Создание представлений VIEWS

Для чего нужны представления?

- При работе с SQL часто возникают ситуации, когда требуется использовать запросы повторно
  - Это особенно актуально при работе с большими или сложными запросами - вы можете запросить представление, основанное на сложном запросе, с помощью простого SELECT оператора
  - Передача чрезмерно больших запросов по сети на ваш сервер PostgreSQL для часто выполняемых подпрограмм может оказаться крайне неэффективной
- По умолчанию, представление лишено физической материализации, поэтому указанный **запрос будет выполняться при каждом обращении** к представлению
- С точки зрения администратора, представления позволяют обеспечить уровень безопасности для данных в БД

Создание представления (View)

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

- **Представления** – это сохраненные в БД именованные запросы
  - Вы можете ссылаться на представления в операторе SELECT, как на таблицы (виртуальные)
  - Столбцы результирующей выборки должны иметь уникальные имена и желаемые типы
- Представления определяются с помощью оператора SELECT
- Если в определении представления задействованы временные таблицы, представление так же создаётся как временное (вне зависимости от присутствия явного указания **TEMPORARY**
- Если при создании представления в предложении **SELECT** было указано *, столбцы, добавляемые в таблицу позже, **частью представления не будут**!

Создание представления (View)

```
CREATE OR REPLACE VIEW public."OrderValues"
AS
  SELECT o.orderid,
    o.custid,
    o.empid,
    o.shipperid,
    o.orderdate,
    o.requireddate,
    o.shippeddate,
    sum(od.qty) AS qty,
    sum(od.qty * od.unitprice *
        (1::numeric - od.discount)::double precision)::numeric(12,2) AS val
   FROM "Sales"."Orders" o
     JOIN "Sales"."OrderDetails" od ON o.orderid = od.orderid
  GROUP BY o.orderid, o.custid, o.empid, o.shipperid,
           o.orderdate, o.requireddate, o.shippeddate;
```

Безопасность

- Владелец представления должен иметь соответствующие права на нижележащие базовые отношения (таблицы и/или представления)
- Для пользователя представления доступ к таблицам, используемым в SELECT-запросе представления, определяется правами владельца представления
  - это позволяет организовать безопасный, но ограниченный доступ к нижележащим таблицам
  - если в представлении используются пользовательские функции пользователь представления должен иметь все права, необходимые для вызова всех функций, задействованных в представлении

Изменяемые представления

- Если представление удовлетворяет следующим требованиям оно может использоваться для модификации данных в исходной таблице:
  - Запрос должен ссылаться на **одну** базовую таблицу (или изменяемое преставление) в предложении **FROM**
  - 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
- Пользователь, выполняющий операции добавления, изменения или удаления данных через представление, должен иметь соответствующие права для этого представления

Изменяемые представления с WHERE

- 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
- **Внимание**:
  - **UPDATE** может изменить строку так, что она больше не будет соответствовать условию WHERE и больше не будет видна через представление
  - **INSERT** 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

Поддержка изменений исходных данных

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

- Управляет поведением автоматически изменяемых представлений, не имеющих триггеров **INSTEAD OF:**
  - Если указание **CHECK OPTION** отсутствует, команды INSERT и UPDATE смогут создавать в этом представлении строки, которые нарушают условия фильтрации, заданные в представлении
- **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** (по умолчанию)
  - Новые строки проверяются по условиям данного представления и всех нижележащих базовых

Параметры представления

```
WITH (имя_параметра [= значение_параметра] [, ... ] )
```

- **check_option (enum)**
  - может принимать значение ***local*** (локально) или ***cascaded*** (каскадно)
  - равнозначен указанию **WITH [CASCADED | LOCAL] CHECK OPTION**
  - изменить этот параметр у существующего представления с помощью ALTER VIEW нельзя!
- **security_barrier (boolean)**
  - следует использовать, если представление должно обеспечивать защиту на уровне строк

Материализованные представления

- Материализованные представления PostgreSQL позволяют физически сохранять результат запроса
  - Они кэшируют результат сложного и затратного запроса и позволяют периодически обновлять этот результат (**WITH DATA**)
- Материализованные представления полезны во многих случаях, когда требуется быстрый доступ к данным, поэтому они часто используются в хранилищах данных и приложениях бизнес-аналитики

```
CREATE MATERIALIZED VIEW view_name
AS
query
WITH [NO] DATA;
```

Загрузка данных в материализованное представление

- При создании представления с параметром **`WITH NO DATA`** представление помечается как **нечитаемое**
  - Вы не можете запрашивать данные из представления, пока не загрузите данные!
- Для загрузки данных в материализованное представление используется оператор **REFRESH MATERIALIZED VIEW**

```
REFRESH MATERIALIZED VIEW view_name;
```

Изменение представления

- Для изменения запроса, определяющего представление, используется оператор **CREATE OR REPLACE VIEW**:

```
CREATE OR REPLACE VIEW view_name
AS
query
```

- Измененный запрос должен генерировать те же столбцы, которые были созданы при создании представления
  - PostgreSQL не поддерживает удаление существующего столбца в представлении
  - Новые столбцы должны иметь те же имена, те же типы данных и в том же порядке, в котором они были указаны при создании представления
  - PostgreSQL позволяет добавлять дополнительные столбцы в конец списка столбцов

Изменение определения представления

- Для изменения имени представления, имени столбца, владельца или схемы представления используется оператор **ALTER VIEW**:

```
ALTER VIEW [IF EXISTS] имя ALTER [COLUMN] имя_столбца SET DEFAULT выражение
ALTER VIEW [IF EXISTS] имя ALTER [COLUMN] имя_столбца DROP DEFAULT
ALTER VIEW [IF EXISTS] имя OWNER TO {новый_владелец|CURRENT_USER|SESSION_USER}
ALTER VIEW [IF EXISTS] имя RENAME [ COLUMN ] имя_столбца TO новое_имя_столбца
ALTER VIEW [IF EXISTS] имя RENAME TO новое_имя
ALTER VIEW [IF EXISTS] имя SET SCHEMA новая_схема
```

Изменение параметров представления

```
ALTER VIEW [IF EXISTS] имя SET (параметр [= значение_параметра] [, ...])
ALTER VIEW [IF EXISTS] имя RESET (имя_параметра [, ... ])
```

- В настоящее время поддерживаются следующие параметры:
  - **check_option (enum)** - изменяет параметр проверки представления: local (локальная) или cascaded (каскадная)
  - **security_barrier (boolean)** - изменяет свойство представления, включающее барьер безопасности (true или false)

Удаление представления

- Чтобы удалить существующее представление в PostgreSQL используется оператор DROP VIEW

```
DROP VIEW [ IF EXISTS ] имя [, ...] [ CASCADE | RESTRICT ];
```

  - **CASCADE** - автоматически удалять объекты, зависящие от данного представления (например, другие представления), и, в свою очередь, все зависящие от них объекты
  - **RESTRICT** - отказать в удалении представления, если от него зависят какие-либо объекты. **Это поведение по умолчанию!**
- Выполнить эту команду может только владелец представления