DI, FCT-NOVA

Sistemas de Bases de Dados

1st Test – 2021/22

Duration: 1 Hour and 30 Minutes

Group 1

Consider a (simplified) database for managing a flying-passenger locator system of a national health authority (something that is quite common since the COVID pandemic), with the following tables (where attributes that form the primary keys are underlined):

routes(NumR,IdC,Origin,Time,Destination) *persons(IdP,NameP,Address,Zip,Sex,Age,CertID)*
flights(NumR,Date) *companies(IdC,NameC,Country,…)*
travels(Code,IdP,Origin,Destination) *travelLegs(Code,NumR,Date,SeatNo)*
vaccinationCert(CertID,IdP,DateV,Doses,Lab,issuedBy)

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

For each passenger, the database stores, in the *persons* table, her id, name, address (including zip code) sex, age, the id of her certificate of vaccination (with a **null** value if the person has no vaccination certificate). Each vaccination certificate has, besides a unique certificate identifier, the identifier of the person, the date of the (last) vaccination shot, number of doses taken, the name of the Lab that made the vaccine, and of the authority that issued the certificate.

The health authority uses this database to track which flights each person is taking, passing through which airports. Each flight has a route number and a date. Route numbers refer to routes that may take place everyday, or once a week, etc. E.g., Route number TP943 refers to daily TAP flights leaving at 1pm from Genève to Lisbon.

A travel is a sequence of flights that a passenger may take. Each one is identified by a reservation code, refers to a passenger, and has and origin and a final destination. Each step (or travel leg) in a travel, refers to a flight (identified by the route number and date), and the seat number reserved for the corresponding passenger in that flight.

Furthermore, at a given moment the *companies* table has 100 tuples, the *routes* table has 2.000 tuples, the *flights* table has 200.000 tuples, the *travels* table 400.000 tuple, *vaccinationCert* with 800.000 tuples, *persons* with 1.000.000 tuples, and *travelLegs* with 20.000.000 tuples. Tuples of all these tables are of variable size and, on average, each tuple of *flights* has 25 bytes, each tuple of either *routes*, *companies*, *travels* or *travelLegs* has 50 bytes, each tuple of *vaccinationCert* has 100 bytes, and each tuple of *persons* has 200 bytes.

The database is implemented in a system using 4KB blocks and a memory of 1,000 blocks.

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) Give an example of an SQL query and corresponding execution for which the LRU buffer replacement strategy (typical of operating system) is not the most adequate, and that an MRU strategy would be more adequate.

1 b) Table *travelLegs* has two candidate keys: the one that is given above as primary key, i.e. *(Code,NumR,Date)*, and *(NumR,Date,SeatNo)* – since there can be no two persons with the same seat number in the same flight.
Do you think that the choice of *(Code,NumR,Date)* as the primary key for the clustering index was wise, or would it be better to have the other key as the one used for the clustering index? Why?

1 c) Which secondary index(es) would you create to improve the efficiency of a query such as the one below (which returns how many seats were occupied in the flight TP943 yesterday):

select count(*SeatNo***) from** *travelLegs* **where** *NumR = 'TP943'* **and** *Date = '07/04/2022'*

1 d) 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?

1 e) Give an example of an SQL query involving more than one table whose execution would be much less efficient if the order by which the attributes are concatenated in the clustered indexes was not the one given above.

Group 2

2 a) *In general DBMSs use slotted-pages for organising the tuples of a table in blocks.*
Explain (1) the main advantages of this organisation of the tuples, (2) which limitations it imposes, and (3) how database management systems circumvent these limitations.

2 b) *Indices speed query processing, but it is usually a bad idea to create indices on every attribute, and every combination of attributes that are potential search keys.*
Explain why.

2 c) What would the occupancy of each leaf node of a B+-tree be if index entries are always inserted in sorted ascending order? Explain why.