Projeto de Bases de Dados,

Parte 4

**Grupo 1**, BD225179L08 – Prof. Gabriel Pestana

Daniel Correia, 80697

Carolina Inês Xavier, 81172

Inês Leite, 81328

Esforço: 14 horas

**Índices**

**a)**

Na query 1, escolhemos índices hash sobre os atributos morada e código das tabelas Arrenda e Fiscaliza com função de dispersão dinâmica porque não temos testes de intervalos (para estes B+ seria mais adequado) mas sim testes de igualdade na operação de join em:

**Arrenda A inner join Fiscaliza F**

**on A.morada = F.morada and A.codigo = F.codigo**

Na query 2, escolhemos índices hash sobre os atributos morada e código das tabelas Posto e Aluga pelas mesmas razões da query anterior aplicados agora sobre a operação de natural join em:

**Posto P natural join Aluga A natural join Estado E**

Os índices hash são os melhores para seleção por igualdade porque cada chave (gerada pela função de dispersão do índice) corresponde a um bucket que armazena um conjunto de dados, logo a função de dispersão aponta logo para o bucket onde os dados procurados se encontram.

A função de dispersão seria dinâmica (ou seja, número de *buckets* a variar ao longo do ciclo de vida do índice) porque, quando adicionarmos novas entidades às tabelas em que estes índices são aplicados serão gerados novos buckets, evitando situações de overflow em algum bucket e reduzindo o tempo de procura dos dados dentro de cada bucket.

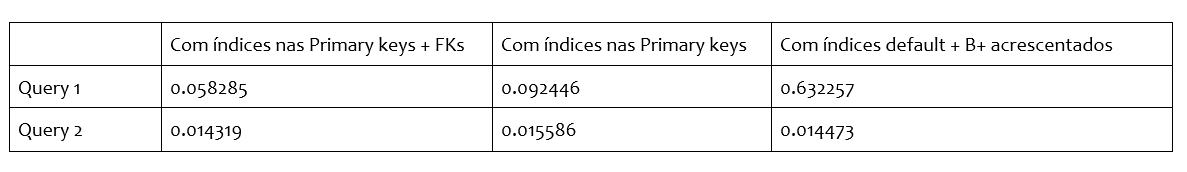
Para além dos índices hash na query 2, também escolhemos criar um índice bitmap sobre o atributo estado da tabela Estado porque este atributo tem um número reduzido de valores possíveis e a query faz um teste de igualdade sobre este atributo (**where E.estado=’aceite’** ) .

Neste caso, o uso de índices bitmap é mais vantajoso porque, sendo um array de bits em que o estado para cada registo é identificado por 1 bit, o índice bitmap será mais compacto (menor espaço em disco/memória) e os testes de igualdade realizados sobre este índice serão mais rápidos porque passa a ser apenas necessário verificar quais os registos com o bit de estado aceite a 1.

**b)** Em MySQL não é possível implementar bitmap e hash-indexes por isso não podemos testar mas sabemos que os índices da alínea anterior seriam os mais indicados para as queries analisadas. Como não é possível implementar os índices desejados, comparamos o desempenho das queries em três situações: com os indices disponiveis por default (índices de Foreign Key e de Primary Key), com índices default excepto os índices das Foreign Keys e com índices default acrescentando índices B+ feitos por nós.

É possível verificar o benefício destes índices comparando os tempos de execução das queries sem índices das Foreign Keys e os tempos com os índices *default* gerados pelo MySQL para as chaves primárias.

Os tempos obtidos foram:



Observando os tempos para correr as queries sem índices em relação às restantes situações, podemos concluir que os índices existentes por default para as Primary Keys e Foreign Keys das tabelas são os que permitem realizar as queries de maneira mais eficiente.

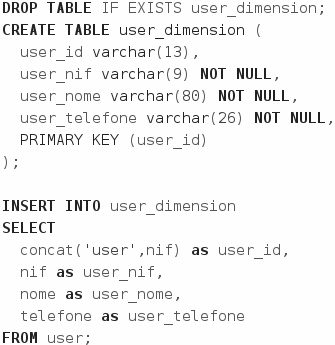
No caso dos índices acrescentados aos já existentes por default, estes não provocaram melhorias de desempenho porque são índices B+ criados sobre atributos de chaves primárias ou estrangeiras.

