# ТЕМА 13. ПРЕДСТАВЛЕНИЯ

**Представления** – этот статические запросы, которые можно использовать как виртуальную интерпретацию таблиц БД. Столбцами представления могут быть столбцы таблиц, агрегатные значения, константы и выражения.

Представления используются:

1) для маскирования деталей реализации (скрыть от пользователя большое количество соединений);

2) как средство формирования отчетов в реальном времени:

– предотвращение нерегламентированных запросов, приводящих к увеличению нагрузок на сервер;

– позволяют избежать блокировок;

3) как механизм безопасности:

– назначение таких прав доступа пользователю, когда он может использовать представление, но не основные таблицы;

– способность делить структуру таблицы, ограничивая строки или столбцы, видимые пользователю.

4) для реализации дополнительных индексов в БД с целью увеличения быстродействия запросов.

**Синтаксис**

```
CREATE [ OR REPLACE ] [ TEMP | TEMPORARY ] [ RECURSIVE ] VIEW имя [ ( имя_столбца [, ...] ) ]
    [ WITH ( имя_параметра_представления [= значение_параметра_представления] [, ... ] ) ]
    AS запрос
```

**Описание**

CREATE VIEW создаёт представление запроса. Создаваемое представление лишено физической материализации, поэтому указанный запрос будет выполняться при каждом обращении к представлению.

Команда CREATE OR REPLACE VIEW действует подобным образом, но если представление с этим именем уже существует, оно заменяется. Новый запрос должен выдавать те же столбцы, что выдавал запрос, ранее определённый для этого представления (то есть, столбцы с такими же именами должны иметь те же типы данных и следовать в том же порядке), но может добавить несколько новых столбцов в конце списка. Вычисления, в результате которых формируются столбцы представления, могут быть совершенно другими.

**Параметры**

**TEMPORARY или TEMP**

С таким указанием представление создаётся как временное. Временные представления автоматически удаляются в конце сеанса. Существующее постоянное представление с тем же именем не будет видно в текущем сеансе, пока существует временное, однако к нему можно обратиться, дополнив имя указанием схемы.

Если в определении представления задействованы временные таблицы, представление так же создаётся как временное (вне зависимости от присутствия явного указания TEMPORARY).

**RECURSIVE**

Создаёт рекурсивное представление. Синтаксис

```
CREATE RECURSIVE VIEW [ схема . ] имя (имена_столбцов) AS SELECT ...;
```

равнозначен

```
CREATE VIEW [ схема . ] имя AS WITH RECURSIVE имя (имена_столбцов) AS (SELECT ...)
SELECT имена_столбцов FROM имя;
```

Для рекурсивного представления обязательно должен задаваться список с именами столбцов.

***имя*** Имя (возможно, дополненное схемой) создаваемого представления.

***имя_столбца*** Необязательный список имён, назначаемых столбцам представления. Если отсутствует, имена столбцов формируются из результатов запроса.

**WITH [ CASCADED | LOCAL ] CHECK OPTION**

Это указание управляет поведением автоматически изменяемых представлений. Если оно присутствует, при выполнении операций INSERT и UPDATE с этим представлением будет проверяться, удовлетворяют ли новые строки условию, определяющему представление (то есть, проверяется, будут ли новые строки видны через это представление). Если они не удовлетворяют условию, операция не будет выполнена. Если указание CHECK OPTION отсутствует, команды INSERT и UPDATE смогут создавать в этом представлении строки, которые не будут видны в нём. Поддерживаются следующие варианты проверки:

**LOCAL**

Новые строки проверяются только по условиям, определённым непосредственно в самом представлении. Любые условия, определённые в нижележащих базовых представлениях, не проверяются (если только в них нет указания CHECK OPTION).

**CASCADED**

Новые строки проверяются по условиям данного представления и всех нижележащих базовых. Если указано CHECK OPTION, а LOCAL и CASCADEDопущено, подразумевается указание CASCADED.

Указание CHECK OPTION нельзя использовать с рекурсивными представлениями.

Заметьте, что CHECK OPTION поддерживается только для автоматически изменяемых представлений, не имеющих триггеров INSTEAD OF и правил INSTEAD. Если автоматически изменяемое представление определено поверх базового представления с триггерами INSTEAD OF, то для проверки ограничений автоматически изменяемого представления можно применить указание LOCAL CHECK OPTION, хотя условия базового представления с триггерами INSTEAD OF при этом проверяться не будут (каскадная проверка не будет спускаться к представлению, модифицируемому триггером, и любые параметры проверки, определённые для такого представления, будут просто игнорироваться). Если для представления или любого из его базовых отношений определено правило INSTEAD, приводящее к перезаписи команды INSERT или UPDATE, в перезаписанном запросе все параметры проверки будут игнорироваться, в том числе проверки автоматически изменяемых представлений, определённых поверх отношений с правилом INSTEAD.

