Database Administration and Tuning
Mini-Project 1 - Report

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

April 5, 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.

```
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.

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.

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

a)

Draw the tree after each insertion.

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.

3 Extendable Hashing Index Schemes

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.

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).

$$Cost = (h_i + n) * (t_T + t_S) \tag{1}$$

[This is not the number of I/O's]

Where:

t_T – time to transfer one block
t_S – time for one seek
n – number of records fetched
h_i – height of B+ tree

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.

$$CheeseProvenance \bowtie_{CheeseProvenance.region-name=Location.region-name} Location \tag{2}$$

Worst case:

$$Cost = (b_r * b_s + b_r)\text{block transfers} + (2 * b_r)\text{seeks} \tag{3}$$

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

$$Cost = (b_r + b_s)\text{block transfers} + 2\text{seeks} \tag{4}$$

[Outputs? How many results?]

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.

$$CheeseProvenance \bowtie_{CheeseProvenance.region-name=Location.region-name} Location \tag{5}$$

Let:

$M = 3$ pages

$$Cost_{merge} = (b_r + b_s) \tag{6}$$

$$Cost_{sort} = 2b_s(\lceil log_{M-1}(b_s/M) \rceil + 1) \tag{7}$$

$$Cost = Cost_{sort} + Cost_{merge} \tag{8}$$

5 Query Optimization and Estimation of Join Sizes

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

a)

b)