RENAN HENRIQUE GOMES DAMAZIO ASSUNÇÃO RA 21038114
ALEX ARANTES GONÇALVES RA 21011214

RELATÓRIO 1

Tópicos Emergentes em Bancos de Dados

Santo André – SP

2019

QUESTÃO 1

Foram criadas 4 relações Rel10000, Rel10000, Rel1000000 e Rel5000000 contendo apenas 2 atributos att1 bigint e att2 numeric, contendo cada uma 10 milhões de tuplas onde o valor de att2 foi gerado completamente randomicamente e o valor de att1 foi gerado de forma a cada relação possuir aproximadamente 10000, 100000, 1000000 e 5000000 valores distintos respectivamente.

Os scripts utilizados para esta questão encontram-se no arquivo Q1.sql. Para todos os teste desta questão foi removido o processamento paralelo e configurada memória de trabalho para 640MB, além de os tempos apresentados ignorarem o tempo de planejamento e serem expressos em milisegundos.

Teste 1 – Hash Aggregate

Este teste avaliou o desempenho do Hash Aggregate, forçando a utilização desse algoritmo através da desativação dos algoritmos de sort. Foram executados 3 testes com cada relação, utilizando a mesma consulta. A tabela 1 mostra os resultados obtidos, assim como a média.

Relação	Rel10000	Rel100000	Rel1000000	Rel5000000
Tempo1	5174	7015	8249	11298
Tempo2	4111	6184	7615	10080
Tempo3	4237	6185	7592	10063
Média	4507	6461	7819	10480

Tabela 1 – Resultados do teste 1

Teste 2 – Group Aggregate

Este teste avaliou o desempenho do Group Aggregate, fazendo exatamente o oposto ao anterior, forçando a utilização de algoritmos de sort e desativando os de hash. Assim como no teste anterior, foram executados 3 testes com cada relação, utilizando a mesma consulta. A tabela 2 mostra os resultados obtidos, assim como a média.

Relação	Rel10000	Rel100000	Rel1000000	Rel5000000
Tempo1	7406	7837	8662	10165
Tempo2	7462	7767	8647	10157
Tempo3	7425	7904	8534	10141
Média	7431	7837	8614	10154

Tabela 2 – Resultados do teste 2

O gráfico 1 traz a síntese de ambos os testes, onde o primeiro teste é destacado na cor azul e o segundo teste na cor laranja

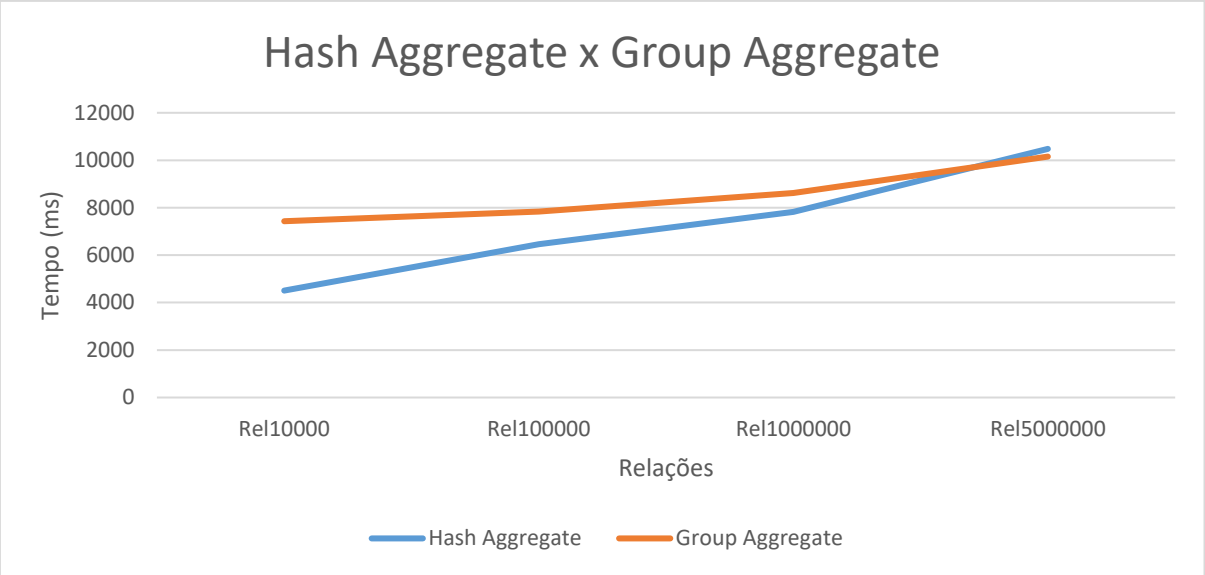

Gráfico 1 – Hash Aggregate x Group Aggregate

Por meio da análise do gráfico é possível fazer algumas conclusões:

I. O número de elementos distintos em att1 afeta diretamente o tempo de execução das consultas;

II. O método de Hash Aggregate teve um desempenho superior para as 3 primeiras relações e um desempenho muito similar para última relação, porém é possível observar que o primeiro teste do Hash Aggregate na última relação destoa completamente dos demais para a mesma relação, jogando a média para cima. Levando tudo isso em conta é possível dizer que o Hash Aggregate apresentou resultados superiores ao Group Aggregate.

