Подзапросы

подзапрос выполняется до основного запроса, это временная таблица с областью видимости запроса, после выполнения запроса данные удаляются. Если возврат одну строку и один столбец, то в основном запросе используется равенство (<>, >=,...)

```
SELECT customer_id, first_name, last_name FROM customer WHERE customer_id = (SELECT MAX(customer_id)
FROM customer);
```

если множество строк:
in, not in в результирующем множестве
<>=all All сравнивает значение с каждым значением в подзапросе
<>=any - если хотя-бы одно выполняется, то true HAVING sum(amount) > ANY (подзапрос)
если в результате подзапроса одно из значений null, а сравнение not in, то будет пустой вывод: результат сравнения с null - unknown

Типы подзапросов

некоррелированный: может быть выполнен отдельно и не ссылается ни на что из другого
techreллированный: когда в подзапросе используется таблица запроса. выполняется по одному разу для каждой строки-кандидата. Может быть проблема с производительностью. часто вместе с update, delete

```
UPDATE customer c
SET c.last_update = (SELECT max(г.rental_date) FROM rental r WHERE r.customer_id = c.customer_id);
```

проверка наличия

```
UPDATE customer c SET c.last_update =
(SELECT max(г.rental_date) FROM rental r WHERE r.customer_id = c.customer_id)
WHERE EXISTS
(SELECT 1 FROM rental r WHERE r.customer_id = c.customer_id);
```

```
select f.title from film f where exist (
select 1 from film_category fc where category_id = 10 and f.film_id = fc.film_id)
```

создание данных подзапросом. Нужно разделить на группы, когда такой группировки нет в базе

```
SELECT pymnt_grps.name, count(*) num_customers FROM
(SELECT customer_id, count(*) num rentals, sum(amount) tot_payments
    FROM payment GROUP BY customer_id) pymnt
INNER JOIN
(SELECT 'Small Fry' name, 0 low_limit, 74.99 high_limit UNION ALL
SELECT 'Average Joes' name, 75 low_limit, 149.99 high_limit UNION ALL
SELECT 'Heavy Hitters' name, 150 low_limit, 9999999.99 high_limit) pymnt_grps
ON pymnt.tot_payments BETWEEN pymnt_grps.low_limit AND pymnt_grps.high_limit
GROUP BY pymnt_grps.name;
```

Обобщенные табличные выражения: создается именованый подзапрос, который потом используется в запросе with actor_s as (запрос) Например общий доход от проката тех фильмов с рейтингом PG, актерский состав которых включает актера, фамилия которого начинается с S

```
WITH actors_s AS (SELECT actor_id, first_name, last_name FROM actor WHERE last name LIKE 'S%'),
actors_s_pg AS (SELECT s.actor_id, s.first_name, s.last_name,f.film_id, f.title FROM actors_s s
INNER JOIN film_actor fa ON s.actor_id = fa.actor_id
INNER JOIN film f ON f.film_id = fa.film_id
WHERE f.rating = 'PG'),
actors_s_pg_revenue AS (SELECT spg.first_name, spg.last_name, p.amount FROM actors_s_pg spg
INNER JOIN inventory i ON i.film_id = spg.film_id
INNER JOIN rental r ON i.inventory_id = r.inventory_id
INNER JOIN payment p ON r.rental_id = p.rental_id)
SELECT spg_rev.first_name, spg_rev.last_name, sum(spg_rev.amount) tot_revenue
FROM actors_s_pg_revenue spg_rev GROUP BY spg_rev.first_name, spg_rev.last_name ORDER BY 3 desc;
```

скалярные подзапросы можно использовать и в select

```
SELECT (SELECT c.first_name FROM customer c WHERE c.customer_id = p.customer_id) first_name,
(SELECT c.last_name FROM customer c WHERE c.customer_id = p.customer_id) last__name,
(SELECT ct.city FROM customer c INNER JOIN address a ON c.address_id = a.address_id
INNER JOIN city ct ON a.city_id = ct.city_id WHERE c.customer_id = p.customer_id) city,
sum(p.amount) tot_payments, count(*) tot_rentals FROM payment p GROUP BY p.customer id;
```

создать новую строку в таблице film_actor и у вас имеются следующие данные:имя и фамилия актера; название фильма

```
INSERT INTO film_actor (actor_id, film_id, last_update) VALUES (
(SELECT actor_id FROM actor WHERE first_name = ’JENNIFER' AND last_name = ’DAVIS'),
(SELECT film_id FROM film WHERE title = 'ACE GOLDFINGER'),
```

```
now());
```

Revision #1
Created 7 April 2025 05:36:37 by Admin
Updated 7 April 2025 05:45:00 by Admin