readme.md

Testovanie a správne spustenie skriptov

- Ako prvé je potreba inicializovať tabuľky, skripty na tento účel sú uložené v zložke **facepage**. Spúšťajte v poradí(initTables->AddData->AsoAcc).
- Pre správnu demonštráciu funkčnosti skriptov sme pridali skripty na pridanie viacerých hodnôt, tieto sú v zložke **adding_more_data**. Spúšťajte v poradí(AddMoreData->AsoAcc)
- Skripty selectov nájdete v zložke **select_from_homework**

Zadanie prvého selectu

1.) Kdo jsou přátelé uživatele/uživatelů se jménem Jan Novák? Očekává se tabulka výsledku se schématem (přihlašovací jméno Jana Nováka, e-mail Jana Nováka, přihlašovací jméno přítele, jméno a příjmení přítele, e-mail přítele).Uspořádejte podle přihlašovacího jména Jana Nováka.

- Informácie o užívateľoch máme v tabulke **"USER"** a informácie o tom kto je s kým priateľ máme v tabulke **ASSOCIATED_ACCOUNTS**
- Tabulka **ASSOCIATED_ACCOUNTS** môže mať aj v stĺpci **"accountNumber1"** a v **"accountNumber2"** ID pána Nováka preto najprv si túto tabuľku upravíme a to

```
    (SELECT "accountNumber2" as "Jan", "accountNumber1" as "Friend" FROM ASSOCIATED_ACCOUNTS)
    UNION
    (SELECT "accountNumber1" as "Jan", "accountNumber2"  as Friend FROM ASSOCIATED_ACCOUNTS )
```

a z tabuľky **"USER"** si vyberieme riadok kde meno sa volá "Jan Novak"

```
 ( SELECT "loginName" as "login_name1" , "email" as "email_1","nameSurname" as "name_1" , "accountNumber" FROM "USER"
```

Tieto dve tabuľky sa napoja pomocou spoločného kľúča

```
   ON  associated."Jan" = t1."accountNumber"
```

a pomocou príkazu inner join

```
( Select * From
    ( SELECT "loginName" as "login_name1" , "email" as "email_1","nameSurname" as "name_1" , "accountNumber" FROM "USE
inner join
    (
        (SELECT "accountNumber2" as "Jan", "accountNumber1" as "Friend" FROM ASSOCIATED_ACCOUNTS)
        UNION
        (SELECT "accountNumber1" as "Jan", "accountNumber2"  as Friend FROM ASSOCIATED_ACCOUNTS )
    ) associated
ON  associated."Jan" = t1."accountNumber"
) leftSide
```

Takto nám vznikne tabuľka s informáciami Jána Nováka a s kľúčom priateľov Jána Nováka

Teraz si upravíme tabuľku **"USER"** na informácie o priateľovi a to nasledovne

```
(select "accountNumber", "nameSurname" as "name_2" ,"loginName" as "login_name2", "email"  as "email_2" from "USER" )
```

a túto tabuľku znova napojíme pomocou inner join na základe kľúču priateľa

```
ON leftSide."Friend" = rightside."accountNumber"
```

Následne nám vznikne celý kód:

```
Select  "login_name1",  "email_1" , "login_name2" , "name_2","email_2" From

(select "accountNumber", "nameSurname" as "name_2" ,"loginName" as "login_name2", "email"  as "email_2" from "USER" )

inner join

(Select * From
    ( SELECT "loginName" as "login_name1" , "email" as "email_1","nameSurname" as "name_1" , "accountNumber" FROM "USE
inner join
    (
        (SELECT "accountNumber2" as "Jan", "accountNumber1" as "Friend" FROM ASSOCIATED_ACCOUNTS)
        UNION
        (SELECT "accountNumber1" as "Jan", "accountNumber2"  as Friend FROM ASSOCIATED_ACCOUNTS )
    ) associated
ON  associated."Jan" = t1."accountNumber"
) leftSide
On leftSide."Friend" = rightside."accountNumber"
```

Zadanie druhého selectu

2.Kolik přátel a kolika různých ročníků (roků narození) mají jednotliví uživatelé se jménem Jan Novák? Očekává se tabulka výsledku se schématem (přihlašovací jméno Jana Nováka, e-mail Jana Nováka, počet přátel, počet ročníků).

