Phase A report of the Database project

George Valavanis
George Andreadis

A find_labmodule_grade function has been implemented in the phaseB database to be able to run 1.2.

The **diploma_same_lab** function fully implements 3.8.

```
CREATE OR REPLACE FUNCTION diploma_same_lab()
RETURNS TABLE (title character varying) AS
$$
DECLARE X character varying; Y integer; proffs bigint; flag_num bigint;
BEGIN

   FOR X IN (SELECT DISTINCT diploma_num FROM "Committee")

   LOOP
      proffs:=(SELECT count(prof_amka) FROM "Committee" WHERE
diploma_num = X) ;
      FOR Y IN 1..10

      LOOP
         flag_num:=(SELECT count(prof_amka) FROM "Committee" JOIN
"Professor" ON(prof_amka=amka) WHERE diploma_num = X AND labjoins
= Y);
         IF(flag_num = proffs) THEN
            RETURN QUERY
            SELECT thesis_title FROM "Diploma" WHERE diploma_num = X;
         END IF;
      END LOOP;
   END LOOP;
END;
$$
LANGUAGE 'plpgsql';
```

The **all_courses** function fully implements 3.9.

```
create or replace function all_courses( c_code character(7))
returns table(course_code character(7), course_title character(100)) as
$$ be
gin
```

```
    return query

    with recursive courses(mc, dep)
        as( select cd.main, cd.dependent
        from "Course_depends" cd
        where cd.dependent = c_code
        union
        select cd.main, courses.dep
        from courses, "Course_depends" cd
        where courses.mc = cd.dependent
    )

    select mc, c.course_title from courses join "Course" c on (courses.mc =
c.course_code);

end;
$$
language 'plpgsql'
```

The **view_committee** view fully implemets 6.1.

```
CREATE VIEW view_committee AS
WITH professors as
    (SELECT amka_std,concat(surname, ' ',name)as fullname
    FROM "Person" NATURAL JOIN "Committee"
    WHERE prof_amka=amka
    ORDER BY NOT supervisor)

    SELECT amka_std as "AMKA" , string_agg(fullname, ',') as "Committee"
    FROM professors
    Group by amka_std
```

Function **insert_school_rules** was used alongside with **random_between** in order to fill School Rules with data.

```
CREATE OR REPLACE FUNCTION random_between(low INT ,high INT)
  RETURNS INT AS
$$ BE
GIN
  RETURN floor(random()* (high-low + 1) + low);
END;
$$ language 'plpgsql' STRICT;

CREATE OR REPLACE FUNCTION insert_school_rules()
RETURNS VOID AS
$$

DECLARE X integer;

BEGIN
```

```
FOR X IN (SELECT DISTINCT academic_year FROM "Semester")

LOOP
      INSERT INTO "School Rules" VALUES
(random_between(2,3),X,random_between(500,800),random_between(40,70));

   END LOOP;
END;
$$
language 'plpgsql';
```

The **students_per_year** view fully implemets 6.2.

```
CREATE VIEW students_per_year AS
WITH passed_courses as
      (SELECT amka, count(course_code) as courses
      FROM "Register" R
      WHERE register_status='pass'
      GROUP BY amka),
courses_1 as
      (SELECT amka, sum(units) as units1
      FROM "Register" NATURAL JOIN "Course"
      WHERE units BETWEEN 1 AND 2
   GROUP BY amka),
courses1_5 as
      (SELECT amka, sum(units*1.5) as units1_5
      FROM "Register" NATURAL JOIN "Course"
      WHERE units BETWEEN 3 AND 4
   GROUP BY amka),
courses_2 as
      (SELECT amka, sum(units*2) as units2
      FROM "Register" NATURAL JOIN "Course"
      WHERE units = 5
   GROUP BY amka),
amka_courses_units as
      (SELECT amka, courses, sum(COALESCE(units1,0) +
COALESCE(units1_5,0) + COALESCE(units2,0)) as units,
date_part('year',entry_date) as entry_year
      FROM passed_courses LEFT JOIN courses_1 USING(amka)
                  LEFT JOIN courses1_5 USING(amka)
                  LEFT JOIN courses_2 USING(amka)
                  NATURAL JOIN "Student"
      GROUP BY amka,courses,entry_date),
```

```
final_table as(
	SELECT entry_year as year, count(amka) as undergraduates
	FROM amka_courses_units
	WHERE courses >= (SELECT min_courses FROM "School Rules"
WHERE year=entry_year)
	AND units >= (SELECT min_units FROM "School Rules" WHERE
year=entry_year)
   GROUP BY entry_year),
years as(
	SELECT academic_year
	FROM "Semester"
	WHERE academic_year NOT IN (SELECT year FROM final_table))

SELECT *
FROM final_table
UNION
SELECT academic_year as year, '0'
FROM years
ORDER BY Year
```