

Dokumentácia k zadaniu z DBS:

Systém riadenia škôl

Obsah

1. Zadanie.....	3
1.1 Dátový model.....	3
1.2 Analytický report.....	3
1.3 Aplikácia.....	3
2. Špecifikácia scenárov	4
3. Diagram dátového modelu	5
3.1 Fyzický dátový model.....	5
3.2 Logický dátový model.....	6
3.3 Opis modelu	6
3.4 Analytický report.....	7
4. Stručný opis návrhu a implementácie	9
4.1 Implementačné prostredie.....	9
4.2 Návrhové rozhodnutia	9
4.3 Opis balíkov a tried.....	9
4.4 Opis implementácie jednotlivých scenárov	10
4.4.1 Scenár filtrovania	10
4.4.2 Scenár agregácie GROUP BY funkcie.....	11
4.4.3 Scenár pridania záznamu	11
4.4.4 Scenár vymazania záznamu	12
4.4.5 Scenár aktualizovania záznamu	12
4.4.6 Scenár zobrazenia detailov záznamu	13
4.4.7 Scenár pageovania záznamov	13

1. Zadanie

1.1 Dátový model

Logický dátový model musí obsahovať min. 7 tabuliek, bez uvažovania číselníkov. Neodporúčame viac ako 9 takýchto tabuliek. V druhom kontrolnom bode semestra sa odovzdáva dátový model implementovaný v databáze (SQL DDL), s tabuľkami, ktoré sú naplnené dátami

- rádovo státisíce riadkov v každej tabuľke
- Je nám jedno ako ich vygenerujete, ale musí to pôsobiť realisticky
 - Existujú online generátory
 - Rôzne tooly, napr. faker
 - A samotný postgres vie tiež generovať dáta.
- Aspoň jedna tabuľka musí obsahovať rádovo milióny záznamov a musí to byť tabuľka s ktorou sa pracuje v scenári, kde sa filtrujú dáta

1.2 Analytický report

V druhom kontrolnom bode semestra sa odovzdáva aj analytický report, ktorý opisuje dáta jednotlivých tabuliek. Je to dokument, ktorý obsahuje pre každú tabuľku:

- Stručný opis spôsobu, akým boli vygenerované dáta
- Počet riadkov
- Pre každý numerický atribút sumarizáciu min, max, avg, dolný kvartil, medián, horný kvartil (pre istotu: aj foreign key je numerický atribút, primary key netreba riešiť, ak je riešený ako sekvencia)
- Pre každý nominálny atribút početnosti jednotlivých hodnôt
- Minimálne pre tú tabuľku s rádovo miliónmi záznamov aj s uvedením postupu (SQL), akým sa študent k daným číslam dopracoval

1.3 Aplikácia

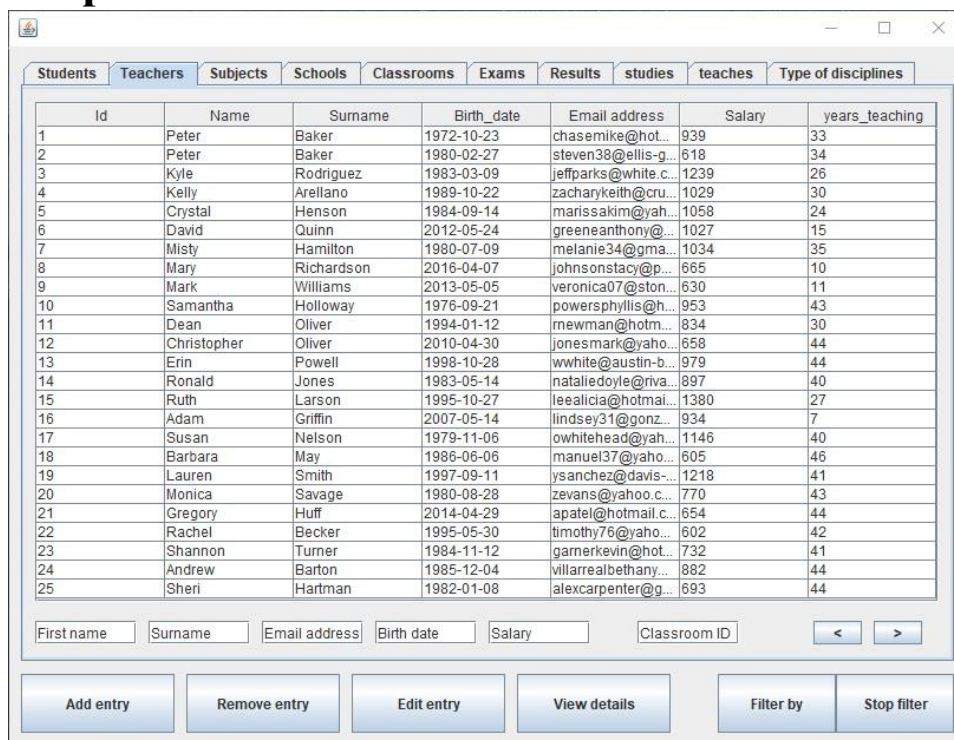
V poslednom kontrolnom bode musí aplikácia realizovať tieto všeobecné scenáre (a podscenáre):

- Zobrazenie prehľadu všetkých záznamov (rozumne stránkovaných, ak je to potrebné), kde v každom riadku figuruje aj číslo, pre ktorého výpočet sa využíva GROUP BY,
 - extend podscenár: Filtrovanie záznamov spĺňajúcich určité kritériá zadané používateľom.
- Zobrazenie konkrétneho záznamu - detail,
- Vytvorenie nového záznamu,
- Aktualizácia existujúceho záznamu,
- Vymazanie záznamu.

