DI, FCT-NOVA **June 11, 2021**

Sistemas de Bases de Dados

3rd Test, 2020/21

Duration: 1 Hour and 30 Minutes

Group 1

Consider (again!) a 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, where the order by which the attributes are concatenated is the one shown above. Moreover, assume that all the relevant foreign key are defined: viz. for *idP* both in *courses* and *students*, referencing *programmes*; for *idC* in *tests*, *testResults* and *occurrences*, referencing *courses*; for *idS* in *testResults* and in *occurrences*, referencing *students*; for *(idC,dateT)* in both *testResults* and *occurrences*, referencing *tests*.

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.

The database is implemented in a system using 4KB blocks and a memory of just 100 block.

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

1 a) Consider the following schedule with 3 concurrent transactions

T1	T2	T3
	select * **from** students	
update students **set** age = 23 **where** idS =111		
	update students **set** age = 22 **where** idS =111	
		update students **set** age = 24 **where** idS =111

1. Is this schedule conflict serializable? If so, what are the conflict-equivalent serial schedules; if not, why not?

2. Is this schedule serializable? If so, what are the equivalent serialisations of the transactions; if not, why not?

1 b) Consider the following concurrent transactions (where, for convenience, all the commands are labelled):

1. **begin transaction**	***A.*** **begin transaction**
2. **select * from** students**;**	***B.*** **select * from** students**;**
3. **insert into** students **values (**1**, ...);**	***C.*** **insert into** students **values (**2**, ...);**
4. **select * from** students**;**	***D.*** **select * from** students**;**
5. **commit;**	***E.*** **commit;**

Present **one** schedule of the operations in the transactions for which the result differs whether the transactions are both executed in Read uncommitted, Read committed or Serializable isolation mode, stating what the result would be for which of the 3 modes (to simplify the presentation of results, assume that initially the table *students* is empty).
[You can write a schedule as a sequence of numbers and letters identifying the order in which the operations above are executed, e.g. 1, 2, A, B, 3, 4, 5, C, D, E]

1 c) Assume that the database management system uses a rigorous 2-phase lock protocol for concurrency control. Give an example of a schedule (**in SQL**) with 3 concurrent transactions that would cause a deadlock.

1 d) Given an example of a schedule (**in SQL**) with 2 concurrent transactions such that, if the database management system uses a timestamp-based protocol, one of the transactions would have to abort, whereas, if a 2-phase lock protocol was used instead, both transactions would end with success.

1 e) The Consistency property in ACID transactions only requires that the consistency of integrity constraints is checked at the end of each transaction, and not after each individual action that constitutes the transaction.
Possibly assuming more foreign key constraints than the ones described above, give an example of a transaction that could not be executed if consistency was checked after each individual action, but would succeed without any problem with the usual consistency of ACID transactions. (In case you assume more foreign keys, clearly state what are these foreign keys).

Group 2

2 a) Show **two alternative ways** to deal with deadlocks in database systems that use lock-based protocols for concurrency control. Explain the advantages and disadvantages of each of those alternative ways.

2 b) *Database Management Systems supporting ACID transactions must include a Recovery System, that is usually log-based and either relies on Deferred Database Modifications or on Immediate Database Modifications.*

Which of the ACID properties (Atomicity and/or Consistency and/or Isolation and/or Durability) do Recovery Systems address? How are those properties addressed differently in systems that rely on deferred versus on immediate database modification?

2 c) *When processing queries over databases that are distributed by fragmentation, the cost of transferring data from one site to another becomes quite significant. This justifies the existence of new strategies for executing queries, that try to minimise the volume of transferred data. One of these strategies is the so-called semijoin strategy.*

Explain what is the semijoin strategy, and give an example where its usage would improve the efficiency of the execution of a query.