QUESTÃO 2

Para a execução deste experimento cada uma das transações foi executava uma única vez para cada valor de memória de trabalho, sendo o bloco atual de 64KB e o futuro de 1MB. O tempo de planejamento de cada uma das transições foi ignorado devido ao baixo impacto para o experimento quando comparado com o tempo de execução. Para efeito de simplificar o experimento o tempo de execução foi truncado para ignorar a parte fracionário e será apresentado em milissegundos. Os resultados do experimento são apresentados na tabela 3.

Transação	Frequência	t Atual	t Dia Atual	t Futuro	t Dia Futuro
T1	405	109	44.145	89	36.045
T2	175	877.238	153.516.650	640.837	112.146.475
T3	89	101	8.989	107	9.523
T4	25	342	8.550	152	3.800
T5	158	104	16.432	103	16.274
T6	41	745	30.545	361	14.801
T7	81	106	8.586	98	7.938

Tabela 3 – Custos futuro e atual de cada Transação

O Gráfico 2 representa a diferença de performance entre os dois cenários de memória de trabalho

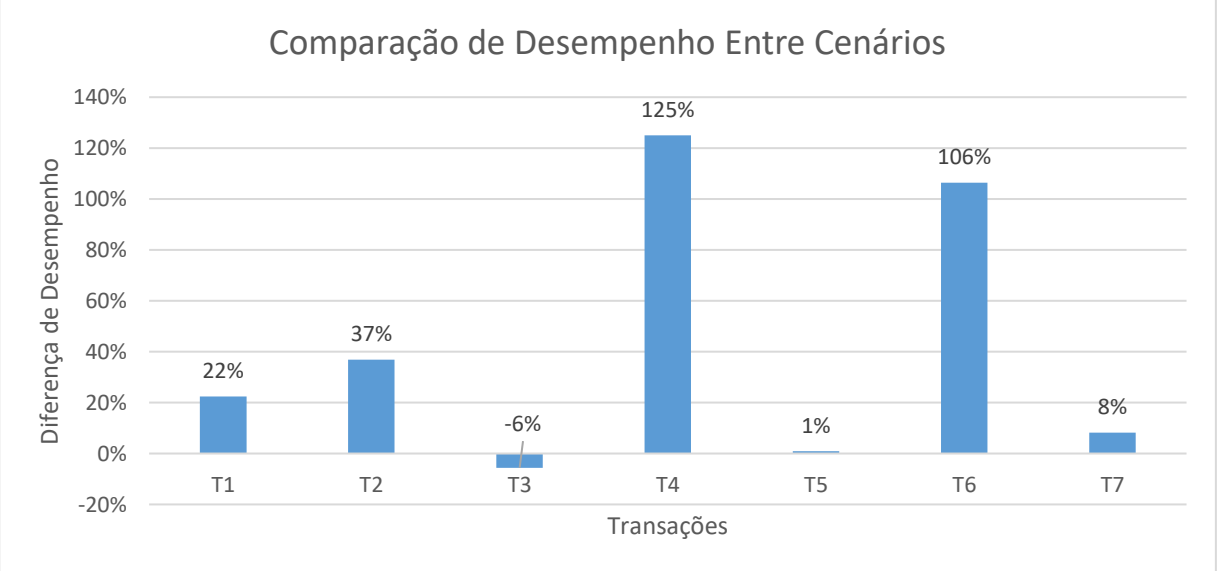

Gráfico 2 – Comparação de desempenho entre cenários

A partir da análise dos resultados do experimento é possível fazer as seguintes observações:

I. Fica claro que a transição T2 deve ser abolida em qualquer um dos cenários, pois apesar da melhora percentual de desempenho de um cenário para outro, ela pode ser facilmente substituída;
II. O aumento da memória de trabalho não afetou o fluxo de execução para nenhuma das transações;
III. As transações T4 e T6 tiveram o aumento de performance mais relevante com o aumento de memória de trabalho, o que se deve ao fato de na operação de hash utilizada nestas transações, o número de buckets ser proporcional a memória de trabalho. Porém ambas ainda continuam sendo uma má opção quando comparadas com as demais em qualquer um dos dois cenário;
IV. Tanto para o primeiro quanto para o segundo cenário recomenda-se a utilização das transações T1, T3, T5 e T7 e desencora-se a utilização das demais;
V. Comparando as transações T2 e T3 percebe-se que elas tem uma sintaxe muito parecida, porém o uso do operador "*not it*" tem um desempenho muito inferior ao do operador "*in*";
VI. O uso de join ao invés de subqueries também mostrou-se muito melhor para o desempenho das transações;