Aplikácia musí mať grafické rozhranie (aplikácia **nemôže** byť realizovaná ako konzolová). Je dôležité aby scenáre boli realizované realisticky - teda aby aplikácia (a teda aj jej používateľské rozhranie) naozaj poskytovala časť funkcionality tak, ako by ju očakával zákazník v danej doméne (napr. namiesto zadavania ID záznamu do modálneho okna si záznam vyberiem kliknutím v tabuľke).

Scenáre, ktoré menia dáta musia byť realizované **s použitím transakcií** (explicitné transakcie, autocommit) a aspoň jeden z nich musí zahŕňať **prácu s viacerými tabuľkami** (typicky vytvorenie záznamu a naviazanie cudzieho kľúča).

2. Špecifikácia scenárov



Id	Name	Surname	Birth_date	Email address	Salary	years_teaching
1	Peter	Baker	1972-10-23	chasemike@hot...	939	33
2	Peter	Baker	1980-02-27	steven38@ellis-g...	618	34
3	Kyle	Rodriguez	1983-03-09	jeffparks@white.c...	1239	26
4	Kelly	Arellano	1989-10-22	zacharykeith@cru...	1029	30
5	Crystal	Henson	1984-09-14	marissakim@yah...	1058	24
6	David	Quinn	2012-05-24	greeneanthony@...	1027	15
7	Misty	Hamilton	1980-07-09	melanie34@gma...	1034	35
8	Mary	Richardson	2016-04-07	johnsonstacy@p...	665	10
9	Mark	Williams	2013-05-05	veronica07@ston...	630	11
10	Samantha	Holloway	1976-09-21	powersphylis@h...	953	43
11	Dean	Oliver	1994-01-12	mewman@hotmail...	834	30
12	Christopher	Oliver	2010-04-30	jonesmark@yaho...	658	44
13	Erin	Powell	1998-10-28	wwhite@austin-b...	979	44
14	Ronald	Jones	1983-05-14	nataliedoyle@riva...	897	40
15	Ruth	Larson	1995-10-27	leealicia@hotmail...	1380	27
16	Adam	Griffin	2007-05-14	lindsey31@gonz...	934	7
17	Susan	Nelson	1979-11-06	owhitehead@yah...	1146	40
18	Barbara	May	1986-06-06	manuel37@yaho...	605	46
19	Lauren	Smith	1997-09-11	ysanchez@davis...	1218	41
20	Monica	Savage	1980-08-28	zevans@yahoo.c...	770	43
21	Gregory	Huff	2014-04-29	apatel@hotmail.c...	654	44
22	Rachel	Becker	1995-05-30	timothy76@yaho...	602	42
23	Shannon	Turner	1984-11-12	garnerkevin@hot...	732	41
24	Andrew	Barton	1985-12-04	villarrealbethany...	882	44
25	Sheri	Hartman	1982-01-08	alexcarpenter@g...	693	44

V hornej časti používateľského rozhrania je možné pristupovať ku všetkým tabuľkám vrátane väzobných tabuľiek prostredníctvom **kariet**. Pod každou označenou tabuľkou sa nachádzajú tlačidlá na **listovanie** a **textfieldy** pre všetky atribúty tabuľky. Pomocou nich je možné:

- Vytvoriť nový záznam tlačidlom **Add entry**
- Aktualizovať existujúci záznam tlačidlom **Edit entry**
- Filtrovať záznamy, ktoré spĺňajú určité kritériá zadané používateľom tlačidlom **Filter by**:
 - Filtrovanie atribútov typu string – zadáva sa substring podľa ktorého aplikácia nájde všetky záznamy obsahujúce daný substring
 - Filtrovanie atribútov typu integer – zadáva sa operačný znak (<, >, = a pod.) určujúci porovnanie s následne daným číslom
 - Filtrovanie atribútov typu date – zadáva sa operačný znak (<, >, = a pod.) určujúci porovnanie s následne daným dátumom vo formáte 'yyyy-MM-dd' (s apostrofom vrátane). Na zastavenie filtra slúži tlačidlo **Stop filter**

