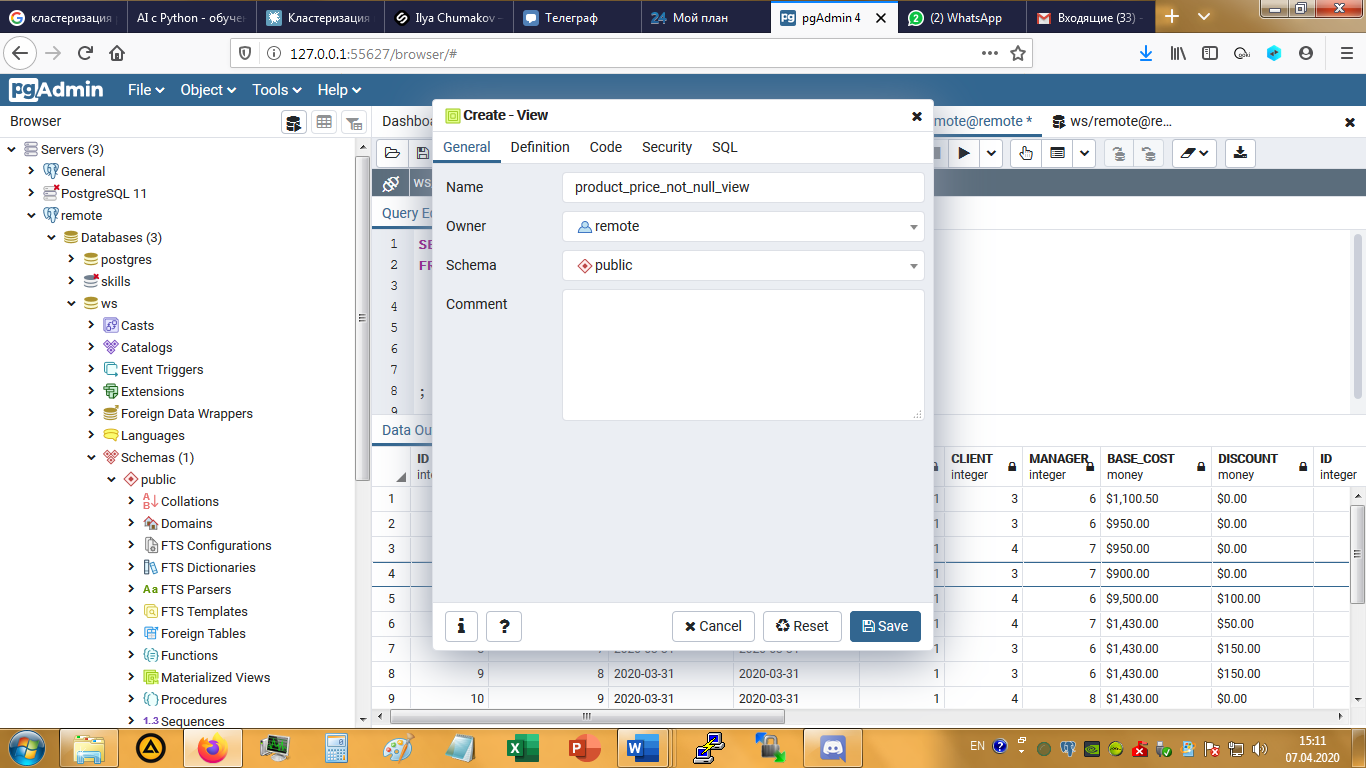


Рисунок ппп – Основные настройки представления

Затем на вкладке Code введём код выборки.

