DI, FCT-NOVA **May 22, 2021**

Sistemas de Bases de Dados

2nd 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, 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. Moreover, **all** your answers must include a brief **justification**.

1 a) Show two execution plans for the following query (what are the names of the students from the Computer Science programme that had at least one high probability occurrence of 'Stranger in room' in a test), and justify which one should have the lowest cost:

select distinct *nameS*
from *occurences* **natural inner join** *students* **natural inner join** *programmes*
where *nameP* = 'Computer Science' **and** *Type* = 'Stranger in room' **and** *Probability* = 'High'

Note: Since drawing a tree can be quite challenging in the browser, you may opt either by writing some (pseudo) relational algebra expressions annotated with the algorithms, or by explaining the way the query might be executed in English (e.g. starts by executing this algorithm over this table with that condition, then takes the result as input for this other algorithm… etc).

1 b) Assume that the DBMS only has the "block nested loop join" algorithm for joins, eventually using index files when available. For each of the join queries below, which is the best join order?

1. *courses* ⋈ *occurences*
2. *occurrences* ⋈ *tests* ⋈ *students*

1 c) Give an example of a join of two or more of the above tables in which the merge-join algorithm is clearly the most adequate.

1 d) Give an example of a query involving two or more tables that would benefit from the usage of histograms by the query optimiser.

1 e) Considering now that the system implements all the algorithms that we've studied during the course, what is the best execution plan for the following query:

select *A.name*
from (select * from *students* **where** *sex* = 'F'**)** *A*
where *A.idade* < 30 **and**
exists (select * from *programmes B* **where** *A.idP* = *B.idP* **and** *A.idade* < 25**)**

Group 2

2 a) *Integrity constraints defined over the database (such as primary keys, foreign key, domain restrictions, assertions, etc) are meant to guarantee the consistency of the data. But, the mere definition of these constraints, even without any data structures for making their implementation more efficient, can help significantly in query optimisation.*

List **two different** ways by which the query optimiser may benefit from the definition of foreign keys, **explaining** why that is so and giving, for each of them, **an example** of a situation (e.g. database and query) in which the benefit is clear.

2 b) Which of the algorithms, block-nested loop join or hash join, is more amenable for adjusting to intra-operation parallelism? Why?

2 c) *We've studied two different ways of combining the execution of operations/algorithms in an execution plan: pipelining and materialisation. Materialisation can always be used, but some algorithms cannot be combined using pipelining.*

Given an example of algorithms that cannot be combined by pipelining and explain why that is so.