Seminarska naloga PB

Class	PB
Created	@May 25, 2021 5:42 PM
Letnik	1
Semester	2
Type	Homework
Updated	@May 30, 2021 11:53 PM

1. Naloga (DDL)

▼ Navodila

Iz tabele x_world, opisane z relacijsko shemo

```
x_world(id, x, y, tid, vid, village, pid, player, aid, alliance, population)
```

ustvarite (CREATE TABLE) in napolnite tabele z naslednjimi relacijskimi shemami in pomeni:

- `pleme(tid, tribe)`
  - šifra in ime plemena (imena vstavite ročno – glej opis datoteke Xworld.sql v teh navodilih)
- `aliansa(aid, alliance)`
  - šifra in ime alianse
- `igralec(pid, player, #tid, #aid)`
  - šifra in ime igralca, njegovo pleme in njegova aliansa
- `naselje(vid, village, x, y, population, #pid)`
  - šifra vasi, ime vasi, x in y koordinati, populacija, šifra igralca lastnika vasi

Iz tabele aliansa odstranite vrstico z vrednostjo aid = 0 ter v tabeli igralec zamenjajte vse vrednosti aid = 0 z NULL. Pri vseh tabelah tudi pravilno določite primarne in tuje ključe. Tabeli naselje dodajte tudi omejitve (CHECK), tako da sprejme le pravilne vnose koordinat in populacije.

▼ Resitev

Kreiramo tabele, pazimo na pravilno definicijo atributov, ter vrstni red kreiranja tabel (tabele "igralec" nemoremo kreirati pred npr. tabelo "pleme").

```
CREATE TABLE IF NOT EXISTS pleme
(
    tid   INTEGER PRIMARY KEY,
    tribe VARCHAR(30)
);

CREATE TABLE IF NOT EXISTS aliansa
(
    aid      INTEGER PRIMARY KEY,
    alliance VARCHAR(30)
);

CREATE TABLE IF NOT EXISTS igralec
(
    pid    INTEGER PRIMARY KEY,
    player VARCHAR(30),
    tid    INTEGER,
    aid    INTEGER,
    FOREIGN KEY (tid) REFERENCES pleme (tid),
    FOREIGN KEY (aid) REFERENCES aliansa (aid)
```

```
);

CREATE TABLE IF NOT EXISTS naselje
(
    vid        INTEGER PRIMARY KEY,
    village    VARCHAR(30),
    x          INTEGER NOT NULL,
    y          INTEGER NOT NULL,
    population INTEGER,
    pid        INTEGER,
    FOREIGN KEY (pid) REFERENCES igralec (pid),
    CONSTRAINT x CHECK (x >= -250 AND x <= 250),
    CONSTRAINT y CHECK (y >= -250 AND y <= 250)
);
```

2. Naloga (DML)

▼ Izpišite šifro in ime igralca z največjim naseljem ter šifro, ime in velikost tega naselja

```
select i.pid, i.player, n.vid, n.village, n.population
from igralec i
         inner join naselje n on i.pid = n.pid
where n.population = (select max(population) from naselje);

# razultat
3802,Bogatin,3391,Stibera,1243
```

▼ Izpišite šifre in imena alians, ki imajo maksimalno število članov.

```
select a.aid, a.alliance
from aliansa a
         right join igralec i on a.aid = i.aid
group by a.aid
having count(i.pid) = 60;

# razultat
27,RS-H1N1™
```

▼ Koliko igralcev ima nadpovprečno veliko naselje?

```
select count(i.pid)
from igralec i
         left join naselje n on i.pid = n.pid
where n.population > (select avg(population) from naselje);

# razultat
7224
```

▼ Izpišite podatke o vseh naseljih igralcev brez alianse, urejeno padajoče po x in nato y koordinati.

```
select n.*
from igralec i
         inner join naselje n on i.pid = n.pid
where i.aid is null
order by x desc, y;

# razultat
35105,04Odin,250,171,544,72
38845,05Thor,249,172,330,72
21410,02Slavs,248,171,787,72
26076,New village,247,-244,538,11104
43834,New village,247,168,74,72
...
```

▼ Katero pleme je najštevilčnejše (glede na skupno populacijo)?

```
select p.tid, count(i.pid) as stevilo
from pleme p
         inner join igralec i on p.tid = i.tid
group by p.tid
order by stevilo desc
limit 1;
```

```
# razultat
6,1061
```

