Database Administration and Tuning
Mini-Project 1 - Report

Henrique Rocha - 68621
Ludijor Barros - 68626
Fábio Martins - 71073

April 20, 2015

1 SQL Server Databases

Present SQL Server T-SQL commands for accomplishing the following tasks:

a)

Create a database named NutrientsDB, containing one log file and three different data files, in three distinct filegroups (i.e., one data file in each filegroup). The log file should have an initial size of 25MB and a maximum size of 250MB. All data files should have an unlimited maximum size, except the one in the primary filegroup, which should have a maximum size of 1GB). The first data file on the first secondary filegroup should have an initial size of 100MB, and the remaining files should have an initial size of 50MB. All files should grow at a rate of 50%, except for the data file in the primary filegroup, which should grow by 5MB, every time this is required.

```sql
CREATE DATABASE NutrientsDB ON
PRIMARY (
       NAME = NutrientsDB,
       FILENAME = 'd:\data\NutrientsDB_01.mdf',
       SIZE = 50MB, MAXSIZE = 1GB, FILEGROWTH = 5MB ),
FILEGROUP NutrientsDB_02 (
       NAME = 'NutrientsDB_02',
       FILENAME = 'd:\data\NutrientsDB_02.ndf',
       SIZE = 100MB, MAXSIZE=UNLIMITED, FILEGROWTH = 50% ),
FILEGROUP NutrientsDB_03 (
       NAME = 'NutrientsDB_03',
       FILENAME = 'd:\data\NutrientsDB_03.ndf',
       SIZE = 50MB, MAXSIZE=UNLIMITED, FILEGROWTH = 50% )
LOG ON (
       NAME = 'NutrientsLog',
       FILENAME = 'd:\data\NutrientsLog.ldf',
       SIZE = 25MB, MAXSIZE = 250MB, FILEGROWTH = 50% );
```

b)

Create a table named Cheese in the NutrientsDB database. The table should have a numeric attribute named cheeseID, that identifies the individual records, an alphanumeric attribute named Type, and four other numeric attributes named Calories, Proteins, Carbohidrates, and Fat. The table should be partitioned so that all tuples where cheeseID is less or equal than 50 are physically stored in the primary filegroup, all tuples where the cheeseID is greater than 50, but less or equal than 100, are physically stored in the first secondary filegroup, and the remaining tuples are physically stored in the second secondary filegroup.

```
USE NutrientsDB;

CREATE PARTITION FUNCTION NutrientsPartitionFunction(NUMERIC)
AS RANGE LEFT FOR VALUES(50, 100);

CREATE PARTITION SCHEME NutrientsScheme
AS PARTITION NutrientsPartitionFunction
TO ([PRIMARY], NutrientsDB_02, NutrientsDB_03);

CREATE TABLE Cheese (
  cheeseID NUMERIC PRIMARY KEY,
  Type NVARCHAR(25),
  Calories NUMERIC,
  Proteins NUMERIC,
  Carbohidrates NUMERIC,
  Fat NUMERIC
) ON NutrientsScheme(cheeseID);
```

c)

In the table named Cheese, the amount of calories is stored in an attribute named Calories, in Kcals per 100 grams. Create an index over the table with a search key corresponding to the calories in cals per 100 grams, including the amount of protein and fat as additional attributes that are not part of the search key. The index should be physically stored in the primary filegroup. Indicate also if the index is clustered or non-clustered, justifying.

```
USE NutrientsDB;
CREATE NONCLUSTERED INDEX Nutrients_Calories_IX
ON Cheese(Calories)
INCLUDE (Proteins, Fat)
ON [PRIMARY];
```

O índice deve ser non-clustered, pois os valores presentes na coluna Calories não têm critério de ordenação.