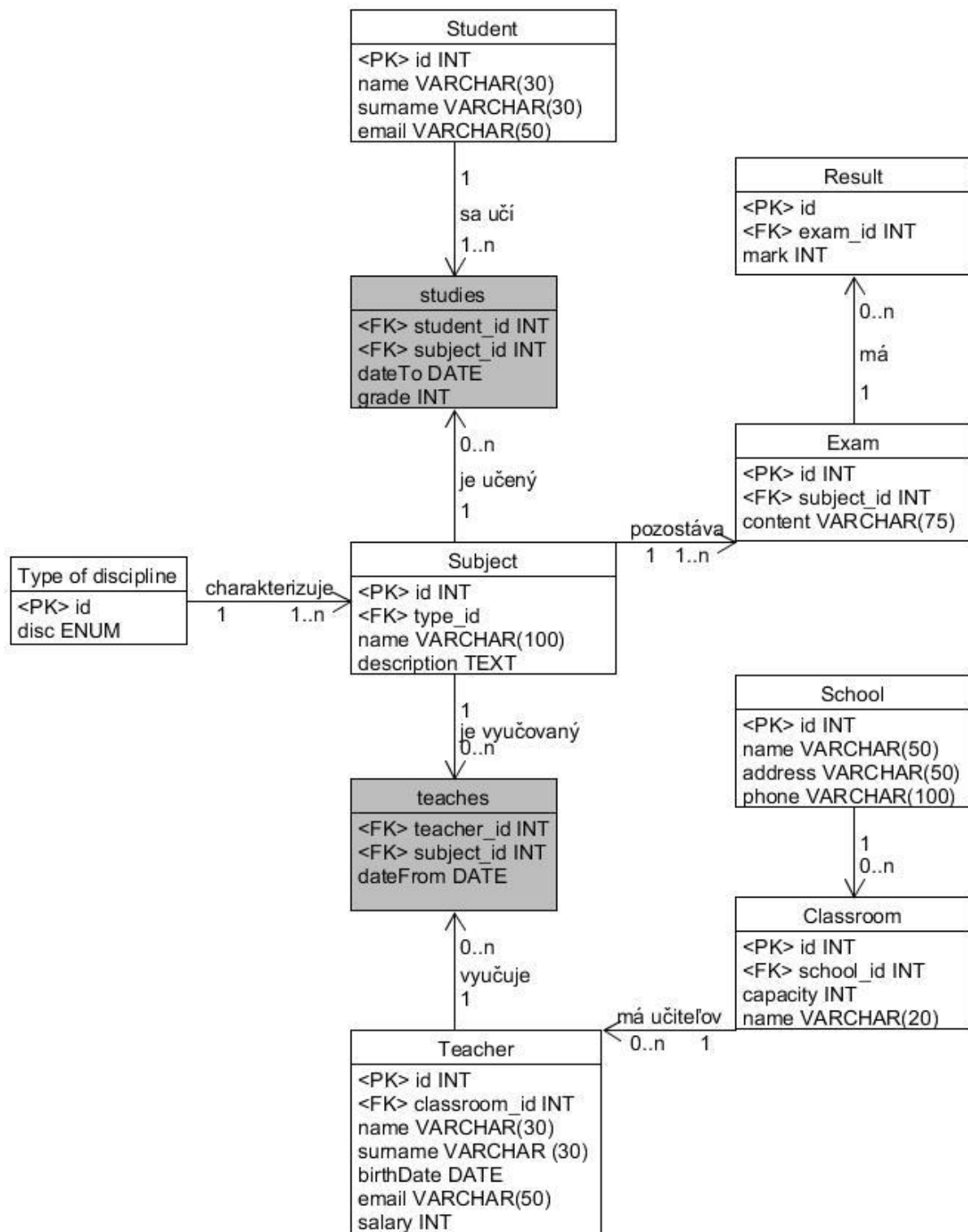
V tabuľke je možné vybrať riadok záznamu, s ktorým následne môžeme urobiť tieto operácie:

- vymazanie pomocou tlačidla **Remove entry** (vymazávanie funguje na kaskádovom princípe. Ak sa vymaže študent, vymažu sa všetky jeho štúdiá vo väzobnej tabuľke studies, ktorý používa jeho ID ako foreign key)
- zobrazenie detailnejších informácií pomocou tlačidla **View details**
- modifikovanie už spomínaným tlačidlom **Edit entry**