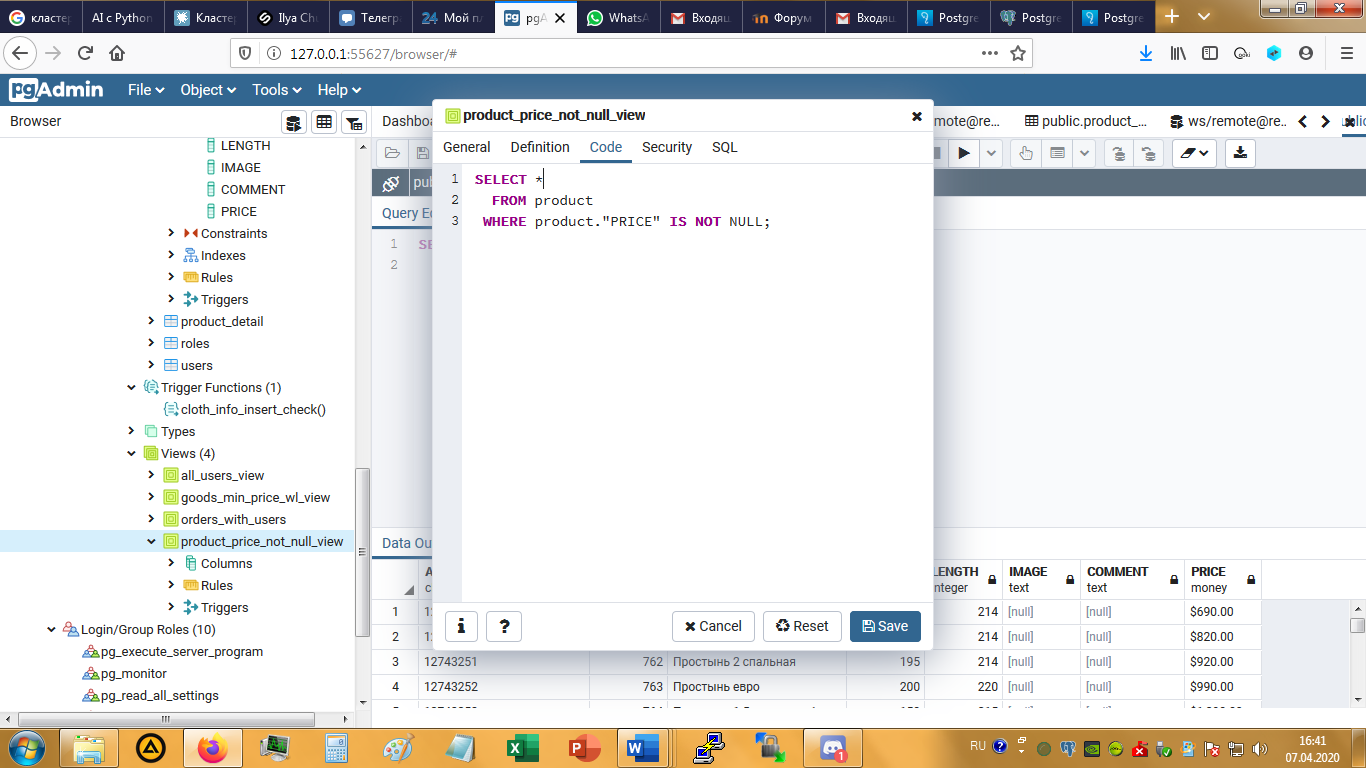


Рисунок шшш – Ввод кода выборки представления

И на вкладке Security зададим настройки прав доступа к различным операциям для различных групп и отдельных пользователей.