Для удаления представлений применяется оператор DROP VIEW.

Будьте аккуратны в определении представления, чтобы получить желаемые имена и типы столбцов. Например, такая команда:

```
CREATE VIEW vista AS SELECT 'Hello World';
```

создаст представление с двумя недостатками: именем столбца по умолчанию будет ?column?, а типом данных — unknown (неизвестный). Если вы хотите получить в представлении строковую константу, лучше сделать так:

```
CREATE VIEW vista AS SELECT text 'Hello World' AS hello;
```

**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.

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.

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.

**Примеры**

Создание представления, содержащего все комедийные фильмы:

```
CREATE VIEW comedies AS
    SELECT *
    FROM films
    WHERE kind = 'Comedy';
```

Эта команда создаст представление со столбцами, которые содержались в таблице film в момент выполнения команды. Хотя при создании представления было указано *, столбцы, добавляемые в таблицу позже, частью представления не будут.

Создание представления с указанием LOCAL CHECK OPTION:

```
CREATE VIEW universal_comedies AS
    SELECT *
    FROM comedies
    WHERE classification = 'U'
    WITH LOCAL CHECK OPTION;
```

Эта команда создаст представление на базе представления comedies, выдающее только комедии (kind = 'Comedy') универсальной возрастной категории classification = 'U'. Любая попытка выполнить в представлении INSERT или UPDATE со строкой, не удовлетворяющей условию classification = 'U', будет отвергнута, но ограничение по полю kind (тип фильма) проверяться не будет.

Создание представления с указанием CASCADED CHECK OPTION:

```
CREATE VIEW pg_comedies AS
    SELECT *
    FROM comedies
    WHERE classification = 'PG'
    WITH CASCADED CHECK OPTION;
```

Это представление будет проверять, удовлетворяют ли новые строки обоим условиям: по столбцу kind и по столбцу classification.

Создание представления с изменяемыми и неизменяемыми столбцами:

```
CREATE VIEW comedies AS
   SELECT f.*,
          country_code_to_name(f.country_code) AS country,
          (SELECT avg(r.rating)
           FROM user_ratings r
           WHERE r.film_id = f.id) AS avg_rating
   FROM films f
   WHERE f.kind = 'Comedy';
```

Это представление будет поддерживать операции INSERT, UPDATE и DELETE. Изменяемыми будут все столбцы из таблицы films, тогда как вычисляемые столбцы country и avg_rating будут доступны только для чтения.

Создание рекурсивного представления, содержащего числа от 1 до 100:

```
CREATE RECURSIVE VIEW public.nums_1_100 (n) AS
   VALUES (1)
UNION ALL
   SELECT n+1 FROM nums_1_100 WHERE n < 100;
```

Заметьте, что несмотря на то, что имя рекурсивного представления дополнено схемой в этой команде CREATE, внутренняя ссылка представления на себя же схемой не дополняется. Это связано с тем, что имя неявно создаваемого CTE не может дополняться схемой.

## Материализованные представления

Материализованные представления в Postgres Pro основаны на системе правил, как и представления, но их содержимое сохраняется как таблица. Основное отличие между:

```
CREATE MATERIALIZED VIEW mymatview AS SELECT * FROM mytab;
```

и этой командой:

```
CREATE TABLE mymatview AS SELECT * FROM mytab;
```

состоит в том, что материализованное представление впоследствии нельзя будет изменить непосредственно, а запрос, создающий материализованное представление, сохраняется точно так же, как запрос представления, и получить актуальные данные в материализованном представлении можно так:

```
REFRESH MATERIALIZED VIEW mymatview;
```

Информация о материализованном представлении в системных каталогах Postgres Pro ничем не отличается от информации о таблице или представлении. Поэтому для анализатора запроса материализованное представление является просто отношением, как таблица или представление. Когда запрос обращается к материализованному представлению, данные возвращаются непосредственно из него, как из таблицы; правило применяется, только чтобы его наполнить.

Хотя обращение к данным в материализованном представлении часто выполняется гораздо быстрее, чем обращение к нижележащим таблицам напрямую или через представление, данные в нём не всегда актуальные (но иногда это вполне приемлемо).