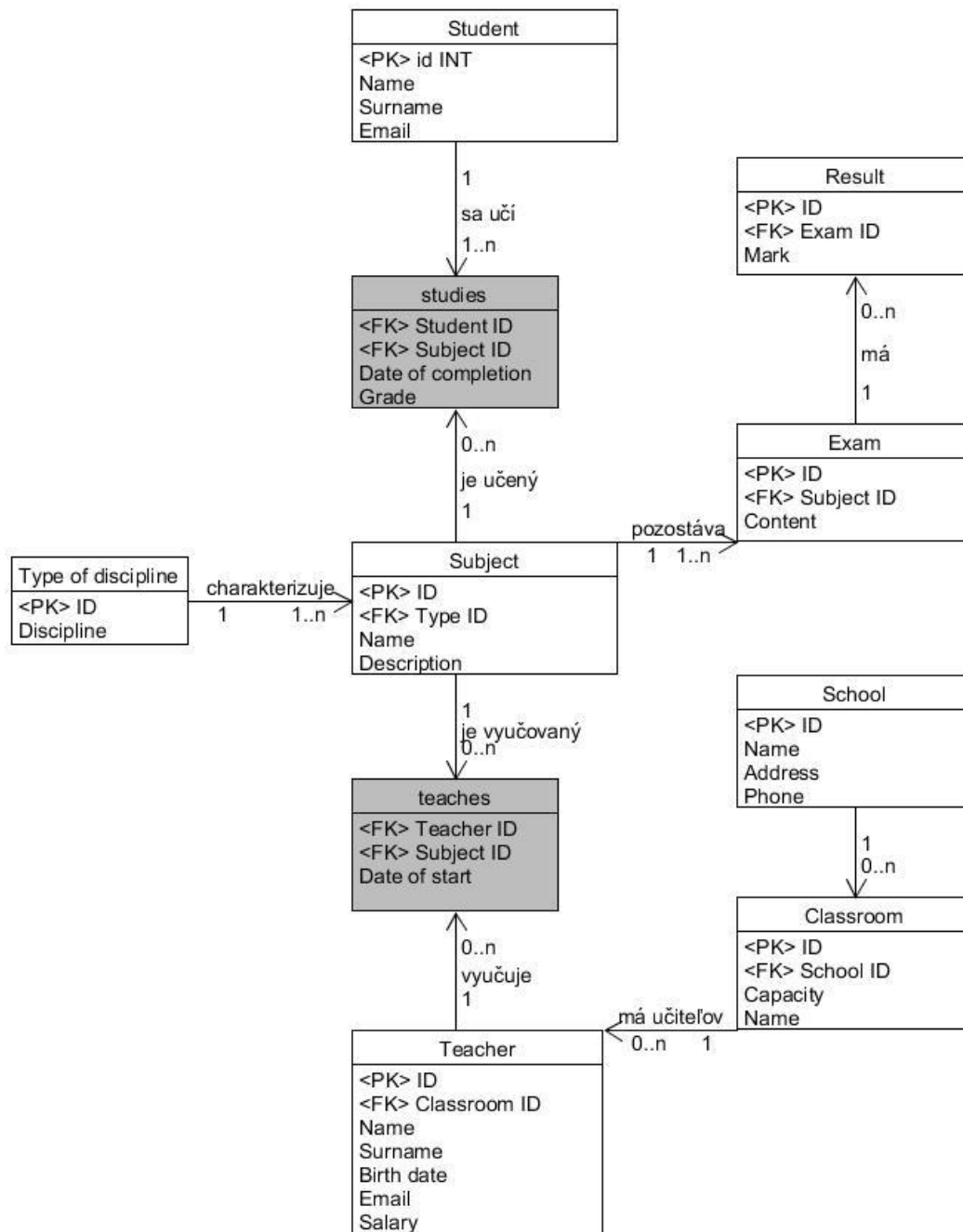
V každej tabuľke je zahrnutý posledný stĺpec (prípadne na posledných dvoch), v ktorom figuruje číslo vypočítané agregáčnou funkciou pri ktorej sa využíva GROUP BY.

3. Diagram datového modelu

3.1 Fyzický datový model



3.2 Logický dátový model

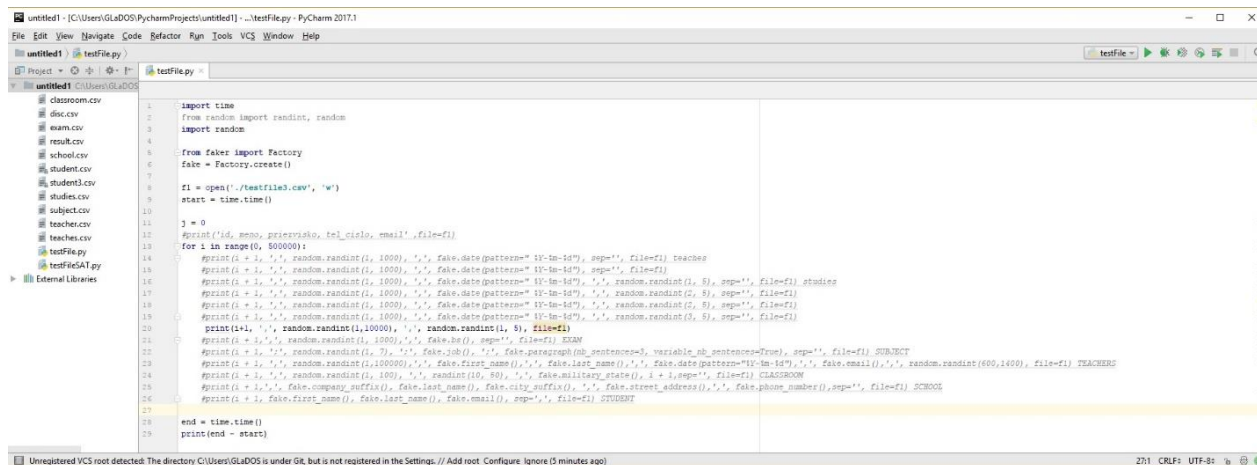


3.3 Opis modelu

Škola obsahuje niekoľko tried. V triede učí niekoľko učiteľov, pričom každý jeden patrí len do jednej. Učiteľ môže učiť viac predmetov, pričom jeden predmet môže byť učený viacerými učiteľmi. Predmet je učený viacerými študentami, pričom každý študent môže študovať viac predmetov. Každému predmetu je priradený typ disciplíny, ktorým sa zaoberá. Každý predmet pozostáva z niekoľkých písomiek. Každá písomka obsahuje niekoľko výsledkov študentov.