- Ako prvé sme si z tabulky **ASSOCIATED_ACCOUNTS** odstránili riadky s redundantnými priatelstvami ktoré boli len v opačnom poradí.

```
SELECT aa1."accountNumber1" account1, aa1."accountNumber2" account2
FROM ASSOCIATED_ACCOUNTS aa1
WHERE NOT EXISTS
    (SELECT *
    FROM ASSOCIATED_ACCOUNTS aa2
    WHERE (aa1."accountNumber1" = aa2."accountNumber2" AND aa1."accountNumber2" = aa2."accountNumber1"))
```

- V tabuľke všetkých užívateľov **"USER"** hľadáme čísla účtov **"accountNumber"** pre všetkých Janov Novákov, tento výber neskôr využijeme na vyhľadávanie Janovich priatelstiev.

```
SELECT "USER"."accountNumber" account3
    FROM "USER"
    WHERE "USER"."nameSurname" = 'Jan Novak'
```

- Tieto dva výbery teraz využijeme na nájdenie priatelstiev Jána Nováka. Teda hľadáme výskyt Janovho **"accountNumber"** v skôr vytvorených stĺpcoch **account1** a **account2**, prípadné vyskyty Jána nováka na riadku uložím do stĺpca **NovakAccounts.account3** ktorý vyberám spoločne s počtom týchto vyskytov(počet priatelstiev Jana Nováka) **COUNT(NovakAccounts.account3)**.

```
SELECT NovakAccounts.account3, COUNT(NovakAccounts.account3) count
FROM (
    SELECT aa1."accountNumber1" account1, aa1."accountNumber2" account2
    FROM ASSOCIATED_ACCOUNTS aa1
    WHERE NOT EXISTS
        (SELECT *
        FROM ASSOCIATED_ACCOUNTS aa2
        WHERE (aa1."accountNumber1" = aa2."accountNumber2" AND aa1."accountNumber2" = aa2."accountNumber1"))
    ) friendsData,
     (SELECT "USER"."accountNumber" account3
     FROM "USER"
     WHERE "USER"."nameSurname" = 'Jan Novak') NovakAccounts
WHERE NovakAccounts.account3 = friendsData.account2 OR  NovakAccounts.account3 = friendsData.account1
GROUP BY NovakAccounts.account3
```

- Následne už len vyberiem **"USER"."accountNumber"** z **"USER"** ktoré porovnávam s **matches.account3**, teda číslo účtu Jána Nováka, ktoré sa vyskytlo v tabulke priatelstiev.

```
SELECT "USER"."nameSurname",  "USER"."loginName", matches.count
FROM "USER",
    (SELECT NovakAccounts.account3, COUNT(NovakAccounts.account3) count
    FROM (
        SELECT aa1."accountNumber1" account1, aa1."accountNumber2" account2
        FROM ASSOCIATED_ACCOUNTS aa1
        WHERE NOT EXISTS
            (SELECT *
            FROM ASSOCIATED_ACCOUNTS aa2
            WHERE (aa1."accountNumber1" = aa2."accountNumber2" AND aa1."accountNumber2" = aa2."accountNumber1"))
        ) friendsData,
         (SELECT "USER"."accountNumber" account3
         FROM "USER"
         WHERE "USER"."nameSurname" = 'Jan Novak') NovakAccounts
    WHERE NovakAccounts.account3 = friendsData.account2 OR  NovakAccounts.account3 = friendsData.account1
    GROUP BY NovakAccounts.account3) matches
WHERE "USER"."accountNumber" = matches.account3;
```

Zadanie tretieho selectu

3.) Kolik přátel a kolika různých ročníků (roků narození) mají jednotliví uživatelé se jménem Jan Novák? Očekává se tabulka výsledku se schématem (přihlašovací jméno Jana Nováka, e-mail Jana Nováka, počet přátel, počet ročníků). Musí být vidět i ty Jany Nováky, kteří nemají žádné přátele.

