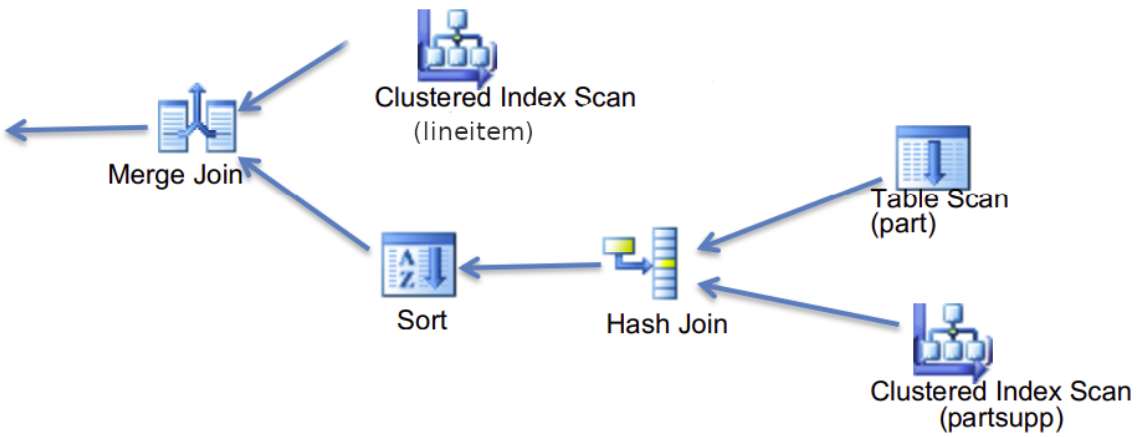
Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação
Sistemas Gerenciadores de Banco de Dados – CK0117 (T02)
Lista para composição da nota da 1ª. AP
Prof. Dr.-Ing. Angelo Brayner

Equipe:

José Douglas Gondim Soares - Matrícula: 485347
Fernanda Costa de Sousa – Matrícula: 485404

1) A complexidade esperada é: **EC = Pr + (Pr * (Ps/k))**. Isso porque agora temos uma área de buffer de tamanho k, o que diminui a quantidade de acesso a disco na ordem de k.

2) Para reduzir o tempo de execução da consulta nós podemos criar um índice primário na tabela *lineitem* sobre o atributo *l_partkey.* Com isso, nós não precisaremos mais fazer um *Table Scan*, fazendo apenas um *Index Scan*, que por definição tem tempo de execução esperado melhor que o *Table Scan.*



3) Read R
For each tuple ts ∈ IndexScan(S) do
While ts.b >= tr.a
if tr.a = ts.b
Result = Result U {(tr,ts)}
End-If
Read R
End-While
End-For

EC = PR + ((HTi(S) + M(S) - 1))
O tempo esperado irá aumentar em relação ao merge-join, pois precisaremos fazer um Index Scan na tabela S, que tem custo (HTi + M - 1).

4) Calculando a ordem da árvore:
$T = \lceil n \rceil * PT + \lfloor (n-1)/2 \rfloor * S$
$T = 8000, PT = 4, S = 4$
$8000 = n * 4 + ((n-1)/2) * 4$
$8000 = 4n + 2 (n-1)$
$n = \lfloor 1333,6 \rfloor = 1333.$

Cada página consegue armazenar 8000 bytes, e cada tupla ocupa 137 bytes, logo, conseguimos colocar no máximo 58 tuplas em cada página.

Como temos 6.002.000 tuplas, precisaremos de (6.002.000 / 58) = 103483 páginas no total.

Cálculo do M:
1 nó - 516
M nós - 103483

Assim, M = 201. Logo, podemos calcular a altura da árvore:
h = log M na base n/2
h = log 201 na base 1333
$h = \lceil 0,81 \rceil =$ **1**.