3.4 Analytický report

Dáta sa generovali prostredníctvom python faker knižnice ktorá obsahovala množstvo potrebných funkcií ako mená, dátum, adresy, telefónne čísla, emaily a texty. Primárne kľúče sa vytvárali sekvenčne za sebou v každej tabuľke nezávisle.



```
1 import time
2 from random import randint, random
3 import random
4
5 from faker import Factory
6 fake = Factory.create()
7
8 f1 = open('testFile3.csv', 'w')
9 start = time.time()
10
11 j = 0
12 #print('id, meno, priezvisko, tel_cislo, email', file=f1)
13 for i in range(0, 500000):
14     #print(i + 1, ',', random.randint(1, 1000), ',', fake.date(pattern='Y-m-d'), sep='', file=f1) teachers
15     #print(i + 1, ',', random.randint(1, 1000), ',', fake.date(pattern='Y-m-d'), sep='', file=f1)
16     #print(i + 1, ',', random.randint(1, 1000), ',', fake.date(pattern='Y-m-d'), ',', random.randint(1, 5), sep='', file=f1) students
17     #print(i + 1, ',', random.randint(1, 1000), ',', fake.date(pattern='Y-m-d'), ',', random.randint(1, 5), sep='', file=f1)
18     #print(i + 1, ',', random.randint(1, 1000), ',', fake.date(pattern='Y-m-d'), ',', random.randint(1, 5), sep='', file=f1)
19     #print(i + 1, ',', random.randint(1, 1000), ',', fake.date(pattern='Y-m-d'), ',', random.randint(1, 5), sep='', file=f1)
20     #print(i + 1, ',', random.randint(1, 10000), ',', random.randint(1, 5), file=f1)
21     #print(i + 1, ',', random.randint(1, 1000), ',', fake.date(pattern='Y-m-d'), sep='', file=f1)
22     #print(i + 1, ',', random.randint(1, 1000), ',', fake.date(pattern='Y-m-d'), ',', fake.date(pattern='Y-m-d'), ',', random.randint(600, 1400), file=f1) TEACHERS
23     #print(i + 1, ',', random.randint(1, 100000), ',', fake.first_name(), ',', fake.last_name(), ',', fake.date(pattern='Y-m-d'), ',', random.randint(600, 1400), file=f1) TEACHERS
24     #print(i + 1, ',', random.randint(1, 100), ',', random.randint(10, 50), ',', fake.date(pattern='Y-m-d'), ',', random.randint(600, 1400), file=f1) TEACHERS
25     #print(i + 1, ',', random.randint(1, 100000), ',', fake.first_name(), ',', fake.last_name(), ',', fake.date(pattern='Y-m-d'), ',', random.randint(600, 1400), file=f1) TEACHERS
26     #print(i + 1, ',', random.randint(1, 100000), ',', fake.first_name(), ',', fake.last_name(), ',', fake.date(pattern='Y-m-d'), ',', random.randint(600, 1400), file=f1) TEACHERS
27
28 end = time.time()
29 print(end - start)
```

School:

- počet vygenerovaných skôl je 100 000
- atribút školy bola poskladaná z 3 rôznych funkcií pre realistickejší názov, na ostatné atribúty (name, address, phone) sa použili fake funkcie

Classroom

- počet vygenerovaných tried je 100 000
- triedna miestnosť obsahuje kapacitu ktorá bola vygenerovaná číslom od 10-50. Cudzie kľúče ktoré predstavujú id pre školu boli generované od 1-100, priemerne to je na školu 1000 tried, na atribút name sa použila jedna fake funkcia
- hodnoty číselného atribútu kapacity (capacity):
 - o min = 10, avg = 29.98027, max = 50
 - o kvartily: dolný = 20, medián = 30, horný = 40

Teacher

- počet vygenerovaných učiteľov je 100 000
- každý učiteľ má mesačnú mzdu (salary), ktorá bola vygenerovaná číslom od 600-1400. Cudzie kľúče ktoré predstavujú triednu miestnosť boli generované od 1-100 000, teda priemerný počet učiteľov na triedu je 1, na ostatné atribúty (name, surname, birthDate, email) sa použili fake funkcie
- hodnoty číselného atribútu mzdy (salary):
 - o min = 600, avg = 999.9004, max = 1400
 - o kvartily: dolný = 800, medián = 999, horný = 1199

Student

- počet vygenerovaných študentov je 1 000 000
- na každý atribút (name, surname, email) sa použili fake funkcie

Type_of_discipline

- počet vygenerovaných disciplín je 7
- obsahuje atribút typu enumerate, ktorý obsahuje 7 rôznych hodnôt (technical, natural, medical, philosophical, artistic, pedagogic, economic)

Subject

- počet vygenerovaných predmetov je 1000
- cudzie kľúče, ktoré predstavujú typ disciplíny boli generované náhodne od 1-7, názov predmetu teda nekorešponduje s typom disciplíny. Na ostatné atribúty (name, description) sa použili fake funkcie

Exam

- počet vygenerovaných písomiek je 10 000
- cudzie kľúče, ktoré predstavujú predmet boli generované od 1-1000, teda priemerný počet písomiek na predmet je 10. Na atribút obsahu látky písomky (content) sa použila fake funkcia