2 B+Tree Index Structures

Consider the problem of inserting the following keys, in the given order, into an empty B+-tree where nodes can hold up to 3 values:

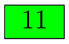
Parmesão, Ilha, Camembert, Fresco, Requeijão, Azeitão, Alverca, Serra, Alcobaça, Roquefort, Flamengo, Emmental, Évora, Creme, Serpa, Quark

1 - Alcobaça
2 - Alverca
3 - Azeitão
4 - Camembert
5 - Creme
6 - Emmental
7 - Évora
8 - Flamengo
9 - Fresco
10 - Ilha
11 - Parmesão
12 - Quark
13 - Requeijão
14 - Roquefort
15 - Serpa
16 - Serra

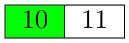
a)

Draw the tree after each insertion. Insertion order: 11, 10, 4, 9, 13, 3, 2, 16, 1, 14, 8, 6, 7, 5, 15, 12

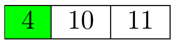
Insert 11 (Parmesão)



Insert 10 (Ilha)

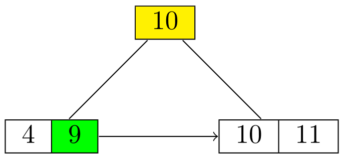


Insert 4 (Camembert)

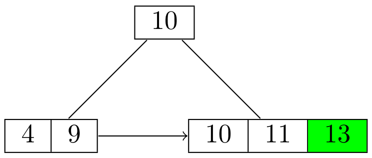


Insert 9 (Fresco)

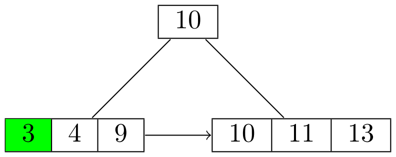
9 is inserted on leaf node [4,10,11], which will split into [4,9] and [10,11], 10 goes up.



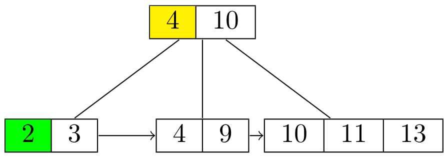
Insert 13 (Requeijão)



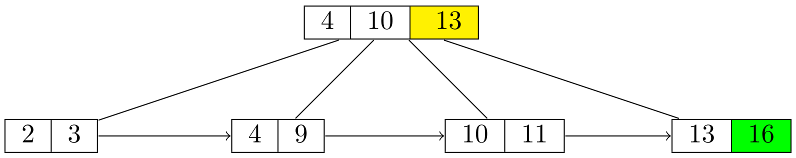
Insert 3 (Azeitão)



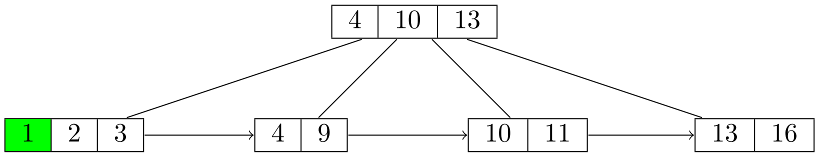
Insert 2 (Alverca)
2 is inserted on leaf node [3,4,9], which will split into [2,3] and [4,9], 4 goes up.



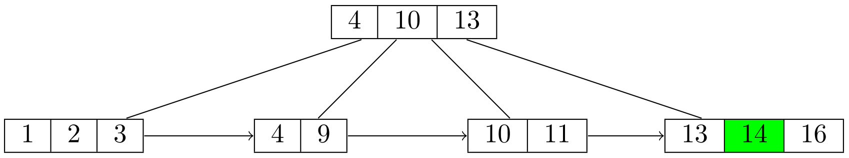
Insert 16 (Serra)
16 is inserted on leaf node [10,11,13], which will split into [10,11] and [13,16], 13 goes up.



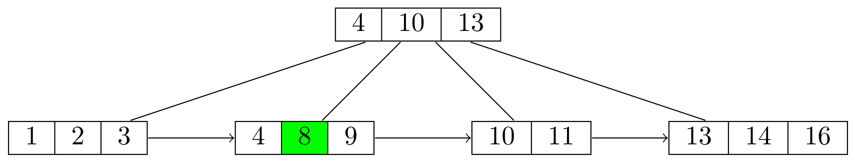
Insert 1 (Alcobaça)



Insert 14 (Roquefort)



