DI- FCT/UNL **May 8th, 2025**

Database Systems

Test-1 2024/25

Duration: 2 hours (limited information)

Group 1 (each question is worth 2,5 values out of 20)

Consider part of a database for a mobile phone operator that provides roaming services

Persons({personID,name,fiscalNR,countryID,...}) Countries({countryID,countryName,code})

Devices({deviceID,number,operatorID,personID}) Operators(operatorID,operatorName,countryID)

Registration(regID,deviceID,operatorID,start,end) Calls(callID,regCaller,regDest,time,duration,cost)

Persons (personID as primary key) stores user data and includes a foreign key countryID referencing **Countries** (countryID as primary key), which holds country names and dial codes. **Operators** (operatorID as primary key) also reference **Countries** to indicate their home country. **Devices** (deviceID as primary key) are foreign keys to **Persons** (personID) and **Operators** (operatorID), indicating ownership and issuing operator of the device contract with the operator. **Registration** (regID as primary key) logs device roaming sessions using foreign keys deviceID and operatorID, where start and end are timestamps (end might be null if the roaming session is active). **Calls** (callID as primary key) track roaming calls, with foreign keys regCaller and regDest to caller and destination of a call, referencing **Registration**, and include details like time, duration (integer in seconds), and cost. For each of these tables there is a secondary B+ tree (non-clustering) index on the primary key attribute(s), created with the column order indicated in the tables. Additionally, there is a composite secondary index on Calls(regCaller,regDest).

The adopted DBMS uses blocks of 4KiB (4096 bytes). The records of all tables have a variable size, and at any given time, the Persons table has 100,000 tuples (tuple size 108 bytes, total 2640 disk blocks), and Devices 200,000 tuples (62 bytes/3028 blocks), the Countries table 195 tuples (58 bytes/3 disk blocks), the Operators table 740 tuples (58 bytes, total 11 disk blocks), the Registration table has 857,000 tuples (28 bytes/5852 blocks), and the Calls table 10,000,000 tuples (28 bytes per tuple/68,360 disk blocks).

A B+ tree node can contain about 100 search keys, and it is known that a seek time is 10ms and the transfer time of a block t_T is 1ms, while the memory only holds 100 blocks.

Note: In this group, whenever examples are requested, these must be exclusively about this database, Additionally, all the answers must contain a brief justification.

1 a) Present two execution plans for the following SQL query, briefly justifying which of the plans has the least cost in the given database.

SELECT personID, name, deviceID, number
FROM Persons NATURAL INNER JOIN Devices NATURAL INNER JOIN Countries
WHERE countryName = 'Portugal' AND name BETWEEN 'Carla' AND 'Carlos'

1 b) Present a query for which the use of hash static file organization would be helpful.

1 c) Indicate whether in your opinion the SQL query below can benefit from the existing indices. Would you introduce extra indices for optimizing the query?

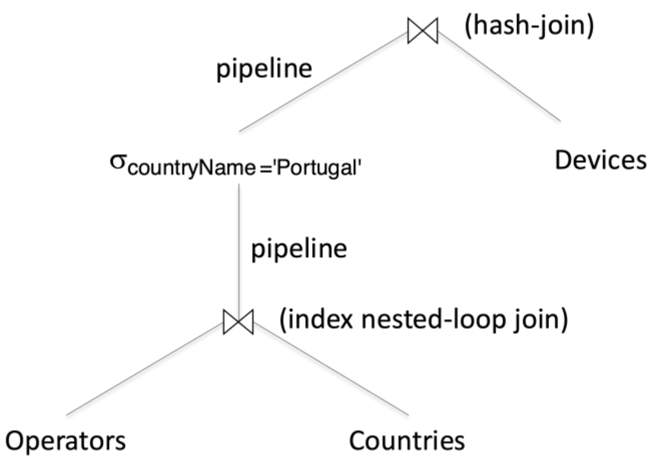
SELECT * FROM Persons NATURAL INNER JOIN Devices
NATURAL INNER JOIN Registrations INNER JOIN Calls ON (regID = regCaller)
WHERE personID = 74578

1 d) Explain if a bitmap index could improve the linear scan execution of the following query:

SELECT SUM(cost) FROM Calls
WHERE duration <= 1000

(empty page – any content will not be graded)

1 e) Consider the execution plan presented in the figure. Determine the least cost of this plan knowing that there are 4 operators in Portugal, and that in Portugal there are 10.000 registered devices. Assume that the height of B+-tree for the non-clustering index of Countries is 2. Devices is used as probe relation.



Group 2 (each question is worth 2,5 values out of 20)

Note: The response to each of the items in this group cannot, under any circumstances, exceed one page.

2 a) Explain why RAID LEVEL 6 is becoming more and more relevant, especially for disks with large capacities.

2 b) Explain why it is needed a NULL bitmap and a DELETED bitmap in bitmap indices.

2 c) If recursive partitioning is not required, the hash-join algorithm has a worst-case complexity of $3(b_r+b_s) + 4*n_h$ block transfers + $2(\lceil b_r / b_b \rceil + \lceil b_s / b_b \rceil) + 2* n_h$ seeks, when b_b disk blocks are used as blocking unit in the partitioning phase and n_h is the total number of partitions. Explain how the formula is derived (you may ignore the $4*n_h$ block transfers cost).

PSEUDO-CODE FOR HASH-JOIN

1. Partition the relation s using hashing function h. When partitioning a relation, one block of memory is reserved as the output buffer for each partition.
2. Partition r similarly.
3. For each i where $0 <= i < n_h$ (where n_h is the total number of partitions):
   (a) Load partition s_i into memory and build an in-memory hash index on it using the join attribute. This hash index uses a different hash function than the earlier one h.
   (b) Read the tuples in partition r_i from the disk one by one. For each tuple t_r locate each matching tuple t_s in s_i using the in-memory hash index. Output the concatenation of their attributes.

Relation s is called the **build** input and r is called the **probe** input.

THE END

(SCRAP PAPER)