Result

- počet vygenerovaných výsledkov je 500 000
- na atribút známky (mark) sa použilo generovanie čísla od 1-5. Cudzie kľúče, ktoré predstavujú písomku boli generované od 1-10 000, priemerný počet výsledkov na písomku je 50.
- hodnoty číselného atribútu známky (mark):
 - o min = 1, avg = 2.997408, max = 5
 - o kvartily: dolný = 2, medián = 3, horný = 4

Studies

- počet vygenerovaných študovaných predmetov je 400 000
- na atribút výslednej známky (grade) sa generovalo číslo od 1-5, pričom 1 mala menšiu pravdepodobnosť, ako ostatné výsledné známky. Pri väzobnej entity sa použili 2 cudzie kľúče, prostredníctvom ktorých sa pre každého študenta s id od 1-100 000 generoval 4 krát cudzí kľúč predmetu od 1-1000. Teda každý študent študuje 4 predmety, pričom každý predmet bude mať priemerne 400 študentov. Na posledný atribút dátumu ukončenia výučby (dateTo) sa použila fake funkcia.
- hodnoty číselného atribútu konečnej známky (grade):
 - o min = 1, avg = 3.499541, max = 5
 - o kvartily: dolný = 3, medián = 4, horný = 5

Teaches

- počet vygenerovaných vyučovaných predmetov je 200 000
- pri väzobnej entity sa použili dva cudzie kľúče, prostredníctvom ktorých sa pre každého učiteľa s id od 1-100 000 generoval 2 krát cudzí kľúč predmetu od 1-1000. Každý učiteľ vyučuje 2 predmety, teda každý predmet bude mať priemerne 200 učiteľov. Na posledný atribút dátumu nástupu do zamestnania (dateFrom) sa použila fake funkcia.

4. Stručný opis návrhu a implementácie

4.1 Implementačné prostredie

Implementačným prostredím je jazyk JAVA s vývojovým prostredím Eclipse s verziou Jee neon.2 spolu s databázovým serverom PostgreSQL, ku ktorému sa pristupovalo cez pgAdmin 3.

4.2 Návrhové rozhodnutia

K databáze sa pristupuje v hlavnej metóde main, v ktorom sa pomocou knižnice org.postgresql.ds.PGPoolingDataSource vytvorí inštancia source, ktorej sa následne nastaví všetky parametre potrebné na pripojenie sa k správnej databáze. Do databázy sa pripája a odpája pri každej novej požiadavke používateľa prostredníctvom metód manažérskej triedy tej tabuľky, s ktorou používateľ aktuálne pracuje. Všetky tieto manažérske triedy tabuliek dedia abstraktnú triedu, ktorá obsahuje potrebné metódy na spracovanie selektov. Nachádzajú sa v balíku **managers**.

4.3 Opis balíkov a tried

4.3.1 Balík managers

Balík tried pre manažovanie tabuliek obsahujú 2 rôzne typy metód. Jeden na rozdiel od druhého slúži pre scenáre aktuálne filtrovaných záznamov. V metóde sa odlišuje tým, že má ako argument štruktúru príslušnej tabuľky ktorá obsahuje dáta podľa ktorých sa má filtrovať následný select. Po selekte získame ID vybraného zoznamu s ktorým pracujeme rovnako ako bez filtra.

Výnimkou sú metódy, ktoré majú v argumentoch už nájdené ID záznamu, vypočítavajú len hodnoty pre už spomínané posledné stĺpce tabuľky (agregačné funkcie s GROUP BY) alebo view details.

Pre listovanie databázou je potrebné mať atribút, ktorý bude slúžiť ako offset v SQL query metódach. Pre efektívne prepínanie medzi filtrovanými záznamami a nefiltrovanými nachádzajúcich sa na odlišných "stranách" sa taktiež vytvorili dvojvariantové atribúty (offsety) a metódy (settery) pre ne.

4.3.2 Balík tableModel

Balík obsahuje triedy pre všetky tabuľky, do ktorých je možné zasahovať používateľom. Číselník Type of disciplines pre tabuľku subject sa tam teda nenachádza. Tieto triedy obsahujú všetky údaje tabuliek ktoré majú vystupovať pri zobrazovaní v používateľskom rozhraní. Obsahujú teda všetky atribúty fyzického dátového modelu. V triedach sú vytvorené ošetrené gettery a settery tak, aby keď používateľ nezadá do textfieldu žiadnu hodnotu, systém to bude považovať za ľubovoľný údaj (v prípade filtrovania) alebo nevyplnený údaj (v prípade pridávania/aktualizovania záznamu).