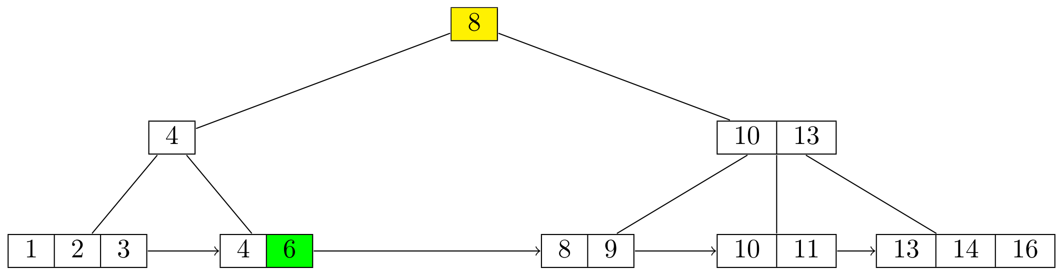
Insert 8 (Flamengo)



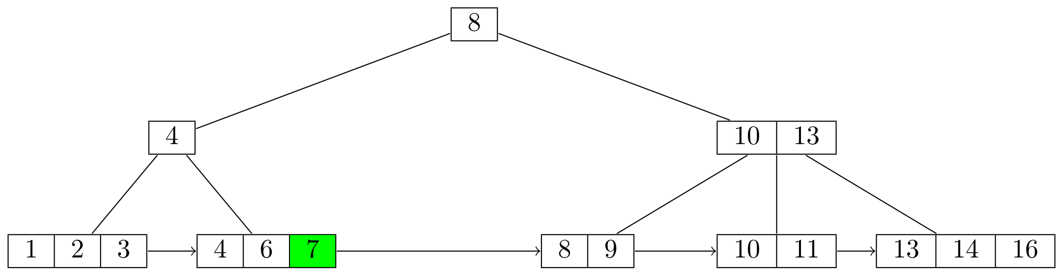
Insert 6 (Emmental)

6 is inserted on leaf node [4,8,9], which will split into [4,6] and [8,9], 8 goes up.

8 is inserted on non leaf node [4|10|13], which will split into [4] and [10|13], 8 goes up. [4] points to the original leaf node and [10|13] points to the new leaf node [8|9].

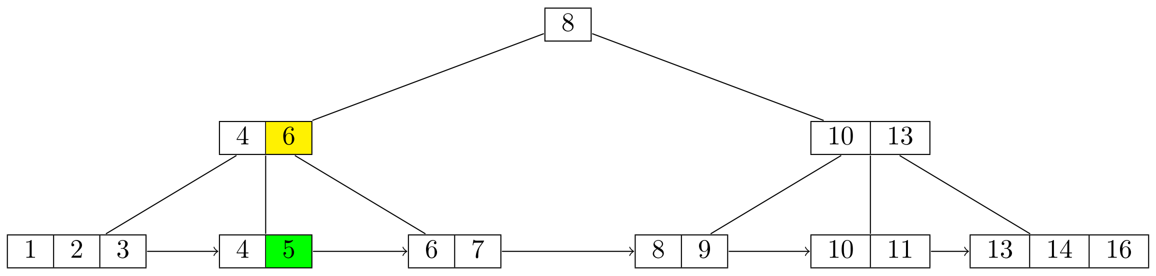


Insert 7 (Évora)



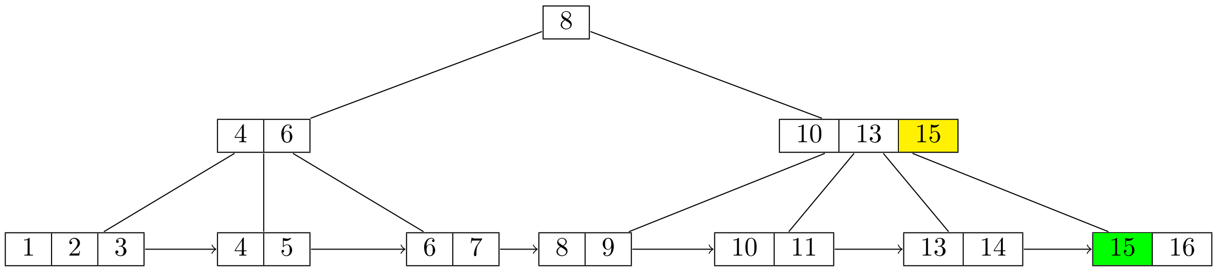
Insert 5 (Creme)

5 is inserted on leaf node [4,6,7], which will split into [4,5] and [6,7], 6 goes up.