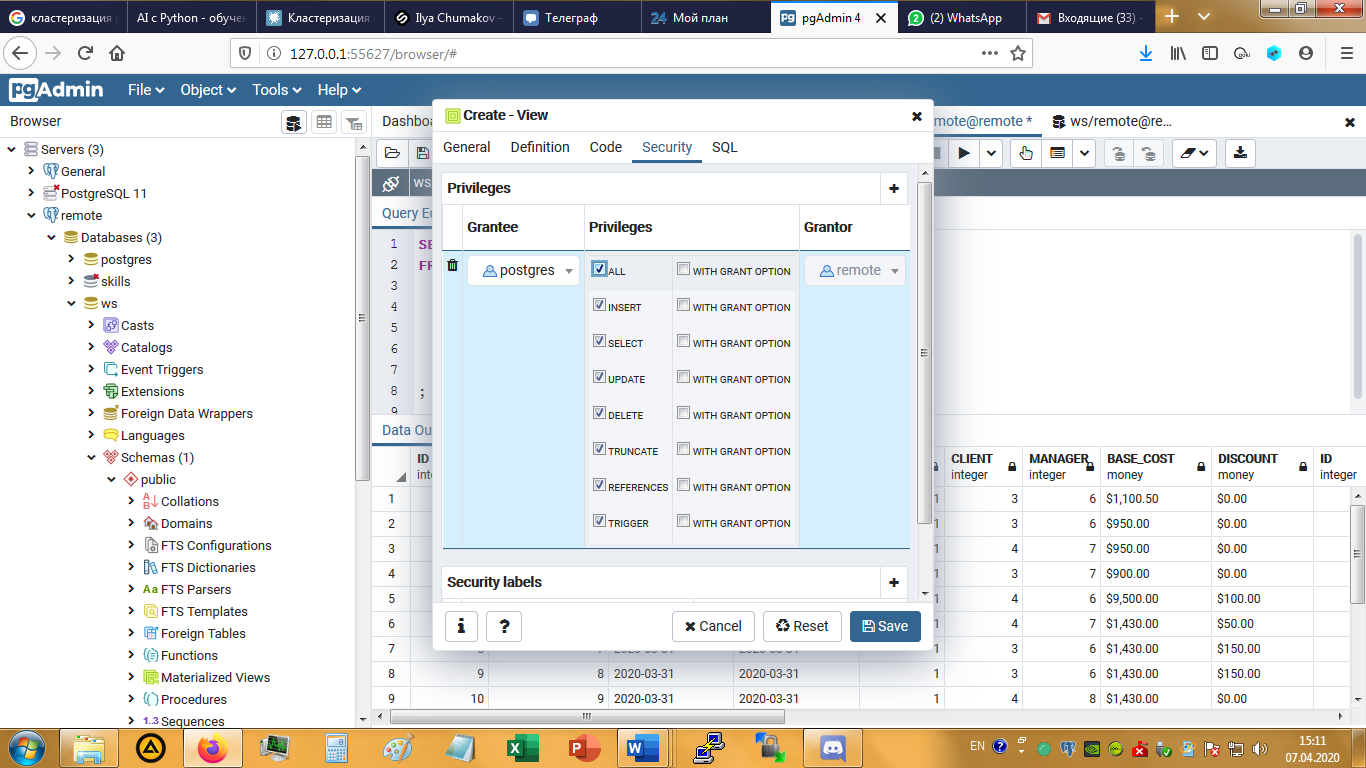


Рисунок ззз – Настройка прав доступа

После этого можно сохранить представление.

Когда представление создано, проверим полученный результат, путём выборки всей информации из созданного представления:

SELECT \* FROM public.all\_users\_view

Результат представлен на рисунке ццц.

