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
  - 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)

- **Представления** – это сохраненные в БД именованные запросы
  - Вы можете ссылаться на представления в операторе SELECT, как на таблицы (виртуальные)
- Представления определяются с помощью оператора SELECT
- Если при создании представления в предложении **SELECT** было указано *, столбцы, добавляемые в таблицу позже, частью представления не будут!

```
CREATE [ OR REPLACE ] [ TEMP | TEMPORARY ] VIEW имя [ ( имя_столбца [, ...] ) ]
AS query
    [ WITH [ CASCADED | LOCAL ] CHECK OPTION ]
```

```
CREATE VIEW HR.EmpPhoneList
AS
SELECT empid, lastname, firstname, phone
FROM HR.Employees;
```

Изменяемые представления

- Если представление удовлетворяет следующим требованиям оно может использоваться для модификации данных в исходной таблице:
  - Запрос должно ссылаться **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

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**)
  - 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**
- 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;
```

```
REFRESH MATERIALIZED VIEW view_name;
```

Изменение представления

- Для изменения определяющего запрос представления используется оператор **CREATE VIEW OR REPLACE VIEW**:

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

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

```
ALTER VIEW [ IF EXISTS ] имя RENAME TO новое_имя
ALTER VIEW [ IF EXISTS ] имя RENAME [ COLUMN ] имя_столбца TO новое_имя_столбца
ALTER VIEW [ IF EXISTS ] имя OWNER TO новый_владелец
ALTER VIEW [ IF EXISTS ] имя SET SCHEMA новая_схема
```

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