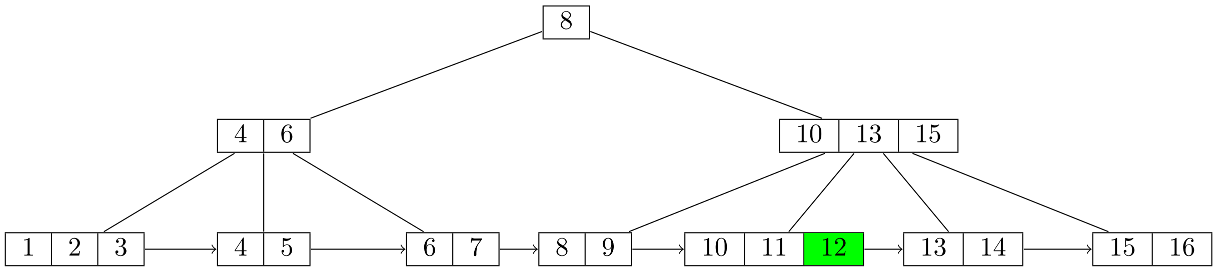


Insert 15 (Serpa)

15 is inserted on leaf node [13,14,16], which will split into [13,14] and [15,16], 15 goes up.



Insert 12 (Quark)

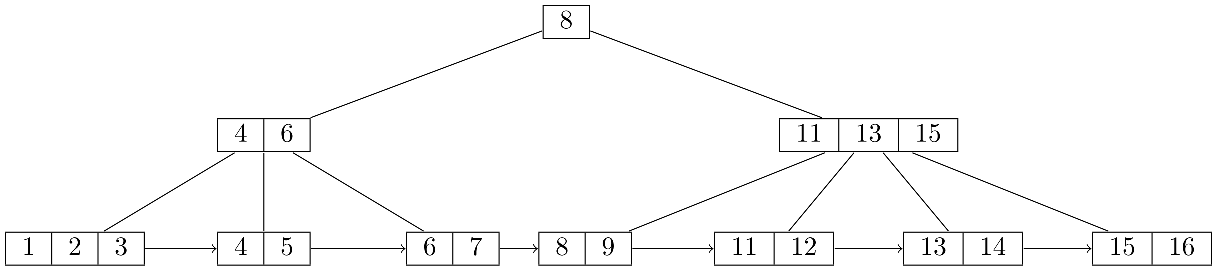


b)

Delete the following keys from the B+tree data structure from the previous exercise: Ilha; Flamengo; Emmental; Serpa. Draw the tree after each deletion.

Deletion order: 10, 8, 6, 15

Delete 10 (Ilha)



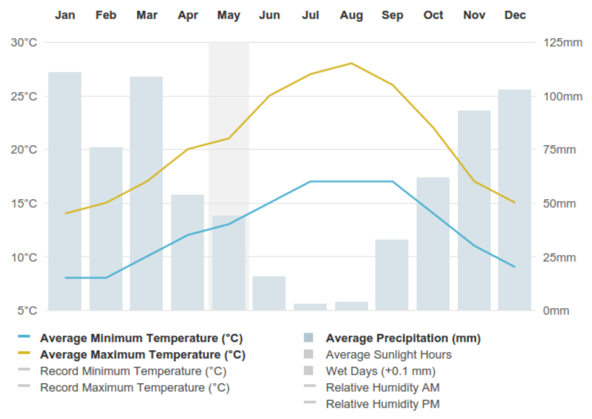
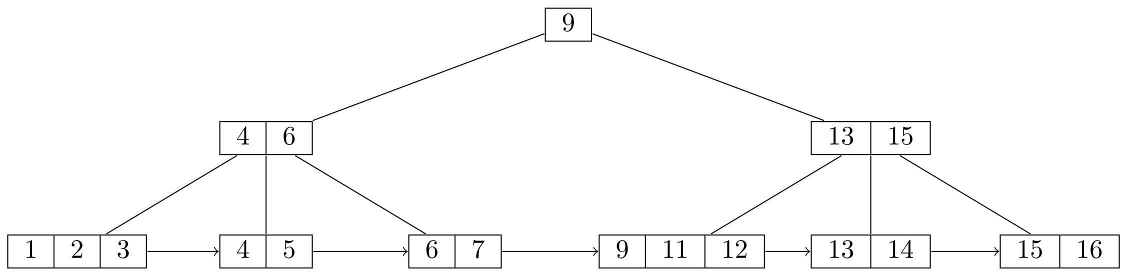


