ΑΝΑΦΟΡΑ

ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΣΚΗΣΗΣ

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

ΜΕΡΟΥΣ Β

Μανουδάκη Χρυσηίς 2019030201

Αγγελίδης Νικόλαος 2019030190

● ΕΡΩΤΗΜΑΤΑ

Συνέχεια των ερωτημάτων της πρώτης φάσης που βρίσκονταν σε αστερίσκους και υλοποίηση των views.

3. Ανάκτηση δεδομένων και υπολογισμοί (υλοποίηση με χρήση συναρτήσεων postgreSQL)

3.8. (*)Χρήση συναρτήσεων:

find_thesis_titles_3_8() : Συνάρτηση η οποία δημιουργεί δύο προσωρινούς πίνακες, ο πρώτος πίνακας εμπεριέχει την συνένωση τον πινάκων Diploma, Committee, Professor με βάση το AMKA. Στη συνέχεια με μια for loop διατρέχουμε τον πίνακα αυτόν ως προς το diplomanum και βρίσκουμε αν τα μέλη της επιτροπής εργάζονται στο ίδιο εργαστήριο για το συγκεκριμένο diplomanum.

```
CREATE OR REPLACE FUNCTION public.find_thesis_titles_3_8(
    )
    RETURNS TABLE(thesis_title character varying)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
declare
_row integer;
BEGIN
    drop table if exists thesis_titles cascade;
    create temp table thesis_titles(thesis_title character varying);

    drop table if exists diplomas_and_supervisors cascade;
    create temp table diplomas_and_supervisors(thesis_title character varying, diploma_num integer, student_amka character varying, prof_amka character varying, labjoins integer);
    insert into diplomas_and_supervisors
    select DS.thesis_title, diploma_num, student_amka, prof_amka, labjoins
    from ("Diploma" Dl natural join "Committee" Cm) as DS join "Professor" as Pr on DS.prof_amka = Pr.amka;

    for _row in select distinct diploma_num from "diplomas_and_supervisors"
    loop
        if (select count(S1.thesis_title)
            from "diplomas_and_supervisors" S1, "diplomas_and_supervisors" S2
            where S1.diploma_num =_row and S2.diploma_num =_row and S1.labjoins < S2.labjoins) = 0 then

                insert into thesis_titles values ((select D.thesis_title from "Diploma" D where diploma_num = _row));
        end if;
    end loop;

    RETURN QUERY select * from thesis_titles;
end;
$BODY$;

ALTER FUNCTION public.find_thesis_titles_3_8()
    OWNER TO postgres;
```

3.9. (*) Χρήση συναρτήσεων:

get_related_courses_3_9(): Συνάρτηση όπου με την χρήση αναδρομής ψάχνει για ένα μάθημα, το οποίο δίνεται ως όρισμα κατά την κλήση της συνάρτησης, τα προαπαιτούμενα και συνιστώμενα μαθήματα του συγκεκριμένου μαθήματος.

```
CREATE OR REPLACE FUNCTION public.get_related_courses_3_9(
    course_c character)
    RETURNS TABLE(course_code character, course_title character)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
BEGIN
    RETURN QUERY
    With Recursive
        R(d, m) as
        (select dependent as d , main as m from "Course_depends"
        union
        select R.d, "Course_depends".main as m
        from R, "Course_depends"
        where R.m = "Course_depends".dependent)
    select c.course_code, c.course_title from R inner join "Course" c on m = c.course_code
    where d = course_c;
end;
$BODY$;

ALTER FUNCTION public.get_related_courses_3_9(character)
    OWNER TO postgres;
```

6. Λειτουργικότητα με χρήση όψεων (views)

6.1. (*) Χρήση συναρτήσεων:

committee_names_6_1(diplomanum integer): Συνάρτηση που επιστρέφει μια text μεταβλητή, η οποία περιέχει μέσα τους καθηγητές που ανήκουν στα υπόλοιπα μέλη της επιτροπής.

view_6_1: Μέσα στο συγκεκριμένο view αναθέτουμε το ΑΜΚΑ του κάθε μαθητή στην πρώτη στήλη ενώ στην δεύτερη μπαίνει ο επιβλέπων καθηγητής και μετά από αυτόν επισυνάπτονται μέσω της κλήσης της προαναφερόμενης συνάρτησης τα υπόλοιπα μέλη της επιτροπής.

```
CREATE OR REPLACE FUNCTION public.committee_names_6_1(
    diplomanum integer)
    RETURNS text
    LANGUAGE 'plpgsql'
    COST 100
    IMMUTABLE PARALLEL UNSAFE
AS $BODY$
declare
committeeNames text;
_row record;
BEGIN
    for _row in select Ps.surname, Ps.name
                from "Committee" Ct join "Person" Ps on Ct.prof_amka= Ps.amka
                where supervisor = 'false' and diploma_num = diplomaNum
    loop
        committeeNames := concat(committeeNames, ', ', _row.surname::text, ' ', _row.name::text);
    end loop;
    RETURN committeeNames;
end;
$BODY$;

ALTER FUNCTION public.committee_names_6_1(integer)
    OWNER TO postgres;


CREATE OR REPLACE VIEW public.view_6_1
 AS
 SELECT super.student_amka,
    concat(super.supername, committee_names_6_1(super.diploma_num)) AS committee_names
   FROM ( SELECT dct.diploma_num,
            dct.student_amka,
            concat(ps.surname::text, ' ', ps.name::text) AS supername
           FROM ("Diploma" d
             JOIN "Committee" ct USING (diploma_num, student_amka)) dct
             JOIN "Person" ps ON dct.prof_amka::text = ps.amka::text
          WHERE dct.graduation_date IS NULL AND dct.supervisor = true) super;

ALTER TABLE public.view_6_1
    OWNER TO postgres;
```