4.3.3 Balík guiModel

Tak ako v predošlom balíku, tento balík obsahuje triedy pre všetky tabuľky, do ktorých je možné zasahovať používateľom. Slúžia na vytvorenie dynamicky sa správajúcich tabuliek v grafickom používateľskom rozhraní, ktorý je predstavený triedou GUI. Tieto triedy používajú prostredníctvom dedenia štruktúru prebranú z knižnice javax.swing.table.AbstractTableModel, ktorej modifikujú mená stĺpcov a pomocné metódy na spracovanie dát dvojrozmerným polom. (prvý index zľava určuje poradie záznamov, druhý index určuje poradie stĺpcov príslušnej tabuľky)

4.4 Opis implementácie jednotlivých scenárov

4.4.1 Scenár filtrovania – ešte pred testovaním podmienok (ktoré hľadajú príslušnú kartu) sa nastavuje atribút filter typu boolean.

```
if(tabbedPane.getSelectedIndex()==1) {  
    filterTeacher = getTeacherData();  
    setTeacherDisplay(scrollTeacher);  
}
```

keď sa filtruje, nastaví sa na true

```
if(tabbedPane.getSelectedIndex()==1)  
    setTeacherDisplay(scrollTeacher);
```

keď sa odfiltruje, nastaví sa na false

Metóda getTeacherData získa dáta z textfielddov do inšancie filterTeacher štruktúry Teacher a zavolá metódu setTeacherDisplay (ktorá sa volá vždy po vykonaní ktoréhokolvek scenáru pre aktualizáciu zobrazenia).

```
public void setTeacherDisplay(JScrollPane scrollPane) throws SQLException {  
    Object[][] data = new Object[25][7];  
    int i = 0;  
    if(!filter)  
        for (Teacher item : teacherM.getAllTeachers()) {  
            data[i][0] = item.getId();  
            data[i][1] = item.getName();  
            data[i][2] = item.getSurname();  
            data[i][3] = item.getBirthDate();  
            data[i][4] = item.getEmail();  
            data[i][5] = item.getSalary();  
            data[i][6] = teacherM.getDuration(data[i+1][0]);  
        }  
    else {  
        for(Teacher item : teacherM.getAllTeachers(filterTeacher)) {  
            data[i][0] = item.getId();  
            data[i][1] = item.getName();  
            data[i][2] = item.getSurname();  
            data[i][3] = item.getBirthDate();  
            data[i][4] = item.getEmail();  
            data[i][5] = item.getSalary();  
            data[i][6] = teacherM.getDuration(data[i+1][0]);  
        }  
    }  
    tableTeacher = new JTable();  
    scrollPane.setViewportView(tableTeacher);  
    tableTeacher.setModel(new ModelTeacher(data));  
}
```

Tá nám načítá dáta tabuľky na jednu stranu, ktorá obsahuje 25 záznamov. Ak máme filter zapnutý, zavolá sa metóda manažérskej triedy getAllTeachers s argumentom filterTeacher, podľa ktorého sa filtruje, začne načítavať dáta a zobrazí ich pomocou metódy tabuľky setModel, ktorá má ako argument triedu, ktorá reprezentuje model údajov Teachera spolu s dátami, ktoré sa uložili prostredníctvom predchádzajúceho for cyklu.

```
@SuppressWarnings("unchecked")  
public List<Teacher> getAllTeachers() throws SQLException {  
    return(selectQuery("SELECT * FROM teacher LIMIT 25 OFFSET " + inkrement));  
}  
  
@SuppressWarnings("unchecked")  
public List<Teacher> getAllTeachers(Teacher t) throws SQLException {  
    return(selectQuery("SELECT * FROM teacher "  
        + "WHERE name LIKE '%" + t.getName() + "%' AND surname LIKE '%" + t.getSurname()  
        + "%' AND birthdate " + t.getBirthDate() + " AND email LIKE '%" + t.getEmail()  
        + "%' AND salary " + t.getSalary() + " AND classroom_id " + t.getClassId()  
        + " LIMIT 25 OFFSET " + filterInkrement));  
}
```

4.4.2 Scenár agregačnej GROUP BY funkcie

Na predchádzajúcom obražku v poslednom riadku for cyklu sa zavolá metóda manažéra triedy Teacher getDuration, ktorý dostane ako argument ID aktuálne prebiehajúceho záznamu.

Táto metóda vykoná nasledujúci SQL príkaz.

```
String createStatementString = "SELECT max(current_date - dateFrom)/365 AS yearAmount "
    + "FROM teaches "
    + "GROUP BY teaches.teacher_id "
    + "HAVING teaches.teacher_id = " + data;
```

4.4.3 Scenár pridania záznamu

Vytvorí sa nová inštancia teacher, do ktorej načítame dáta z textfieldu pomocou metódy `getTeacherData`. Následne sa zavolá manažérska metóda `insertTeacher`

```
if(tabbedPane.getSelectedIndex()==1) {
    Teacher teacher = getTeacherData();
    if(teacherM.insertTeacher(teacher))
        JOptionPane.showMessageDialog(null, "New subject has been created successfully.");
}
```

Najprv sa vytvorí SQL query, ktorý postupne generuje ID automaticky od najvyššieho existujúceho ID čísla. Ostatné atribúty stĺpcov sa dosadia až potom pomocou Statement metód set, pre ktoré sa najprv museli stringy z textfieldov rozparsovať, prípadne formátovať do SQL typov.

```
String createStatementString = "INSERT INTO teacher(id, classroom_id, name, surname, birthdate, email, salary) "
    + "VALUES(nextval('teacher_id_seq'),?,?,?,?);";
stmt = (PreparedStatement) conn.prepareStatement(createStatementString);
int salara = Integer.parseInt(s.getSalary());
int FK = Integer.parseInt(s.getClassId());

SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
Date parsed = format.parse(s.getBirthDate());
java.sql.Date sql = new java.sql.Date(parsed.getTime());

stmt.setInt(1, FK);
stmt.setString(2, s.getName());
stmt.setString(3, s.getSurname());
stmt.setDate(4, sql);
stmt.setString(5, s.getEmail());
stmt.setInt(6, salara);
stmt.executeUpdate();
conn.commit();
```

