Darbas su duomenimis

5. Inner joins

10. Joins

Kas yra join?

- desc customer;
- desc address;
- Norime pamatyti kliento vardą, pavardę ir adresą vienoje eilutėje.

- Foreign key constrain – jį galima naudoti, norint užtikrinti, kad customer lentelėje įvesti adresai egzistuotų address lentelėje.

Dekarto sandauga

- SELECT c.first_name, c.last_name, a.address FROM customer c JOIN address a;

  Jeigu nenurodome kaip jungiame lenteles, duomenų bazės serveris gražina dekarto sandaugą, dar vadinama kaip *cross join.*

Inner join

- SELECT c.first_name, c.last_name, a.address FROM customer c JOIN address a ON c.address_id = a.address_id ;
- SELECT c.first_name, c.last_name, a.address FROM customer c INNER JOIN address a ON c.address_id = a.address_id;
- JOIN = INNER JOIN
- SELECT c.first_name, c.last_name, a.address FROM customer c INNER JOIN address a USING (address_id);
- Jeigu stulpelių vardai sutampa galime naudoti USING.

Join 3+ lenteles

- Desc address;
- Desc city;
- Prie klientų vardų, pavardžių ir adreso prijungsime miestą.
- SELECT c.first_name, c.last_name, 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;
- Jungimo eiliškumas nėra svarbus!
- Galimas skirtumas, kurį pastebėsite, eilučių gražinimo eiliškumas.

Lentelių jungimas norimu eiliškumu

- SELECT STRAIGHT_JOIN c.first_name, c.last_name, ct.city

FROM city ct

INNER JOIN address a

ON a.city_id = ct.city_id

INNER JOIN customer c

ON c.address_id = a.address_id;

Subquery prijungimas

```
SELECT c.first_name, c.last_name, addr.address, addr.city
    FROM customer c
      INNER JOIN
       (SELECT a.address_id, a.address, ct.city
        FROM address a
          INNER JOIN city ct
          ON a.city_id = ct.city_id
       WHERE a.district = 'California'
       ) addr
      ON c.address_id = addr.address_id;
```

Subquery prijungimas

- Gali būti naudingas
  - Skaitomumui
  - Vykdymo trukmei sumažinti

Tos pačios lentelės prijungimas kelis kartus

- Norite sužinoti filmus, kur vaidino du konkretūs aktoriai

Užklausa gražina filmus, kur vaidino vienas arba kitas aktorius:

```
SELECT f.title
    FROM film f
     INNER JOIN film_actor fa
     ON f.film_id = fa.film_id
    INNER JOIN actor a
     ON fa.actor_id = a.actor_id
    WHERE ((a.first_name = 'CATE' AND a.last_name = 'MCQUEEN')
      OR (a.first_name = 'CUBA' AND a.last_name = 'BIRCH'));
```

Tos pačios lentelės prijungimas kelis kartus

```
SELECT f.title
    FROM film f
     INNER JOIN film_actor fa1
     ON f.film_id = fa1.film_id
     INNER JOIN actor a1
     ON fa1.actor_id = a1.actor_id
    INNER JOIN film_actor fa2
     ON f.film_id = fa2.film_id
     INNER JOIN actor a2
     ON fa2.actor_id = a2.actor_id
  WHERE (a1.first_name = 'CATE' AND a1.last_name = 'MCQUEEN')
   AND (a2.first_name = 'CUBA' AND a2.last_name = 'BIRCH');
```

Self-joins

- self-referencing foreign key
  https://pencilprogrammer.com/self-referencing-foreign-key-in-mysql/

- film lentelė turėtų stulpelį prequel_film_id
  - FIDDLER LOST II – title, FIDDLER LOST - prequel

```
SELECT f.title, f_prnt.title prequel
    FROM film f
     INNER JOIN film f_prnt
      ON f_prnt.film_id = f.prequel_film_id
    WHERE f.prequel_film_id IS NOT NULL;
```

Pratimai

1. Parašykite užklausą, kuri gražintų filmo pavadinimą ir jame vaidinusius aktorius, kurių vardai yra JOHN.
2. Parašykite užklausą, kuri gražina adresus, kurie yra tame pačiame mieste.