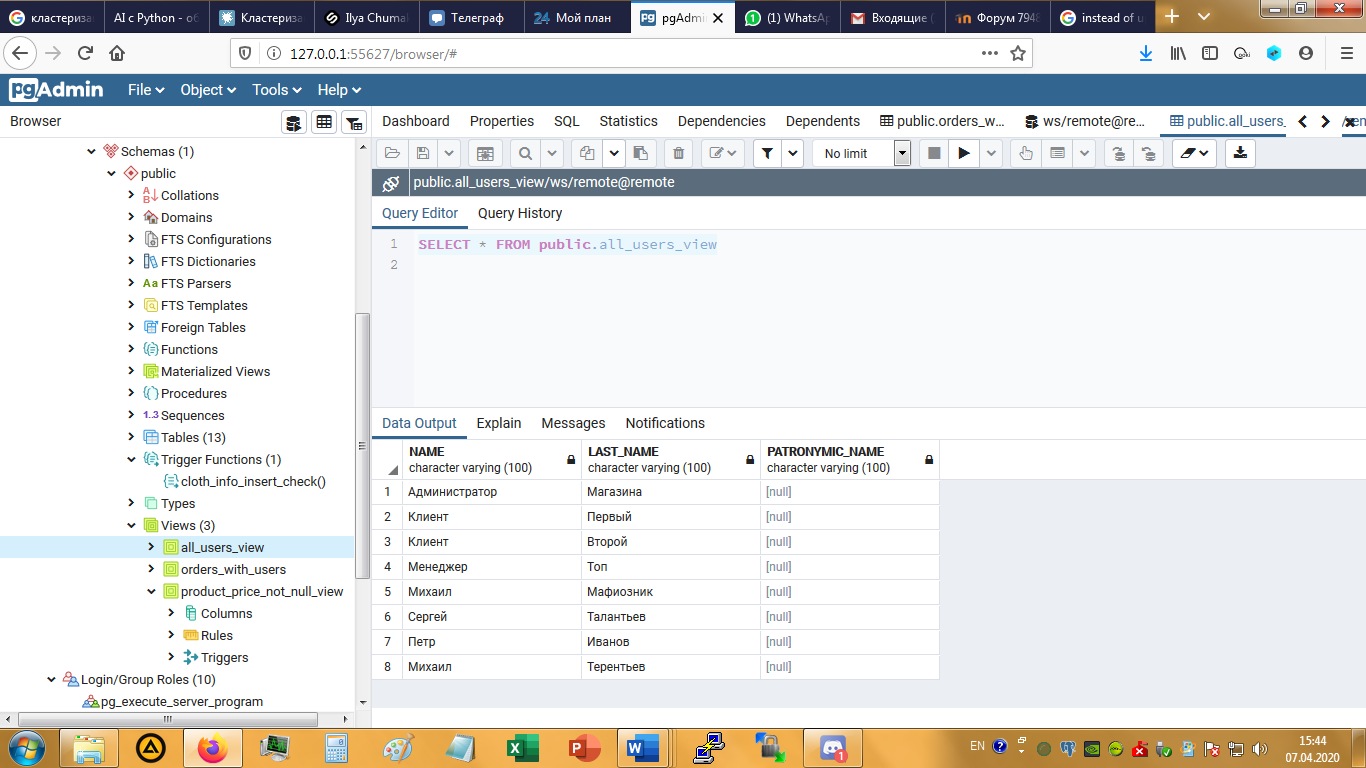


Рисунок ццц – Результат работы представления

Для представления, созданного на инструкции:

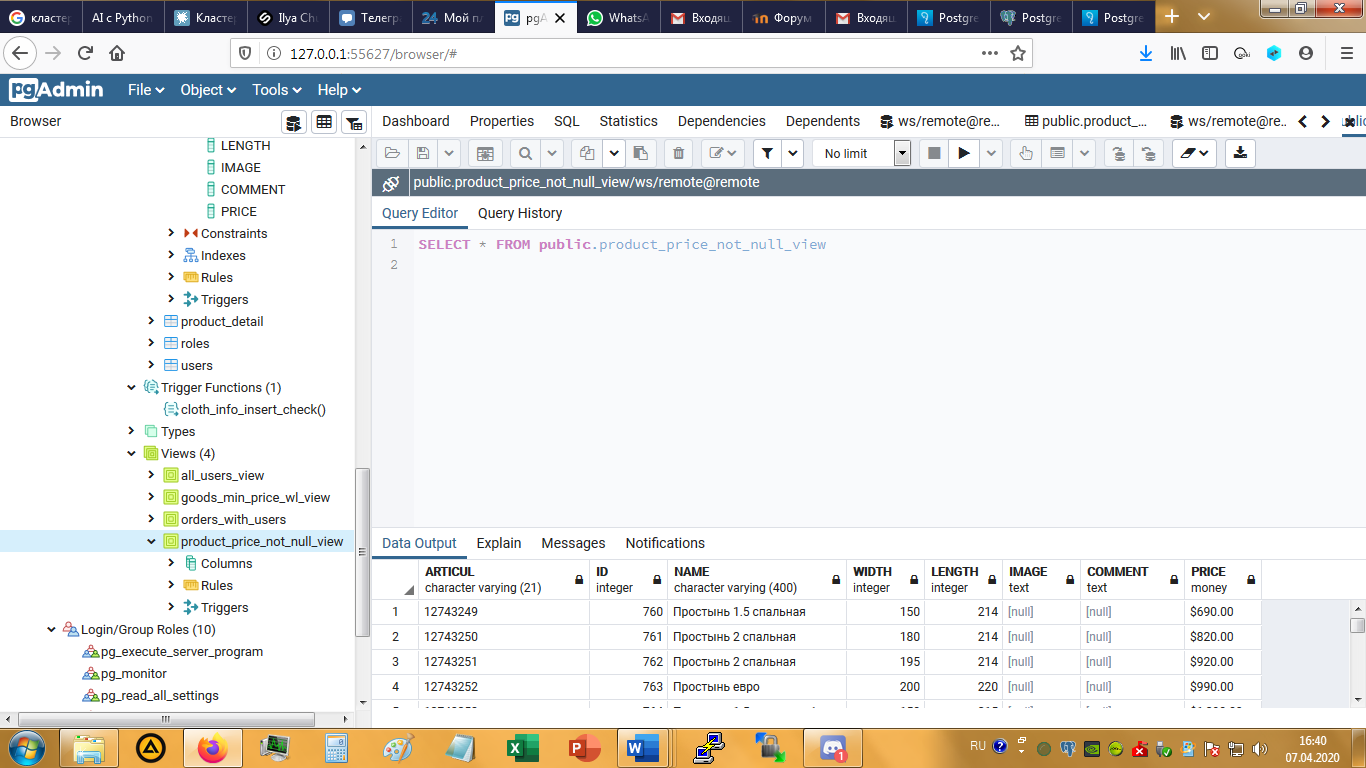


Рисунок ььь – Результат работы представления

В запросах участвующих в создании представлений допускается использование любых конструкций, таких как WHEREи др. У пользователя, создающего представление, должны быть разрешения на выборку из соответствующих таблиц. Можно создать представление на основе других представлений либо представлений и таблиц. Но есть ограничения. Нельзя использовать в выборке: предложения ORDER BY, COMPUTE и COMPUTE BY; предложение INTO; временную таблицу. Кроме того, не следует использовать внешние объединения при создании представлений. Представление может ссылаться максимум на 250 столбцов. Команда CREATE VIEW не может использоваться совместно с другими командами SQL Server в одном пакете команд. Нельзя создавать представления внутри хранимых процедур. Оператор UNION невозможно использовать при создании представлений.

С помощью представлений пользователи могут иметь доступ и изменять только те данные, которые они видят. Остаток базы данных для них недоступен и невидим. К примеру, некоторым пользователям вы решили не давать доступа к столбцам role, login, salt и password в таблице users. Вы можете создать представление (назовите его all\_users\_view), в котором будут выбираться все столбцы, кроме указанных выше, а затем дать пользователям полное разрешение на операции с этим представлением. И только избранным пользователям вы даете доступ к настоящей таблице. Это будет выглядеть следующим образом:

CREATE OR REPLACE VIEW public.all\_users\_view

AS

SELECT users."NAME",

users."LAST\_NAME",

users."PATRONYMIC\_NAME"

FROM users;

ALTER TABLE public.all\_users\_view

OWNER TO remote;

GRANT ALL ON TABLE public.all\_users\_view TO remote;

GRANT TRIGGER ON TABLE public.all\_users\_view TO postgres;

GRANT INSERT, SELECT, UPDATE, DELETE ON TABLE public.all\_users\_view TO postgres WITH GRANT OPTION;

Представления удобны тем, что их можно создавать в любой момент, выдавая разрешение с помощью команды GRANT или отменяя его с помощью REVOKE. Представления могут модифицировать данные в исходных таблицах, при этом помните, что никаких промежуточных таблиц нет – изменения сразу попадают в исходные таблицы.

Изменения, которые производятся с помощью представлений, могут воздействовать только на один объект. Пусть представление создано на основе двух таблиц, order и user.

SELECT

"order"."ID",

"order"."DATE\_CREATE",

users."NAME",

users."LAST\_NAME"

FROM

"order"

JOIN users ON users."ID" = "order"."CLIENT"

;

Из представлений можно производить выборку, как из обычной таблицы. Но к ним нельзя применять операторы INSERT, UPDATE и DELETE, кроме случаев когда представление является простым.Однако это можно обойти при помощи триггеров, речь о которых пойдёт в последующих темах. Для внесения изменений в свойства существующих представлений можно использовать ALTER VIEW - для редактирования; CREATE VIEW – для создания новых представлений; DROP VIEW – для удаления представлений.