4.4.4 Scenár vymazania záznamu

Zavolá sa funkcia deleteRow/deleteFilterRow.

```
if(tabbedPane.getSelectedIndex()==1) {           // Teachers
    if(!filter) {
        if(!teacherM.deleteRow(tableTeacher.getSelectedRow()))
            JOptionPane.showMessageDialog(null, "Select subject for deleting first.");
        else
            setTeacherDisplay(scrollTeacher);
    }
    else {
        if(!teacherM.deleteFilterRow(tableTeacher.getSelectedRow(), filterTeacher))
            JOptionPane.showMessageDialog(null, "Select subject for deleting first.");
        else
            setTeacherDisplay(scrollTeacher);
    }
}
```

Ktorá pozostáva z dvoch SQL príkazov (ak sa jedná o zfiltrovaný záznam, v prvom sa vyselektuje ID aj s WHERE, kde vystupuje filterTeacher, podľa ktorého sa filtruje)

```
String createStatementString = "SELECT id FROM teacher "
+                               "WHERE name LIKE '%" + filter.getName() + "%' AND surname LIKE '%" + filter.getSurname()
+                               "%' AND birthdate " + filter.getBirthDate() + "%' AND email LIKE '%" + filter.getEmail()
+                               "%' AND salary " + filter.getSalary() + "%' AND classroom_id " + filter.getClassId()
+                               "LIMIT 1 OFFSET " + (x+filterInkrement);
Statement st = conn.createStatement();
ResultSet rs = st.executeQuery(createStatementString);
int id = 0;
if(rs.next()) {
    id = rs.getInt("id");
}
createStatementString = "DELETE FROM teacher WHERE id = " + id;
```

4.4.5 Scenár aktualizovania záznamu

Aktivuje sa podobne ako pri scenári vymazania záznamu, len stým rozdielom, že pred filtračnou podmienkou sa načítajú dáta z textfieldov do inštancie teacher, ktorý sa pošle do argumentu funkcie editRow/editFilterRow. Tieto funkcie sa od predošlej líši len druhým query príkazom, v ktorých vystupujú atribúty tabuľky, do ktorých sa pomocou už spomínanej statement funkcie načítajú dáta z inštancie teacher.

```
String createStatementString = "SELECT id FROM teacher "
+                               "WHERE name LIKE '%" + filter.getName() + "%' AND surname LIKE '%" + filter.getSurname()
+                               "%' AND birthdate " + filter.getBirthDate() + "%' AND email LIKE '%" + filter.getEmail()
+                               "%' AND salary " + filter.getSalary() + "%' AND classroom_id " + filter.getClassId()
+                               "LIMIT 25 OFFSET " + (x+filterInkrement);
Statement st = conn.createStatement();
ResultSet rs = st.executeQuery(createStatementString);
int id = 0;
if(rs.next())
    id = rs.getInt("id");
createStatementString = "UPDATE teacher SET name = ?, surname = ?, email = ?, "
+                       "birthdate = ?, salary = ?, classroom_id = ? WHERE id = " + id;
int idClass = Integer.parseInt(s.getClassId());
int salary = Integer.parseInt(s.getSalary());
SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
Date parsed = format.parse(s.getBirthDate());
java.sql.Date sql = new java.sql.Date(parsed.getTime());
stmt = (PreparedStatement) conn.prepareStatement(createStatementString);
stmt.setString(1, s.getName());
stmt.setString(2, s.getSurname());
stmt.setString(3, s.getEmail());
stmt.setDate(4, sql);
stmt.setInt(5, salary);
stmt.setInt(6, idClass);
```

4.4.6 Scenár zobrazenia detailov záznamu

Najprv sa zavolá funkcia, ktorá vráti číslo ID označeného riadku tabuľky. Následne sa zavolá manažérska funkcia viewDetail, ktorá vráti string, v ktorom sú všetky dodatočné informácie o zázname. String sa zobrazí v novom okne cez metódu showMessageDialog.

```
if(tabbedPane.getSelectedIndex()==1) {           // Teachers
    if(!filter)
        idView = teacherM.getId(tableTeacher.getSelectedRow());
    else
        idView = teacherM.getId(tableTeacher.getSelectedRow(), filterTeacher);
    if(idView == -1)
        JOptionPane.showMessageDialog(frame, "Select entry row for viewing first.");
    else
        JOptionPane.showMessageDialog(null, teacherM.viewDetail(idView));
}
```

4.4.7 Scenár pageovania záznamov

Zavolá sa funkcia na nastavenie offsetu o +25 alebo -25 (keď sa stlačí spätná šípka) a následne sa zavolá setTeacherDisplay na zaktualizovanie údajov v používateľskom rozhraní tabuľky.

```
JButton button_1 = new JButton(">");
button_1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        if(filter) teacherM.setfilterInkrement(25);
        else      teacherM.setInkrement(25);
        try {
            setTeacherDisplay(scrollTeacher);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
});
```