Computer Science Mid-Term Exam
2nd year - November 2023

Total duration: *1h30*
Authorised documents: *None.*

— The max grade is 20.

— The exam is 4 pages long.

Exercise 1 : SQL Queries (11 pts)

The FIMI department organises a cross race, in which 1st and 2nd year students can take part. Students register in teams. A team may be made up of students from different years (1st and 2nd year). An itinerary is composed of distinct stages. The same stage may be part of several itineraries. The stages composing an itinerary each have an order number in the itinerary. Each team chooses to take part in a single itinerary. The race has a fixed duration. On each stage of the itinerary, one team member is running. The members change across the stages. The winning team is the one that covers the greatest distance in the given time. In this way, a team can cover the chosen itinerary several times during the time allocated. The last time a team starts the chosen itinerary may lead to only a subset of its stages being validated. Each time a participant fully completes a stage, the start date and the time spent completing the stage are stored in the database.

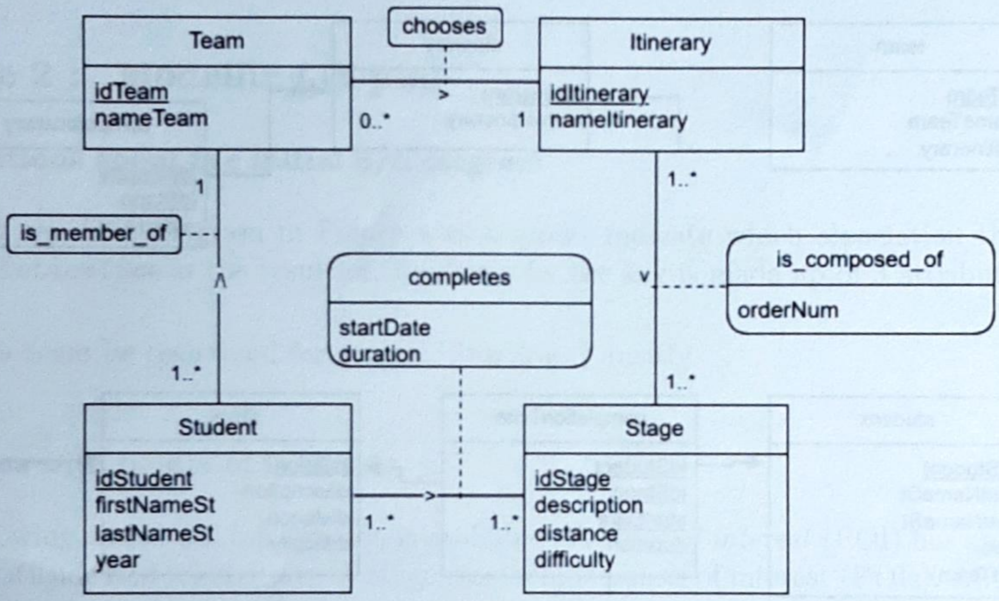

FIGURE 1 – Database E/A diagram

The FIMI department has created a database whose relational schema is described below. The graphical version is shown in Figure 2 and the corresponding Entity/Association model in Figure 1. The attributes forming the key of a relationship are underlined. For simplicity, the attribute types are not indicated in the relationships. The attributes `distance`, `difficulty`,

orderNum, **year** and **duration** are numbers. The **startDate** attribute is of type date. All the other attributes are character strings.

itinerary(idItinerary, nameItinerary)
Each itinerary is uniquely identified by its identifier. Its name is also memorised.
stage(idStage, description, distance, difficulty)
Each stage is uniquely identified by an identifier. Its description, the distance to be covered and its level of difficulty from 1 to 4 are stored.
compoItinerary(idItinerary,idStage, orderNum)
A tuple is created in this table for each stage that is part of an itinerary. The attributes **idItinerary** and **idStage** are foreign keys to the **itinerary** and **stage** relations respectively. As an itinerary is a series of stages, the order number of the stage in the itinerary is also stored.
student(idStudent, firstNameSt, lastNameSt, year, idTeam)
Each student has a unique identifier. All FIMI students are represented in this table. The **idTeam** field is a foreign key referring to the **team** table. It is only filled in for students taking part in the cross run as part of a team (otherwise it is undefined).
team(idTeam, nameTeam, idItinerary)
Each tuple in this table, uniquely identified (**idTeam**), represents one of the participating teams. The name of the team and the itinerary in which it is registered are stored. The **idItinerary** attribute is a foreign key referencing the **itinerary** relation.
completionTime(idStudent, idStage, startDate, duration) A row in this relation stores that a student has completed a stage in **duration** minutes, starting at the **startDate** date. Only cases where the stage has been completed are recorded. The **idStudent** and **idStage** attributes are foreign keys referencing the **student** and **stage** relations respectively. Une ligne de cette relation mémorise qu'un élève a parcouru une étape en **duree** minutes, en commençant à la date **dateDeb**.