Outer joins (išoriniai jungimai)

- Inventory lentelė – 958 filmai.
- Film lentelė – 1000 filmų.

```
SELECT f.film_id, f.title, count(*) num_copies
    FROM film f
      INNER JOIN inventory i
      ON f.film_id = i.film_id
   GROUP BY f.film_id, f.title;
```

Outer joins (išoriniai jungimai)

```
SELECT f.film_id, f.title, count(i.inventory_id) num_copies
    FROM film f
     LEFT OUTER JOIN inventory i
     ON f.film_id = i.film_id
    GROUP BY f.film_id, f.title;
```

Outer joins (išoriniai jungimai)

```
ELECT f.film_id, f.title, i.inventory_id
    FROM film f
      INNER JOIN inventory i
       ON f.film_id = i.film_id
     WHERE f.film_id BETWEEN 13 AND 15;
```

VS.

```
SELECT f.film_id, f.title, i.inventory_id
FROM film f
       LEFT OUTER JOIN inventory i
       ON f.film_id = i.film_id
     WHERE f.film_id BETWEEN 13 AND 15;
```

Left vs. Righ join

Nusako, kuri lentelė atsakinga sprendžiant, kiek eilučių bus gražinta.

Visada naudokite left outer join.

3+ lentelės

```
SELECT f.film_id, f.title, i.inventory_id, r.rental_date
    FROM film f
      LEFT OUTER JOIN inventory i
      ON f.film_id = i.film_id
      LEFT OUTER JOIN rental r
      ON i.inventory_id = r.inventory_id
    WHERE f.film_id BETWEEN 13 AND 15
```

Cross joins – dekarto sandauga

```
SELECT c.name category_name, l.name language_name
    FROM category c
     CROSS JOIN language l;
```

96 eilutės (16 category lentelės eilučių × 6 language lentelės eilutės).

Cross joins panaudojimas

Norite sugeneruoti lentelę, kuri turėtų visas 2020 metų dienas.

```
SELECT days.dt, COUNT(r.rental_id) num_rentals
    FROM rental r
     RIGHT OUTER JOIN
    (SELECT DATE_ADD('2005-01-01',
     INTERVAL (ones.num + tens.num + hundreds.num) DAY) dt
     FROM
     (SELECT 0 num UNION ALL
      SELECT 1 num UNION ALL
      SELECT 2 num UNION ALL
      SELECT 3 num UNION ALL
     SELECT 4 num UNION ALL
      SELECT 5 num UNION ALL
      SELECT 6 num UNION ALL
     SELECT 7 num UNION ALL
      SELECT 8 num UNION ALL
      SELECT 9 num) ones
      CROSS JOIN
    (SELECT 0 num UNION ALL
      SELECT 10 num UNION ALL
     SELECT 20 num UNION ALL
      SELECT 30 num UNION ALL
      SELECT 40 num UNION ALL
      SELECT 50 num UNION ALL
      SELECT 60 num UNION ALL
      SELECT 70 num UNION ALL
      SELECT 80 num UNION ALL
      SELECT 90 num) tens
      CROSS JOIN
     (SELECT 0 num UNION ALL
      SELECT 100 num UNION ALL
      SELECT 200 num UNION ALL
      SELECT 300 num) hundreds
     WHERE DATE_ADD('2005-01-01',
      INTERVAL (ones.num + tens.num + hundreds.num) DAY)
       < '2006-01-01'
    ) days
    ON days.dt = date(r.rental_date)
   GROUP BY days.dt
    ORDER BY 1;
```

Natural joins

- Duomenų bazės serveris pats nusprendžia, kokį sujungimo būdą naudoti.

```
SELECT c.first_name, c.last_name, date(r.rental_date)
    FROM customer c
     NATURAL JOIN rental r;

SELECT cust.first_name, cust.last_name, date(r.rental_date)
    FROM
     (SELECT customer_id, first_name, last_name
      FROM customer
     ) cust
      NATURAL JOIN rental r;
```

Užduotys

1. Naudodamiesi Customer ir Payment lentele parašykite užklausą, kuri gražintų kiekvieno kliento mokėjimų sumą.
2. Perrašykite pirmos užduoties užklausą naudodami right outer join.
3. Parašykite užklausą, kuri sugeneruotų skaičių aibę nuo 1 iki 100.