- Zadanie je veľmi podobné druhému zadaniu len s tým rozdielom že je treba vybrať aj Jánov Novákov bez priateľov. Toto dosiahneme pomocou unionu kde vyberáme loginName, nameSurname, email podľa zadania pričom pomocou letfJoinu kontroluje napojenie na tabuľku **ASSOCIATED_ACCOUNTS** pričom kontrolujeme pomocou podmienky IS NULL pravu alebo ľavú časť **ASSOCIATED_ACCOUNTS** a súčasne že na vyberanom riadku je údaj s **nameSurname** Jan Novák.

```
--viď 2 select--
      +
UNION
SELECT "nameSurname", "loginName", 0
FROM "USER"  t1
LEFT JOIN ASSOCIATED_ACCOUNTS t2 ON  (t2."accountNumber1" = t1."accountNumber"  OR  "accountNumber2" = t1."accountNumb
WHERE t1."nameSurname" = 'Jan Novak' AND (t2."accountNumber1" IS NULL  or t2."accountNumber2"  IS NULL);
```

Zadanie štvrtého selectu

4.) Kdo má nejvíce přátel a kolik? Takových uživatelů může být více. Očekává se tabulka výsledku se schématem (přihlašovací jméno, jméno a příjmení, e-mail, počet přátel).

- Znova podobne ako v selectoch 2,3 hľadáme údaje priatelov v **ASSOCIATED_ACCOUNTS**, teraz už nie len pre Jána Nováka. Z tejto tabuľky zistíme čísla účtov priatelov na základe ktorých tvoríme INNER JOIN na číslo účtov z **"USER**. Následne si len vyberiem údaje potrebné zo zadania a hlavne si ukladám počet priatelov **numOfFriends** pre každého užívateľa s priateľmi. Dostávame výber **usersWithFriends**.

```
SELECT f2.login, f2.name, f2.mail, f2.count numOfFriends
FROM
    (SELECT u1."loginName" login, u1."nameSurname" name, u1."email" mail, COUNT(u1."loginName") as count
    FROM "USER" u1
    INNER JOIN (
        SELECT aa1."accountNumber1", aa1."accountNumber2"
        FROM ASSOCIATED_ACCOUNTS aa1
        WHERE NOT EXISTS
            (SELECT *
            FROM ASSOCIATED_ACCOUNTS aa2
            WHERE (aa1."accountNumber1" = aa2."accountNumber2" AND aa1."accountNumber2" = aa2."accountNumber1"))
        ) f1
    ON u1."accountNumber" = f1."accountNumber1" OR u1."accountNumber" = f1."accountNumber2"
    GROUP BY u1."loginName", u1."nameSurname", u1."email") f2
ORDER BY f2.count DESC
```