▼ Izpišite število alians z nadpovprečnim številom članov

```
select count(res.aid)
from (select a.aid, count(i.pid) as stevilo
      from aliansa a
               inner join igralec i on a.aid = i.aid
      group by a.aid
      having stevilo > (
          select avg(tmp.count)
          from (select count(i.pid) as count
                from aliansa a
                         inner join igralec i on a.aid = i.aid
                group by a.aid) as tmp
      )
     ) as res;

# rezultat
43
```

▼ Napišite shranjeno funkcijo popObmocja(x, y, razdalja), ki za poljubne koordinate vrne populacijo na območju od vključno [x, y] do [x+razdalja, y+razdalja]. Izpišite rezultat za klica popObmocja(40, 40, 10) in popOmbocja(40, 40, 20).

```
def pop_obmocja(x, y, distance):
    cur.execute(
        "select sum(population) from naselje where x > {} and y > {} and x < {} and y < {}".format(x - distance,
                                                                                                   y - distance,
                                                                                                   x + distance,
                                                                                                   y + distance))
    result = cur.fetchone()
    return result[0]

print(pop_obmocja(40, 40, 10)) # izpise: 38065
print(pop_obmocja(40, 40, 20)) # ispise: 168972
```

▼ Izpišite imena igralcev, ki imajo vsa svoja naselja na območju x, ki je med 150 in 200 in y, ki je med 0 in 100.

```
select i.player
from igralec i
where (
          select count(*)
          from igralec ii
                   inner join naselje nn on ii.pid = nn.pid
          where nn.x > 150
            and nn.x < 200
            and nn.y > 0
            and nn.y < 100
            and ii.pid = i.pid
      ) = 0;

# rezultat
Natars
Multihunter
Al ajz on mi
WaRoR
Тута Бугарин
```

▼ Izpišite šifre, imena in delež celotne populacije alians, ki imajo vsaj 3% vse populacije v igri. Rezultat uredite padajoče po deležu

```
select a.aid,
       a.alliance,
       count(i.pid) / sum(n.population) as percentage
from aliansa a
         inner join igralec i on a.aid = i.aid
         inner join naselje n on i.pid = n.pid
group by a.aid
having percentage >= 0.03
order by percentage desc;
```

```
# rezultat
1,TG-TS,0.5000
```

▼ Igralec »Sirena« želi preimenovati vsa svoja naselja na naslednji način. Uredil jih bo po populaciji, najmočnejše bo »Grad 01«, naslednje »Grad 02« in tako dalje. Nalogo lahko rešite v več korakih (zaporedju poizvedb).

```
cur.execute("select vid from igralec i inner join naselje n on i.pid = n.pid where i.player = 'Sirena' order by n.population desc"
ordered_result = cur.fetchall()

for index, res in enumerate(ordered_result):
    vid, = res
    cur.execute("update naselje set village = 'Grad {}' where vid = {}".format("{0:02}".format(index + 1), vid))
cur.commit()
```

3. Naloga (DDL)

▼ Napišite transakcijo (zaporedje ukazov), ki bo združila člane alians HORDA in CAR v novo imenovano alianso HORDA-CAR.

```
start transaction;
# ustvarimo novo alianso, izmislimo si nov aid (123)
insert into aliansa (aid, alliance)
values (123, 'HORDA-CAR');
# ustrezno posodobimo fk zahtevanih igralcev
update igralec i
set i.aid = 123
WHERE i.aid IN (
    select a.aid
    from aliansa a
    where a.alliance = 'HORDA'
       or a.alliance = 'CAR'
);
commit;
```

▼ Napišite bazni prožilec, ki bo ob spremembah vrednosti aid v tableli igralec preveril, če aliansa še lahko sprejme novega člana. Sicer vrnite napako.

```
delimiter //
CREATE TRIGGER before_player_insert
    BEFORE INSERT
    ON igralec
    FOR EACH ROW
BEGIN
    DECLARE player_count INTEGER;
    SET player_count = (
        select count(i.pid) as n
        from aliansa a
                 inner join igralec i on a.aid = i.aid
        WHERE a.aid = NEW.aid
        group by a.aid
    );
    IF player_count >= 60 THEN
        SIGNAL SQLSTATE '45000' SET message_text = 'Player count is limited to max 60 per alliance.';
    END IF;
end //
delimiter ;

# preverimo delovanje z spodnjim insertom (mora vrniti napako)
# izpise: [45000][1644] Player count is limited to max 60 per alliance.
select count(*)
from aliansa a
where a.aid = 27;
INSERT INTO igralec (pid, player, tid, aid)
VALUES (123, 'Test igralec', 7, 27);
```