6.2. (*) Χρήση συναρτήσεων:

get_num_students_of_year_6_2(current_year integer): Η συγκεκριμένη συνάρτηση επιστρέφει τον αριθμό των φοιτητών που ικανοποιούν τις προϋποθέσεις αποφοίτησης και δεν έχουν ακόμη εκπονήσει διπλωματική εργασία του συγκεκριμένου έτους που δίνεται σαν όρισμα και τους επιστρέφει.

get_final_table_6_2(): Αυτή η συνάρτηση με την χρήση της προαναφερόμενης κατασκευάζει τον πίνακα τον οποίο θα εμφανίζει το view, καλώντας τα έτη για τα οποία θέλουμε να βρούμε τους φοιτητές, με τον εξής τρόπο. | Βάλε στον πίνακα την χρονιά X | Βρες πόσοι μαθητές είναι για την χρονιά X |

view_6_2: Η όψη αυτή απλά καλεί την συνάρτηση που αναφέρθηκε πριν και εμφανίζει τον πίνακα που θα γυρνάει ως αποτέλεσμα.

```sql
CREATE OR REPLACE FUNCTION public.get_num_students_of_year_6_2(
    current_year integer)
    RETURNS integer
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
declare
_row record;
counter integer := 0;
num_of_obligatory bigint :=0; -- ==35
num_of_non_obligatory bigint :=0; -- >=14
num_of_passed_courses bigint :=0; -- >=49
sum_of_units bigint :=0; -- >= 180
check_for_diploma bigint :=-1;
BEGIN
    drop table if exists students_amka cascade; --student amka that studied more than 9 semesters
    create temp table students_amka(amka character varying);
    insert into students_amka
    select distinct st.amka from "Semester" s, "Student" st
    where ((current_year - (select substring(st.amka, 1, 4)::integer) > 5) or
    ((current_year - (select substring(st.amka, 1, 4)::integer) = 5) and s.academic_season = 'spring'));

    for _row in select amka from "students_amka"
    loop
        num_of_obligatory := (select count(course_code) from "Register" r natural join "Course" c
                              where amka = _row.amka and register_status = 'pass' and obligatory);

        num_of_non_obligatory := (select count(course_code) from "Register" r natural join "Course" c
                                  where amka = _row.amka and register_status = 'pass' and not obligatory);

        num_of_passed_courses := num_of_obligatory + num_of_non_obligatory;

        sum_of_units := (select sum(units) from "Register" r natural join "Course" c
                         where amka = _row.amka and register_status = 'pass');

        check_for_diploma := (select count(student_amka) from "Diploma" d where student_amka = _row.amka);

        IF ((num_of_passed_courses >= (select min_courses from "SchoolRules" where year=current_year))
            and (sum_of_units >= (select min_units from "SchoolRules" where year=current_year))
            and (check_for_diploma = 0))
            THEN
            counter := counter + 1;
        END IF;
    end loop;
    RETURN counter;
END;
```

```sql
$BODY$;

ALTER FUNCTION public.get_num_students_of_year_6_2(integer)
    OWNER TO postgres;
```

```
CREATE OR REPLACE FUNCTION public.get_final_table_6_2(
    )
    RETURNS TABLE(year integer, num_of_students integer)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
declare
iterator integer;
BEGIN
    drop table if exists final_table cascade; --
    create temp table final_table(yearr integer, count integer);

    for iterator in 2012..2022
    loop
        insert into final_table values(iterator, get_num_students_of_year_6_2(iterator));
    end loop;
    RETURN QUERY
    select * from final_table;
END;
$BODY$;

ALTER FUNCTION public.get_final_table_6_2()
    OWNER TO postgres;


CREATE OR REPLACE VIEW public.view_6_2
 AS
 SELECT get_final_table_6_2.year,
    get_final_table_6_2.num_of_students
   FROM get_final_table_6_2() get_final_table_6_2(year, num_of_students);

ALTER TABLE public.view_6_2
    OWNER TO postgres;
```

Σημείωση: Για την 2η φάση της εργασία στο 6.2, η συνάρτηση είναι πολύ αργή και ίσως χρειαστεί να τρέξει για πολλά λεπτά. Επίσης, το υπόλοιπο κομμάτι της εργασίας δεν υλοποιήθηκε λόγω έλλειψης χρόνου.