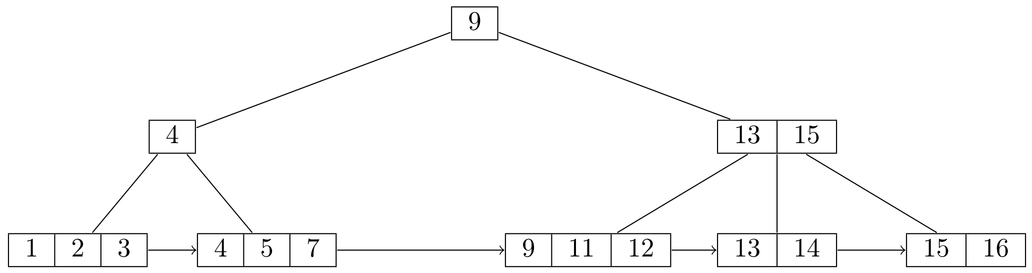
Figure 1: Average Conditions

Dry and clear this evening. Overnight, areas of mist and fog will gradually spread inland from the east, most extensive over northern areas, with only patchy fog further south. A grass frost is possible in the south under clear skies.

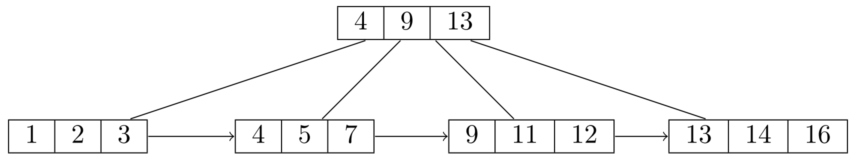
Delete 8 (Flamengo)



Delete 6 (Emmental)



Delete 15 (Serpa)



3 Extendable Hashing Index Schemes

Consider the extendable hashing indexing mechanism introduced in the theoretical classes. Show how data records with the following keys can be stored in buckets which individually can hold 2 records, using extendable hashing and considering the least significant bits first in the directory of the resulting data structure.

Azeitão, Camembert, Emmental, Serra , Camembert, Camembert, Creme, Alverca curado

Consider that binary representations for keys were determined by a simple hash function, resulting in:

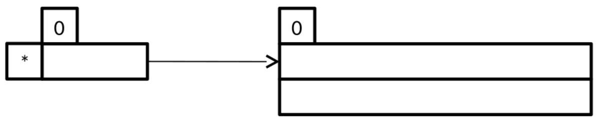


Figure 2: Estado inicial

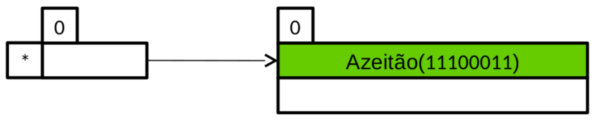


Figure 3: Azeitão - 11100011

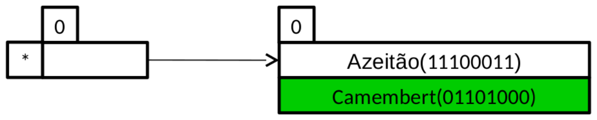


Figure 4: Camembert - 01101000

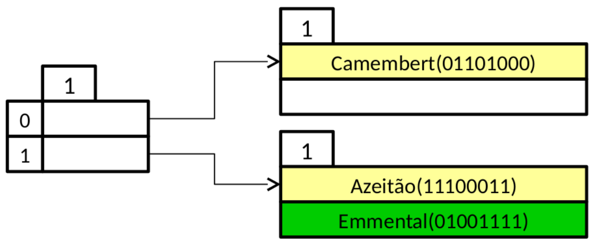


Figure 5: Emmental - 01001111

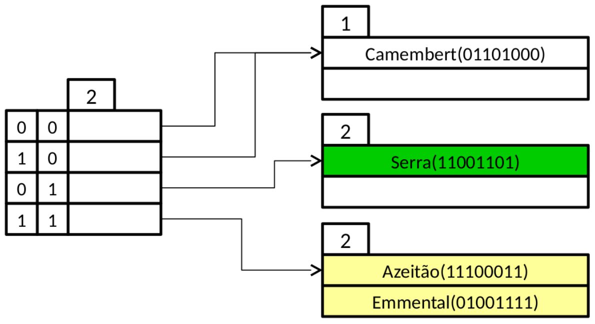


Figure 6: Serra - 11001101

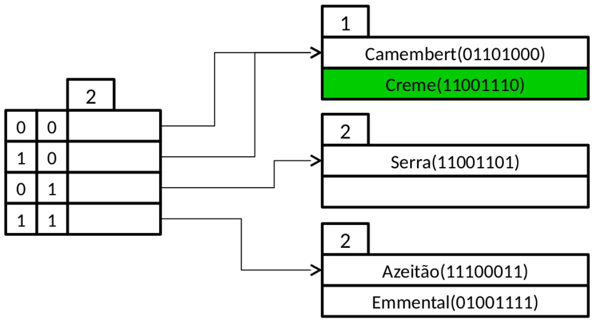


Figure 7: Creme - 11001110

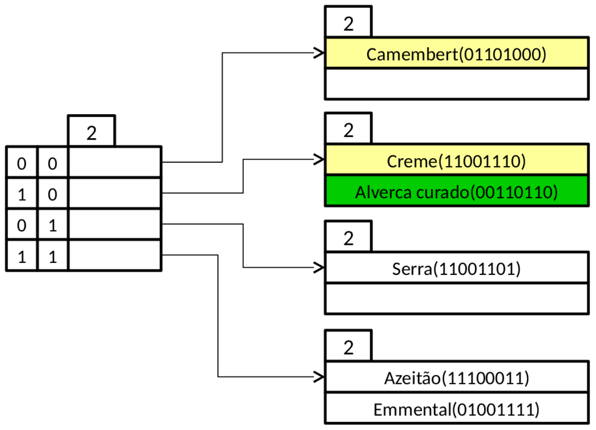


Figure 8: Alverca curado - 00110110

4 Estimating the Cost of Relational Algebra Operations

Consider the following relational schema:

CheeseProvenance(cheese-name, region-name)
Location(region-name, climate-type)

The relation CheeseProvenance stores information about the region where each cheese type is produced, and the relation Location stores information about the the regions that produce cheese.

All tuples have fixed size. The relation CheeseProvenance takes 1000 blocks and the relation Location has 2300 blocks. Each page of CheeseProvenance contains 120 tuples and each page of Location contains 100 tuples.

Compute the number of I/Os performed by each of the following algorithms:

Data:

$b_r = 1000$ blocks
$b_s = 2300$ blocks
$|R|/b_r = 120$ tuples/block
$|S|/b_s = 100$ tuples/block

a)

Selection on the Location relation where the filtering condition is climate-type = 'Dry', assuming there is an index on the table over the attribute climate-type.

$$\sigma_{climate-type='Dry'}(Location) \tag{1}$$

The selection operation is performed over an index, because this index isn't over the primary key, we can assume that it is non-clustering. Besides that, we are only testing for equality, that means that we must use algorithm 4 (A4) to retrieve multiple records (multiple locations).

$$Cost = (h_i + n) \tag{2}$$

Where:

n – number of records fetched (n=|S|, worst case)
h_i – height of B+ tree

With each I/O operation requiring a seek and a block transfer.

b)