4. Nalog (ODBC)

▼ Navodilo

V programskem jeziku Python napišite program, ki se priključi na podatkovno bazo in za celotno igralno polje izračuna gostoto populacije in gostoto populacije določenega plemena. Gostoto računajte na območjih velikosti 10×10 polj po formulah:

$$Gostotaprebivalstva = \frac{Skupnapopulacijanaobmocju}{100}$$

$$Gostotaplemena = \frac{Skupnapopulacijaplemenanaobmocju}{100}$$

Rezultate izračunane gostote (za vsako izmed 50×50=2500 območij) shranite nazaj v svojo bazo v tabeli z imenoma gostotaPopulacije in gostotaPlemena. Za izračun gostote plemena lahko izberete poljubno pleme. Priporočljivo je, da naredite Python funkcijo, ki ima parameter ime plemena.

▼ Resitev

```
# ustvarimo tabele pred zacetkom
def init_tables():
    cur = c.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS gostota_populacije ("
                "oid INTEGER PRIMARY KEY, "
                "density NUMERIC)")
    cur.execute("CREATE TABLE IF NOT EXISTS gostota_plemena ("
                "oid INTEGER, "
                "density NUMERIC,"
                "tid INTEGER,"
                "PRIMARY KEY (oid, tid),"
                "FOREIGN KEY (tid) REFERENCES pleme(tid))")


# igralno polje obsega (x,y) koordinate od (-400,-400) do (400,400)
def calculate_population_density(tribe_name=None):
    cur = c.cursor()
    step = 10
    oid = 0
    tid = None
    if tribe_name:
        cur.execute("SELECT tid FROM pleme WHERE tribe = '{}'".format(tribe_name))
        tid_res = cur.fetchone()
        tid = tid_res[0]
    for x in range(-400, 400 - step, step):
        oid += 1
        for y in range(-400, 400 - step, step):
            loc_cond = "x between {} and {} and y between {} and {}".format(x, x + step, y, y + step)
            if tribe_name:
                cur.execute(
                    "SELECT SUM(res.population) FROM "
                    "(SELECT DISTINCT vid, population FROM naselje n "
                    "inner join pleme p join igralec i on i.pid = n.pid where i.tid = {} and {}) as res".format(
                        tid, loc_cond))
            else:
                cur.execute(
                    "SELECT SUM(res.population) FROM "
                    "(SELECT DISTINCT vid, population FROM naselje n "
                    "inner join pleme p join igralec i on i.pid = n.pid where {}) as res".format(
                        loc_cond))
            density = cur.fetchone()[0]
            if density:
                if tribe_name:
                    exec_ignore_duplicate(cur,
                                          "INSERT INTO gostota_plemena VALUES ({}, {}, {})".format(oid, density / 100,
                                                                                                   tid))
                else:
                    exec_ignore_duplicate(cur,
                                          "INSERT INTO gostota_populacije VALUES ({}, {})".format(oid, density / 100))
            cur.commit()

init_tables()
calculate_population_density('Rimljani')
```

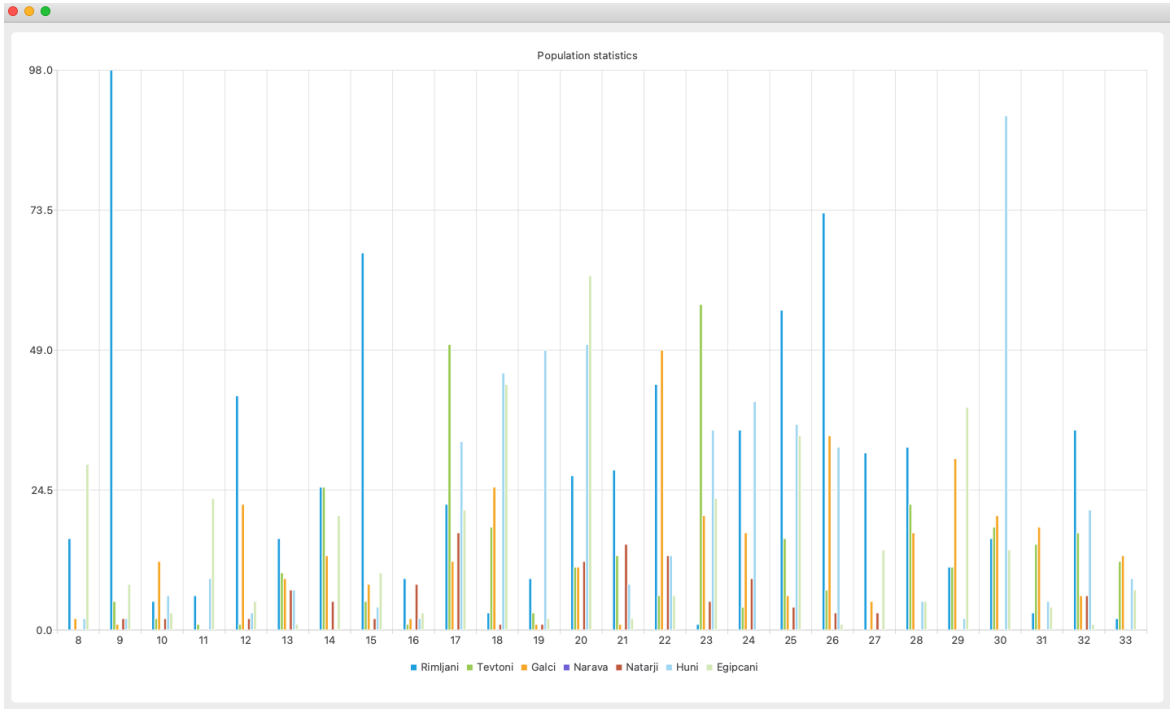
5. Naloga (ODBC)

