**DI, FCT-NOVA** **April 10, 2021**

# Sistemas de Bases de Dados

## 1st Test, 2020/21

**Duration: 1 Hour and 30 Minutes**

## Group 1

Consider a (simplified) database for managing occurrences of a proctoring systems for tests done by students, with the following tables (where attributes that form the primary keys are underlined):

*courses(idC,nameC,idP)*
*programmes(idP,nameP,coordinator)*
*testResults(idS,IdC,dateT,results)*
*students(idS,nameS,idP,sex,age,photo)*
*tests(IdC,dateT,NameT)*
*occurrences(idS,idC,dateT,Moment,Type,Probability)*

Each of these tables has a B+ tree clustered index over the primary key.

For each student, the database stores her id, name, sex, age, the id of the programme in which the student is enrolled, and her photo. There is a table for storing the various programmes, where each programme has an id, a name, and the name of the programme's coordinator, and a table for storing course (i.e. curricular units), where each course has an id, a name and the id of the programme in which the course is taught (note that, this way, a course may only be taught in one programme).

Each course may have several tests, and this is stored in the *tests* table. Each test, in this simplified database, only has the id of the course to which the test belongs, the date in which it occurs, and the name of the test (e.g. '1st test', 'midterm teste', etc). The table *testResults* stores information about which students made which test (with the id of the student, and the identification of the test – i.e. the id of the course and the date of the test) and the result/grade of each student in the test.

Finally, the *occurrences* table stores the various events that the proctoring system flagged while each student was doing each test. For example, an occurrence (1,'A',10.04.2021,600,'Stranger in room','High') means that in the test of course 'A' made by student 1 on April 10, 2021, 600 seconds after the test started the system detected a stranger in the room with High probability.

Tuples of all these tables are of variable size. Furthermore, at a given moment the *courses* table has 500 tuples and each tuple occupies 25 bytes, the *programmes* table has 50 tuples each with 50 bytes on average, the *tests* table has 5.000 tuples each with 20 bytes, the *testResults* 1,000,000 tuples, each with 10 bytes, and *occurrences* table has 10,000,000 tuples each with 30 bytes (i.e. the *occurrences* table is about 300MB). Finally, the *students* table has 50,000 tuples and each tuple occupies 3MB on average but if one excludes the *photo* attribute, the remaining attributes only occupy 25 bytes.

**Note:** In this group, whenever an example is asked for, the example **must** be in terms of the database above. Moreover, **all** your answers must include a brief **justification**.

**1 a)** How would a DBMS, like the ones you studied in the course, store the *students* table?

**1 b)** By using this database over time, most of the tuples in the *occurrences* table become almost irrelevant. In fact, it is highly unusual to query occurrences from tests that were done long ago (e.g. from previous years).
Most DBMS have a mechanism of organisation of data that may significantly improve the efficiency in situations like this. What is that mechanism, and how could it be used to improve the efficiency of queries that only care about tests done e.g. in the current academic year?

**1 c)** The *tests* table has a primary key composed by the concatenation of 2 attributes: *idC* and *dateT*. As such, the clustering index can be created taking into account the concatenation *(idC,dateT)* or the concatenation *(dateT,idC)*. Which one would you prefer for this particular database (given its expected usage), and why?

**1 d)** Assume that, besides the clustered index, there are 3 non-clustered B+-tree indices defined on the *students* table: one over the attribute *name*, another over the attribute *idP* and yet another over the concatenation of those two attributes. Which of these 3 indices would make the execution of the following query more efficient?

**select** *idP* **from** *students* **where** *nome* = *'José Pinto de Sousa'*

**1 e)** Give an example of an SQL query whose execution would be much more efficient if there were bitmap indices, and which indices would you have to create for that?

## Group 2

**2 a)** *Database Management Systems, usually don't rely on many of the host Operating System's functionalities. In particular, the best DBMS don't rely on the buffer management functionality of the Operating System.*
Explain why.

**2 b)** Briefly explain the advantages and disadvantages of log-structured merge (LSM) over normal B+-trees.
In your explanation, give an example of a situation (e.g. a table and its expected usage) where an LSM tree would be quite appropriate and an example where it would be inappropriate.

**2 c)** List at least one advantage and one disadvantage of each of the following strategies for storing a relational database:

- Store each relation in one file.
- Store multiple relations (perhaps even the entire database) in one file