

# ПРЕДСТАВЛЕНИЯ

План работы:

1. Создание, изменение и удаление представлений.
2. Использование представлений для упрощения сложных запросов.
3. Материализованные представления.
4. Права доступа к представлениям.

## Стандартные представления

**Представления (Views)** в базах данных – это **виртуальные таблицы**, основанные на результатах запроса к одной или нескольким физическим таблицам. Представления не хранят данные сами по себе; вместо этого, они содержат SQL-запрос, который выполняется каждый раз при обращении к представлению. Они служат удобным инструментом для упрощения сложных запросов, обеспечения безопасности данных и повышения гибкости структуры базы данных.

### Синтаксис

Команда **CREATE VIEW** используется для **создания** нового представления.

```
1 ✓ CREATE VIEW view_name AS  
2   SELECT column1, column2, ...  
3   FROM table_name  
4   WHERE condition;
```

- **view\_name:** Имя создаваемого представления.
- **SELECT column1, column2, ...:** Запрос, определяющий столбцы и данные, которые будут доступны через представление.
- **FROM table\_name:** Таблица или таблицы, из которых берутся данные.
- **WHERE condition:** (Необязательно) Условие, фильтрующее данные, отображаемые в представлении.

```
1 ✓ CREATE VIEW high_price_products AS  
2   SELECT product_id, product_name, price  
3   FROM products  
4   WHERE price > 100;
```

#### Пример использования команды **CREATE VIEW**

Это создаст представление **high\_price\_products**, которое отображает только товары из таблицы **products**, у которых цена больше 100.

Команда **ALTER VIEW** используется для **изменения** существующего представления. По сути, это замена определения запроса, на котором основано представление. В большинстве СУБД синтаксис подразумевает замену текущего определения на новое.

### Запрос История запросов

```
1 ✓ ALTER VIEW view_name AS
2   SELECT column1, column2, ... -- Новое определение SELECT запроса
3   FROM table_name
4   WHERE condition;

1 ✓ ALTER VIEW high_price_products AS
2   SELECT product_id, product_name, price, category
3   FROM products
4   WHERE price > 150;
```

#### Синтаксис и пример запроса

Это изменит определение представления **high\_price\_products**, добавив столбец **category** и изменив условие фильтрации цены на "больше 150".

Команда **DROP VIEW** используется для **удаления** существующего представления.

### Запрос История запросов

```
1 DROP VIEW view_name;
```

#### Преимущества

Представления могут значительно **упростить сложные запросы**, разбив их на более логичные и управляемые части.

Преимущества:

- Упрощение сложных запросов:** Представления позволяют скрыть сложность запросов, представляя данные в удобном и понятном виде.
- Повторное использование кода:** Определение сложного запроса можно сохранить в виде представления и использовать его многократно в разных запросах.
- Абстракция данных:** Представления позволяют скрыть детали структуры таблицы, предоставляя пользователям только необходимые данные.
- Согласованность:** Представления гарантируют, что все пользователи будут использовать один и тот же запрос для получения определенного набора данных.

#### Материализованные представления

**Материализованные представления (Materialized Views)** – это физические таблицы, хранящие результат выполнения запроса. Данные в материализованных представлениях не обновляются автоматически при изменении данных в базовых таблицах; их нужно обновлять вручную или по расписанию.

## Синтаксис

### Запрос История запросов

```
1 CREATE MATERIALIZED VIEW materialized_view_name AS  
2   SELECT column1, column2, ...  
3   FROM table_name  
4   WHERE condition;
```

Создание материализованного представления

### Запрос История запросов

```
1 REFRESH MATERIALIZED VIEW materialized_view_name;
```

Обновление материализованного представления

### Запрос История запросов

```
1 DROP MATERIALIZED VIEW materialized_view_name;
```

Удаление материализованного представления

## Преимущества

- Ускорение сложных запросов:** значительно ускоряют выполнение сложных запросов, особенно если базовые таблицы часто меняются, а данные в представлении не требуют немедленного обновления.
- Снижение нагрузки на базу данных:** снижают нагрузку на базу данных, так как сложные вычисления выполняются заранее и сохраняются в материализованном представлении.
- Поддержка исторических данных:** можно использовать для хранения исторических данных или результатов агрегации данных за определенный период.

## Случаи использования

Когда использовать материализованные представления:

- Когда требуется **значительно ускорить** выполнение сложных запросов, которые не требуют немедленного обновления данных.
- Когда необходимо **хранить** исторические данные или результаты агрегации данных.
- Когда **нагрузка** на базу данных должна быть **снижена**.

# Права доступа представлений

Как и к обычным таблицам, к представлениям **можно** применять права доступа, чтобы контролировать, кто может читать и изменять данные через представление.

Основные права доступа:

- **SELECT:** Право на чтение данных из представления;
- **INSERT:** Право на вставку данных через представление (если представление позволяет это);
- **UPDATE:** Право на обновление данных через представление (если представление позволяет это);
- **DELETE:** Право на удаление данных через представление (если представление позволяет это);
- **REFERENCES:** Право на создание внешнего ключа, ссылающегося на представление;
- **TRIGGER:** Право на создание триггеров для представления.