## Изменяемые представления

Простые представления становятся изменяемыми автоматически: система позволит выполнять команды INSERT, UPDATE и DELETE с таким представлением так же, как и с обычной таблицей. Представление будет автоматически изменяемым, если оно удовлетворяют одновременно всем следующим условиям:

* Список FROM в запросе, определяющем представлении, должен содержать ровно один элемент, и это должна быть таблица или другое изменяемое представление.
* Определение представления не должно содержать предложения WITH, DISTINCT, GROUP BY, HAVING, LIMIT и OFFSET на верхнем уровне запроса.
* Определение представления не должно содержать операции с множествами (UNION, INTERSECT и EXCEPT) на верхнем уровне запроса.
* Список выборки в запросе не должен содержать агрегатные и оконные функции, а также функции, возвращающие множества.

Автоматически обновляемое представление может содержать как изменяемые, так и не изменяемые столбцы. Столбец будет изменяемым, если это простая ссылка на изменяемый столбец нижележащего базового отношения; в противном случае этот столбец будет доступен только для чтения, и если команда INSERT или UPDATE попытается записать значение в него, возникнет ошибка.

Если представление автоматически изменяемое, система будет преобразовывать обращающиеся к нему операторы INSERT, UPDATE и DELETE в соответствующие операторы, обращающиеся к нижележащему базовому отношению. При этом в полной мере поддерживаются операторы INSERT с предложением ON CONFLICT UPDATE.

Если автоматически изменяемое представление содержит условие WHERE, это условие ограничивает набор строк, которые могут быть изменены командой UPDATE и удалены командой DELETE в этом представлении. Однако UPDATE может изменить строку так, что она больше не будет соответствовать условию WHERE и, как следствие, больше не будет видна через представление. Команда INSERT подобным образом может вставить в базовое отношение строки, которые не удовлетворят условию WHERE и поэтому не будут видны через представление (ON CONFLICT UPDATE может подобным образом воздействовать на существующую строку, не видимую через представление). Чтобы запретить командам INSERT и UPDATE создавать такие строки, которые не видны через представление, можно воспользоваться указанием CHECK OPTION.

Если автоматически изменяемое представление имеет свойство security\_barrier (барьер безопасности), то все условия WHERE этого представления (и все условия с герметичными операторами (LEAKPROOF)) будут всегда вычисляться перед условиями, добавленными пользователем представления.

Более сложные представления, не удовлетворяющие этим условиям, по умолчанию доступны только для чтения: система не позволит выполнить операции добавления, изменения или удаления строк в таком представлении. Создать эффект изменяемого представления для них можно, определив триггеры INSTEAD OF, которые будут преобразовывать запросы на изменение данных в соответствующие действия с другими таблицами. За дополнительными сведениями обратитесь к CREATE TRIGGER. Так же есть возможность создавать правила (CREATE RULE), но на практике триггеры проще для понимания и применения.

Учтите, что пользователь, выполняющий операции добавления, изменения или удаления данных в представлении, должен иметь соответствующие права для этого представления. Кроме того, владелец представления должен иметь сопутствующие права в нижележащих базовых отношениях, хотя пользователь, собственно выполняющий эти операции, может этих прав не иметь.

## Материализованные представления

Команда CREATE MATERIALIZED VIEW определяет материализованное представление запроса. Заданный запрос выполняется и наполняет представление в момент вызова команды (если только не указано WITH NO DATA). Обновить представление позже можно, выполнив REFRESH MATERIALIZED VIEW.

Команда CREATE MATERIALIZED VIEW подобна CREATE TABLE AS, за исключением того, что она запоминает запрос, порождающий представление, так что это представление можно позже обновить по требованию. Материализованные представления сходны с таблицами во многом, но не во всём; например, не поддерживаются временные материализованные представления и автоматическая генерация OID.