Darbas su duomenimis

9. Subqueries

11. Sąlygos

Subquery –užklausa užklausoje

Užklausa, kuri yra kitoje užklausoje. Jinai gražina

1. Vieną eilutę ir vieną stulpelį
2. Daug eilučių ir vieną stulpelį
3. Daug eilučiu ir daug stulpelių

Subquery

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

Subquery tipai

Noncorrelated subqueries – jeigu įvykdžius subquery atskirai gausime atsakymą.

Scalar subqueries

Subqueries, kurie naudojami su palyginimo operacijomis

=, <>, <, >, <=, >=

```
SELECT city_id, city
    FROM city
   WHERE country_id <>
    (SELECT country_id FROM country WHERE country = 'India');
```

Daug eilučių vienas stulpelis subqueries

Tokiems subqueries naudojam in arba not in operatorius.

```
SELECT city_id, city
     FROM city
     WHERE country_id NOT IN
      (SELECT country_id
      FROM country
      WHERE country IN ('Canada','Mexico'));
```

Operatorius ALL

```
SELECT first_name, last_name
    FROM customer
    WHERE customer_id <> ALL
     (SELECT customer_id
      FROM payment
     WHERE amount = 0);
```

Toks pats rezultatas kaip ir

```
SELECT first_name, last_name
FROM customer
WHERE customer_id NOT IN
 (SELECT customer_id
  FROM payment
  WHERE amount = 0)
```

Operatorius ALL

```
SELECT customer_id, count(*)
    FROM rental
    GROUP BY customer_id
    HAVING count(*) > ALL
     (SELECT count(*)
      FROM rental r
        INNER JOIN customer c
        ON r.customer_id =
c.customer_id
        INNER JOIN address a
        ON c.address_id = a.address_id
        INNER JOIN city ct
        ON a.city_id = ct.city_id
        INNER JOIN country co
        ON ct.country_id = co.country_id
      WHERE co.country IN
('United States','Mexico','Canada')
      GROUP BY r.customer_id);
```

Operatorius ANY

```
SELECT customer_id, sum(amount)
  FROM payment
  GROUP BY customer_id
  HAVING sum(amount) > ANY
   (SELECT sum(p.amount)
    FROM payment p
      INNER JOIN customer c
      ON p.customer_id =
c.customer_id
      INNER JOIN address a
      ON c.address_id = a.address_id
      INNER JOIN city ct
      ON a.city_id = ct.city_id
      INNER JOIN country co
      ON ct.country_id = co.country_id
     WHERE co.country IN
('Bolivia','Paraguay','Chile')
     GROUP BY co.country
    );
```

Daug stulpeliu, daug eilučių subqueries

```
SELECT fa.actor_id, fa.film_id
    FROM film_actor fa
    WHERE fa.actor_id IN
     (SELECT actor_id FROM actor
WHERE last_name = 'MONROE')
     AND fa.film_id IN
     (SELECT film_id FROM film
WHERE rating = 'PG');
```

```
SELECT actor_id, film_id
    FROM film_actor
    WHERE (actor_id, film_id) IN
     (SELECT a.actor_id, f.film_id
      FROM actor a
        CROSS JOIN film f
      WHERE a.last_name =
'MONROE'
      AND f.rating = 'PG');
```

Correlated subqueries

```
SELECT c.first_name, c.last_name
    FROM customer c
    WHERE 20 =
     (SELECT count(*) FROM rental r
      WHERE r.customer_id = c.customer_id);
```

Dar sudėtingiau

```
SELECT c.first_name, c.last_name
    FROM customer c
    WHERE
     (SELECT sum(p.amount) FROM payment p
      WHERE p.customer_id = c.customer_id)
      BETWEEN 180 AND 240;
```

Exists operatorius

Atraskime visus klientus, kurie išsinuomojo bent po vieną filmą iki 2005-05-25

```
SELECT c.first_name, c.last_name
     FROM customer c
     WHERE EXISTS
      (SELECT 1 FROM rental r
       WHERE r.customer_id = c.customer_id
         AND date(r.rental_date) < '2005-05-25');
```

Exists operatorius

Exists operatorius gali gražinti 0, 1 arba daug eilučių.

Taip pat galima naudoti ir NOT EXISTS

```
SELECT a.first_name, a.last_name
     FROM actor a
     WHERE NOT EXISTS
      (SELECT 1
       FROM film_actor fa
         INNER JOIN film f ON f.film_id = fa.film_id
       WHERE fa.actor_id = a.actor_id
         AND f.rating = 'R');
```

Manipuliavimas duomeminis naudojantis correlated subqueries

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

UPDATE customer c
SET c.last_update =
 (SELECT max(r.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);
```

Manipuliavimas duomeminis naudojantis correlated subqueries

```
DELETE FROM customer
WHERE 365 < ALL
 (SELECT datediff(now(), r.rental_date) days_since_last_rental
  FROM rental r
  WHERE r.customer_id = customer.customer_id);
```

Naudojimo pavyzdžiai

Norime pridėti nauju stulpelių apibendrindami turimus duomenis

```
SELECT c.first_name, c.last_name,
       pymnt.num_rentals, pymnt.tot_payments
     FROM customer c
       INNER JOIN
        (SELECT customer_id,
           count(*) num_rentals, sum(amount) tot_payments
         FROM payment
         GROUP BY customer_id
        ) pymnt
       ON c.customer_id = pymnt.customer_id;
```

Naudojimo pavyzdžiai

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

Common table expressions a.k.a. CTEs

```
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
    ) -- end of With clause
  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;
```

Subqueries SELECT operatoriuje

```
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;
```

Subqueries ORDER BY operatoriuje

```
SELECT a.actor_id, a.first_name, a.last_name
    FROM actor a
    ORDER BY
     (SELECT count(*) FROM film_actor fa
      WHERE fa.actor_id = a.actor_id) DESC;
```

Užduotys

1. Construct a query against the film table that uses a filter condition with a noncorrelated subquery against the category table to find all action films (category.name = 'Action‘).
2. Rework the query from Exercise 9-1 using a correlated subquery against the category and film_category tables to achieve the same results.