▼ V Pythonu napišite GUI aplikacijo (Qt ali podobno), ki se priključi na podatkovno bazo in v obliki grafov izriše rezultate izračunane gostote poselitev iz četrte naloge. V okviru te naloge lahko realizirate tudi celotno četrto nalogo, brez shranjevanja vmesnih rezultatov.

```
class StatsWindow(QChartView):

    def __init__(self, *args, **kwargs):
        super(StatsWindow, self).__init__(*args, **kwargs)

        self.tribes = []
        self.oids = []
        self.fetch_data()

        # initialise database connection
        self.c = connect_db()

        self.setRenderHint(QPainter.Antialiasing)

        chart = QChart()
        self.setChart(chart)

        chart.setTitle('Population statistics')

        chart.setAnimationOptions(QChart.SeriesAnimations)

        series = self.getSeries()
        chart.addSeries(series)

        axis = QBarCategoryAxis()
        axis.append(map(lambda x: str(x), self.get_oids()))

        chart.createDefaultAxes()

        chart.setAxisX(axis, series)

        chart.legend().setVisible(True)
        chart.legend().setAlignment(Qt.AlignBottom)

    def getSeries(self):
        series = QBarSeries()
        stats = self.get_population_stats()
        for tid in stats:
            s = QBarSet(self.get_tribe_name(tid))
            s.append(stats.get(tid))
            series.append(s)
        return series

    def fetch_data(self):
        cur = self.c.cursor()
        cur.execute("SELECT tid, tribe from pleme ORDER BY tid ASC")
        self.tribes = cur.fetchall()
        cur.execute("select distinct oid from gostota_plemena UNION select distinct oid from gostota_populacije")
        self.oids = cur.fetchall()

    def get_population_stats(self):
        cur = self.c.cursor()
        m = {}
        cur.execute("select * from gostota_plemena "
                    "UNION "
                    "select oid, density, null from gostota_populacije order by oid")
        results = cur.fetchall()
        for res in results:
            oid, density, tid = res
            if not m.get(oid):
                m[oid] = {}
            m[oid][tid] = density
        res = {}
        for tid in self.get_tribe_ids():
            res[tid] = []
            for oid in self.get_oids():
                res[tid].append(m[oid].get(tid) if m[oid].get(tid) else 0)
        return res

    def get_tribe_name(self, tid):
        return self.get_tribe_names()[tid - 1]

    def get_tribe_names(self):
        return list(map(lambda x: str(x[1]), self.tribes))

    def get_tribe_ids(self):
        return list(map(lambda x: x[0], self.tribes))

    def get_oids(self):
        return list(map(lambda x: x[0], self.oids))


def run_app():
    import sys
    from PyQt5.QtWidgets import QApplication
```

```
app = QApplication(sys.argv)
screen = app.primaryScreen()
size = screen.size()
w = StatsWindow()
w.setGeometry(0, 0, size.width(), size.height())
w.show()
sys.exit(app.exec_())
```

Ko zazenemo program, se nam izrise okno s sledecim grafom:



▼ Iz programa Microsoft Excel se priključite na podatkovno bazo in v obliki grafov izrišite rezultate izračunane gostote poselitev iz četrte naloge.

Ta naloga mi ni uspela, saj sem imel tezave z povezovanjem Excela na MySQL preko ODBC-ja na MacOS-ju.