Команда **GRANT** используется для предоставления прав доступа к представлению.

Запрос История запросов

```
1 GRANT privilege1, privilege2, ... ON view_name TO user1, user2, ...;
```

В свою очередь команда **REVOKE** используется для отзыва прав доступа к представлению.

Запрос История запросов

```
1 REVOKE privilege1, privilege2, ... ON view_name FROM user1, user2, ...;
```

## Важно

Важные замечания о правах доступа:

1. Права доступа к представлению не обязательно совпадают с правами доступа к базовым таблицам.
2. Представления могут использоваться для ограничения доступа к определенным столбцам или строкам таблицы.
3. При предоставлении прав на изменение данных через представление (INSERT, UPDATE, DELETE) необходимо убедиться, что представление является обновляемым (updatable).
4. Для материализованных представлений обычно предоставляют только право SELECT, так как изменение данных производится путем обновления самого представления.

## Практика

Создайте тестовую базу данных **views\_lab** и несколько таблиц, которые будут использоваться для создания представлений. Таблицы должны содержать достаточное количество данных, чтобы продемонстрировать полезность использования представлений.

Запрос История запросов

```
1      CREATE TABLE customers (
2          customer_id SERIAL PRIMARY KEY,
3          customer_name VARCHAR(255),
4          city VARCHAR(255)
5      );
6
7      CREATE TABLE orders (
8          order_id SERIAL PRIMARY KEY,
9          customer_id INTEGER REFERENCES customers(customer_id),
10         order_date DATE,
11         total_amount DECIMAL(10, 2)
12     );
13
14     -- Заполнение таблиц данными (пример)
15     INSERT INTO customers (customer_name, city) VALUES
16         ('Alice Smith', 'New York'),
17         ('Bob Johnson', 'Los Angeles'),
18         ('Charlie Brown', 'Chicago');
19
20     INSERT INTO orders (customer_id, order_date, total_amount) VALUES
21         (1, '2023-01-15', 150.00),
22         (1, '2023-02-20', 200.00),
23         (2, '2023-03-10', 100.00),
24         (3, '2023-04-05', 250.00);
```

## Представления

Создайте представление `customer_names` на основе таблицы `customers`, которое отображает только `customer_id` и `customer_name`.

Запрос История запросов

```
1      CREATE VIEW customer_names AS
2          SELECT customer_id, customer_name
3              FROM customers;
```

Data Output Сообщения Notifications

CREATE VIEW

Запрос завершён успешно, время выполнения: 40 msec.

- **Создайте** представление `new_york_customers`, которое отображает клиентов из таблицы `customers`, проживающих в городе "New York".
- **Создайте** представление `customer_order_details`, которое объединяет информацию о клиентах из таблицы `customers` и заказах из таблицы `orders` и отображает `customer_name, order_date` и `total_amount` (используйте JOIN).

- **Измените** представление `customer_names` так, чтобы оно также отображало столбец `city`.
- **Удалите** представление `new_york_customers`.
- **Создайте** представление `customer_order_summary`, которое для каждого клиента вычисляет общее количество заказов и общую сумму заказов.
- **Используйте** представление `customer_order_summary` для получения списка клиентов, у которых общая сумма заказов превышает 300 (**используйте SELECT**).
- **Создайте** представление `top_customers`, которое основано на представлении `customer_order_summary` и отображает только 3 клиентов с наибольшей общей суммой заказов.

## Материализованные представления

Создайте материализованное представление `monthly_sales`, которое вычисляет общую сумму продаж за каждый месяц.

```

Запрос История запросов
1 CREATE MATERIALIZED VIEW monthly_sales AS
2   SELECT EXTRACT(MONTH FROM order_date) AS month, SUM(total_amount) AS total_sales
3     FROM orders
4   GROUP BY EXTRACT(MONTH FROM order_date);
Data Output Сообщения Notifications
SELECT 4

Запрос завершён успешно, время выполнения: 33 msec.

```

- **Выполните запрос** к материализованному представлению `monthly_sales` для просмотра данных.
- **Добавьте** новые данные в таблицу `orders` и обновите материализованное представление `monthly_sales`.
- **Сравните** время выполнения запроса, вычисляющего общую сумму продаж за каждый месяц непосредственно из таблицы `orders` и из материализованного представления `monthly_sales`. **Используйте EXPLAIN ANALYZE**.

## Предоставление прав

Создадим нового пользователя `test_user` с паролем «123».

```

Запрос История запросов
1 CREATE USER test_user WITH PASSWORD '123';
Data Output Сообщения Notifications
CREATE ROLE

Запрос завершён успешно, время выполнения: 71 msec.

```

- **Предоставьте** пользователю `test_user` право **SELECT** на представление `customer_names`.
- **Отзовите** у пользователя `test_user` право **SELECT** на представление `customer_names`.
- **Самостоятельно создайте** 8 любых представлений для базы данных «**Склад**» из которых 3 - материализованные.

**!** После выполнения самостоятельного задания, удалите базу данных `views_lab` и «**Склад**».