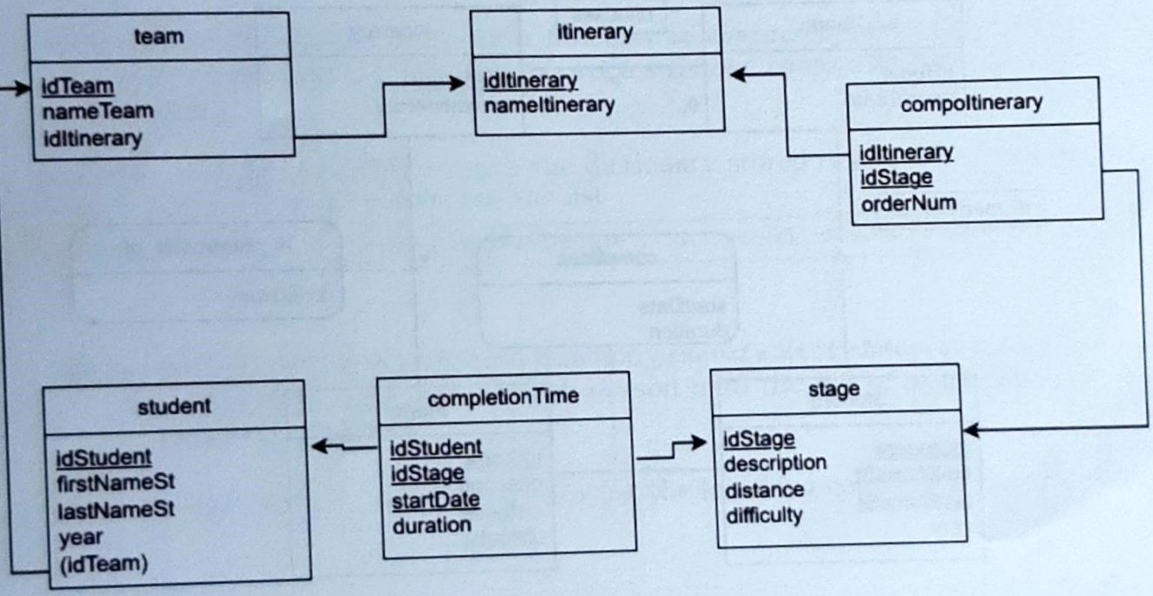

FIGURE 2 – Graphical representation of the database relational schema

Question : Write the SQL queries corresponding to the following searches. Take care to eliminate duplicates where the result should not contain any. For each query, use as few tables as possible.

1. INSA would like to calculate the number of small bottles of water to be purchased for the participants. It plans to buy 10 bottles per participant. Express the query to calculate the total number of bottles to be ordered.

2. Give all the itinerary identifiers with at least 3 stages of difficulty greater than 2.
3. Rank the teams in descending order of distance covered. Display the team, the total distance covered and the total time. If the distances are equal, the results are ranked in ascending order according to the total time taken.

4. Give the last name, first name and year of study of the participants in the "*no cardio united*" team who have never run the stage with the identifier *ETP5.*

5. Give the last names of the students, the year of study and the team number, of the students who completed the stage with the identifier '*ETP5*' within less than the average time taken on this step.

6. List in alphabetical order the composition of the team in which the student *Antoine Dupont* participates. It is assumed that there are no two people called Antoine Dupont in the database.

7. The "*Les lutins agiles*" team has registered to an itinerary. Display the stages of the itinerary they have to complete, with an identifier, description, distance and level for each stage. In addition, the stages must be displayed in order of their order number (first, the stage with the number 1 ...etc).

Exercise 2 : Modelling (5 pts)

2.1 Questions about the initial E/A diagram

1. Using the E/A diagram in Figure 1 as a guide, indicate which association the relation `completionTime` is the result of. Explain why the key is made up of 3 attributes.

2. Can a team be registered for several itineraries ? Justify.

2.2 Stages with points of interest

The following additional information is available : A *point of interest* (POI) has an identifier, a name, a latitude and a longitude. A stage can include points of interest (POIs). All the points of interest listed belong to at least one stage. Each point of interest offers one challenge and one challenge only. A challenge has an identifier and a description, for example, *do 20 crunches.* The same challenge can be proposed by several points of interest. When a student completes a challenge, the time at which he or she does so and the time it takes are memorised.

1. Propose an extension to the E/A diagram in Figure 1 that takes into account this new information.

2. Indicate the changes to be made to the relational model following the transformation of your E/A schema

Exercise 3 : Dictionaries (4 pts)

In order to answer readers' questions about the articles published, a national newspaper stores, for each article written, the name and telephone number of a person whom readers can contact if they have any questions.

A programme is created to automate processing. It uses a dictionary that associates each item identifier with a list containing the name of the contact person and their telephone number.

Given the variable `articles`, defined in the code below, answer the various questions.

```
articles = {
'GraphèmeCerveauHumain231023' :  ['Jean Dupont' , '0666666666'],
'PerfUltraEndu241023' : ['Alain Durand' , '0666666667'],
'MortSubiteNourrisson241023' : ['Jean Dupont' , '0666666666'],
'ChantAmourMoucheVinaigre' : ['Alain Durand' , '0666666667'],
'ChatGPTExpMultiMedia211023' : [ 'Lina Romero', '0677777777'],
'GrippeAviaireFouBassan201023': ['Alain Durand' , '0666666667']
}
```

(3.1) Write a Python code to display the name and telephone number of the contact person for the article identified with `'PerfUltraEndu241023'`.

The display will take the form :

```
Name: Alain Durand, phone: 0666666667
```

(3.2) Write the Python code for a function `is_contact_of(articles, name_p)`, which returns the list of identifiers of articles whose contact is the person with the name `name_p`

Example : calling the function with the dictionary stored in the *articles* variable above and the name *'Alain Durand'* will return the list :

```
['PerfUltraEndu241023', 'ChantAmourMoucheVinaigre', 'GrippeAviaireFouBassan201023']
```

(3.3) Write the Python code for a function `generate_dict_names(articles)`, which returns a dictionary associating each person with the list of article identifiers for which they are a contact.

Note : a dictionary `d` can be created using the following code : `d = dict()`.