TP2 — Academic records

Advanced Databases (RSD/AI)

The objective is to progressively build a database named `academic_records`, intended to manage the academic results of the students of the computer science department. The database makes it possible to record students, subjects and their evaluations, and to automatically compute the averages as well as the final result.

1 Conceptual schema

student(id, full_name, group_no, overall_average, result)

- `id`: unique identifier of the student.
- `full_name`: last name and first name(s), in uppercase.
- `group_no`: tutorial group number (1 to 5).
- `overall_average`: weighted overall average, computed automatically.
- `result`: final result (PASSED or FAILED).

subject(id, name, coefficient)

- `id`: unique identifier of the subject.
- `name`: title of the subject.
- `coefficient`: weight of the subject in the overall average (1 to 5).

evaluation(student_id, subject_id, td, tp, exam, subject_average)

- `student_id`: identifier of the student.
- `subject_id`: identifier of the subject.
- `td, tp, exam`: grades out of 20 (tutorial, lab, exam).
- `subject_average`: average for a subject, computed automatically.

2 Constraints

- Each student must be uniquely identified.
- Each subject must be uniquely identified.
- A student's group must be between 1 and 5.
- A subject's coefficient must be between 1 and 5.
- The TD, TP and Exam grades must be between 0 and 20.
- A student can be enrolled in several subjects.
- A student can have only one evaluation per subject.
- Each evaluation must include three grades: TD, TP and Exam.
- The average of a module is computed automatically as follows:

$$\text{subject\_average} = \frac{\frac{td+tp}{2} + exam}{2}$$

- A student's overall average is a weighted average of the subject averages:

$$\text{overall\_average} = \frac{\sum_i (\text{subject\_average}_i \times \text{coefficient}_i)}{\sum_i \text{coefficient}_i}$$

- Automatic decision: if `overall_average` ≥ 10 then `PASSED` else `FAILED`.

Note

In this handout, a subject's coefficient is bounded to **1–5** (and not 1–20, which is the grading scale itself) — a subject does not weigh more than 5 times another one in the overall average.

3 Work to do

Build the database progressively according to the following steps:

1. **Create a role named `manager`.** This role will be used to administer the database and to carry out all the design operations.
2. **Create the `academic_records` database.** The database must be created with the `manager` role as its owner, and configured with the appropriate locale settings.
3. **Create the necessary domains.** Define the domains corresponding to the business rules (grades, coefficients, groups…) in order to centralize the constraints used in the tables.
4. **Create the tables.** Create the tables of the schema and define the primary keys, foreign keys and business constraints. The tables to create are: `student`, `subject`, `evaluation`.
5. **Create a trigger function and its associated trigger.** The function must automate the enforcement of the business rules: computing the averages, updating the final result and keeping the data consistent.
6. **Populate the database with a sample data set.** Insert a consistent set of students, subjects and evaluations to allow testing.
7. **Run validation tests.** Check that the trigger works, test insertions, updates, data consistency and the results obtained.