Block Nested Loop Join, with CheeseProvenance as the outer relation and the join condition is on region-name. Present the costs of the worst and best cases.

$$CheeseProvenance \bowtie_{CheeseProvenance.region-name=Location.region-name} Location \tag{3}$$

Worst case:

$$Cost = (b_r * b_s + b_r)\text{block transfers} \tag{4}$$

Best case (if b_r and b_s fits in memory):

$$Cost = (b_r + b_s)\text{block transfers} \tag{5}$$

c)

Sort-Merge Join, assuming that only the relation Location is ordered on region-name, the relation CheeseProvenance is ordered on cheese-name and that you can have 3 pages in memory when sorting the relations.

$$CheeseProvenance \bowtie_{CheeseProvenance.region-name=Location.region-name} Location \tag{6}$$

Let:

$M = 3$ pages

$$Cost_{merge} = (b_r + b_s) \tag{7}$$

$$Cost_{sort} = 2b_s(\lceil log_{M-1}(b_s/M)\rceil + 1) \tag{8}$$

$$Cost = Cost_{sort} + Cost_{merge} \tag{9}$$

5 Query Optimization and Estimation of Join Sizes

Consider the two relations of Question 4. Consider also the following information, regarding the two relations:

$V(climate, Location) = 20$

Estimate the number of tuples that results from the expression:

$CheeseProvenance \bowtie (\sigma_{climate='dry'} Location)$

For $\sigma_{climate='dry'} Location$

2300 Location blocks

100 Tuples for each Location block

$n_{Location} = 2300 \times 100 = 230000$

$n_{DryLocation} = \frac{n_{Location}}{V(climate, Location)} = \frac{230000}{20} = 11500$

$Location \cap CheeseProvenance$ is a foreign key in $CheeseProvenance$ referencing $Location$ implies that the number of tuples resulting from $Location \bowtie CheeseProvenance$ is exactly the same as the number of tuples in $CheeseProvenance$

1000 CheeseProvenance blocks

120 Tuples for each CheeseProvenance block

$n_{CheeseProvenance} = 1000 \times 120 = 120000$

We know that, $(\sigma_{climate='dry'} Location) \bowtie CheeseProvenance$ produces the same results as $\sigma_{climate='dry'}(Location \bowtie CheeseProvenance)$

$n_{Result} = \frac{n_{CheeseProvenance}}{V(climate, Location)} = \frac{120000}{20} = 6000$

6 External talk: Casos Reais na Administração de Bases de Dados

Answer the following questions, on the subject of the invited talk given by engineer Wilson Lucas, that you had the opportunity to attend on the 24th of May 2015.

a)

One of the tasks of a DBA is to assure the high availability of the Databases being managed. This can be done in several ways. However, independently of the technique used, there are always trade-offs that one must take into account when choosing how to implement it (or even if it is worth implementing). Name and explain one such trade-off.

To guarantee data high availablity in a database, it's important to ensure that the information is ready to be accessed fastly and eficiently at any moment.

To ensure such availablity, there are multiple techniques that a database administrator can apply. One of those techniques consists on creating indexes over table columns. Creating indexes can increase a faster access to information. However, the space that an index occupies is considerable compared to the space that a table occupies. It is wise to create indexes only over the needed columns and not for every column, in order to minimize the impact of indexes on disk storage.

b)

In theory, once our database is fully optimized, it should not be necessary to change it any further. In practice, on a database that is being used in a functioning organization, this is not the case. Explain why.

In a functioning organization, the way that data is accessed may not be uniform, leading to access data through unsuitable indexes. Altough there are mechanisms to choose a good query plan, it may be slow due to inexistent indexes. The database administrator has the goal to understand what's the best way to access information and act to improve it. A good index today may be worse tomorrow.

Another factor to take in account, is that data creation and deletion tends to degrade the access to it, due to internal fragmentation. A good maintenance practice is to recreate indexes over time, so they can occupy less space and be accessed sequentially.

There is also the necessity do backup databases in order to decrease or prevent data loss caused by a failure. The existence of a database administrator is essential in critical situations like this, as data loss could have great costs for an organization.