- ako posledné potrebujeme už len vyfiltrovať užívateľov ktorí majú maximálny počet. Ten zistím rovnako ako kód vyššie. Len s tým rozdielom že vyberáme len údaj **MAX(t2.count)**, ktorý porovnávame s **usersWithFriends.numOfFriends** z predchádzajúceho selectu, teda zostávajú len údaje, ktoré sa rovnajú maximálnemu počtu priateľov.

```
SELECT *
FROM
    (SELECT f2.login, f2.name, f2.mail, f2.count numOfFriends
     FROM
        (SELECT u1."loginName" login, u1."nameSurname" name, u1."email" mail, COUNT(u1."loginName") as count
        FROM "USER" u1
        INNER JOIN (
            SELECT aa1."accountNumber1", aa1."accountNumber2"
            FROM ASSOCIATED_ACCOUNTS aa1
            WHERE NOT EXISTS
                (SELECT *
                FROM ASSOCIATED_ACCOUNTS aa2
                WHERE (aa1."accountNumber1" = aa2."accountNumber2" AND aa1."accountNumber2" = aa2."accountNumber1"))
            ) f1
        ON u1."accountNumber" = f1."accountNumber1" OR u1."accountNumber" = f1."accountNumber2"
        GROUP BY u1."loginName", u1."nameSurname", u1."email") f2) usersWithFriends
WHERE usersWithFriends.numOfFriends =
    (SELECT MAX(t2.count) maxCount
     FROM
        (SELECT u2."loginName", COUNT(u2."loginName") as count
        FROM "USER" u2
        INNER JOIN (
            SELECT aa1."accountNumber1", aa1."accountNumber2"
            FROM ASSOCIATED_ACCOUNTS aa1
            WHERE NOT EXISTS
                (SELECT *
                FROM ASSOCIATED_ACCOUNTS aa2
                WHERE (aa1."accountNumber1" = aa2."accountNumber2" AND aa1."accountNumber2" = aa2."accountNumber1"))
            ) t1
        ON u2."accountNumber" = t1."accountNumber1" OR u2."accountNumber" = t1."accountNumber2"
        GROUP BY u2."loginName") t2);
```

Zadanie piateho selectu

5.) Kteří uživatelé nemají žádného přítele? Očekává se tabulka výsledku se schématem (přihlašovací jméno, jméno a příjmení, e-mail).

```
SELECT "loginName", "nameSurname","email"
FROM "USER"  t1
LEFT JOIN ASSOCIATED_ACCOUNTS t2 ON t2."accountNumber1" = t1."accountNumber"  OR  "accountNumber2" = t1."accountNumber
WHERE t2."accountNumber1" IS NULL  or t2."accountNumber2"  IS NULL
```

- Vyberáme **loginName**, **nameSurname**,**email** podľa zadania pričom pomocou letfJoinu kontroluje napojenie na tabuľku **ASSOCIATED_ACCOUNTS** pričom kontrolujeme pomocou podmienky IS NULL pravu alebo ľavú časť **ASSOCIATED_ACCOUNTS**.

Výsledné tabuľky selectov

1.select

	"login_name1"	"email_1"	"login_name2"	"name_2"	"email_2"
1	Janko123	janko@php.net	pepa	Pepa Dvořák	pepa@stud.fit.vutbr.cz
2	pepega12	novakjano6@pepega.com	pepa	Pepa Dvořák	pepa@stud.fit.vutbr.cz
3	Janko123	janko@php.net	nshillaber2	Nerty Shillaber	nshillaber2@bloglovin.com
4	Janko123	janko@php.net	dbittlestone4	Dagny Bittlestone	dbittlestone4@cnn.com
5	Janko123	janko@php.net	koshee7	Kellie OShee	koshee7@histats.com

2.select

	nameSurname	loginName	COUNT
1	Jan Novak	Janko123	3
2	Jan Novak	pepega12	1

3.select

	nameSurname	loginName	NUMOFFRIENDS
1	Jan Novak	Janko123	3
2	Jan Novak	clownfiesta12	0
3	Jan Novak	pepega12	1

4.select

	LOGIN	NAME	MAIL	NUMOFFRIENDS
1	Janko123	Jan Novak	janko@php.net	3
2	pepa	Pepa Dvořák	pepa@stud.fit.vutbr.cz	3

5.select

	"loginName"	"nameSurname"	"email"
1	alucius0	Adan Lucius	alucius0@php.net
2	alemmanbie1	Ashil Lemmanbie	alemmanbie1@cdbaby.com
3	pgood3	Pammie Good	pgood3@yolasite.com
4	iprisley5	Isidore Prisley	iprisley5@dedecms.com
5	gsoppitt6	Georgianne Soppitt	gsoppitt6@nbcnews.com
6	jsauven8	Janaye Sauven	jsauven8@ihg.com
7	clownfiesta12	Jan Novak	novakjano6@pepega.com