O uso dos índices B+ default levam a uma maior eficiência na realização das queries porque, sem qualquer índice implementado, para aceder a qualquer registo é necessário percorrer toda a tabela para se encontrar o registo desejado, enquanto que, com índices implementados, sabendo os registos a que queremos aceder, é possível aceder-lhes com menor número de comparações/testes através do seu índice.

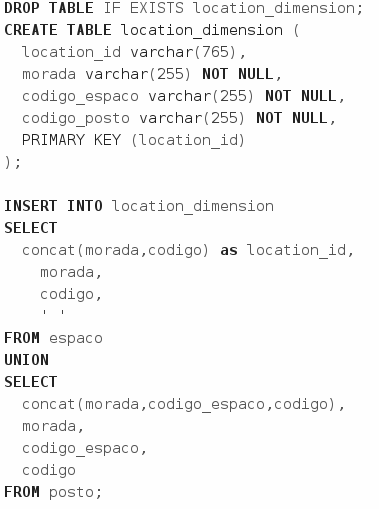
Os índices para os conjuntos morada e código são benéficos porque nas queries estes são os atributos acedidos nas operações, logo com um índice conjunto é possível aceder aos registos pretendidos utilizando menos operações.

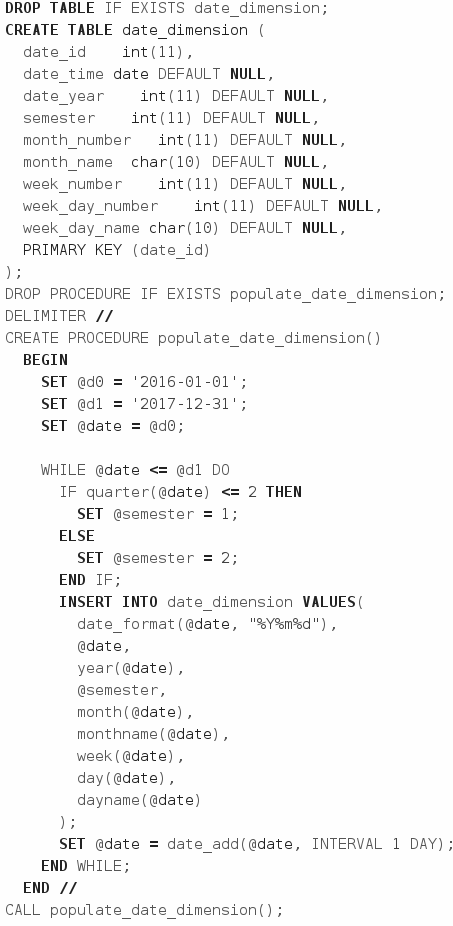
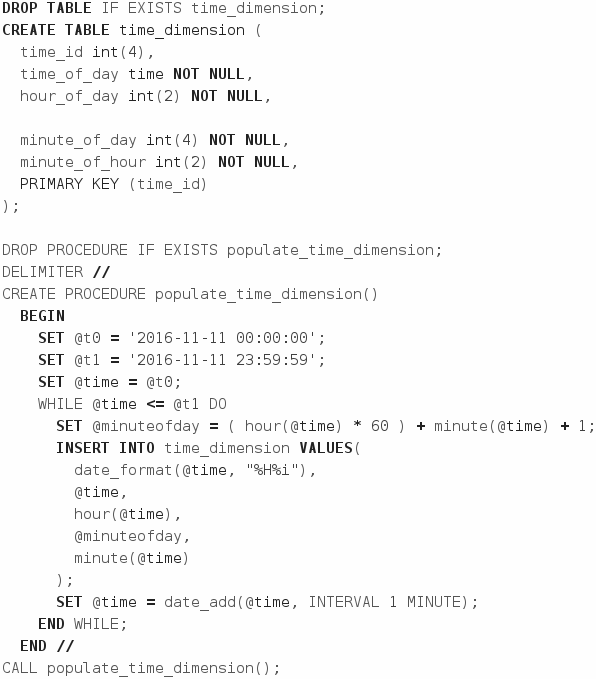
Com base nestas observações, os índices default estão implementados sobre os mesmos atributos para os quais gostaríamos de criar os índices hash (para ambas as queries). Como nestas queries são realizadas operações de join com estes atributos, um índice de hash sobre estes atributos seria mais eficiente do que o índice B+ existente por default, porque B+ é mais adequado para testes de intervalo, enquanto que o índice hash é mais adequado aos testes de igualdade realizados nas operações do tipo join.

**Data Warehouse**

User dimension

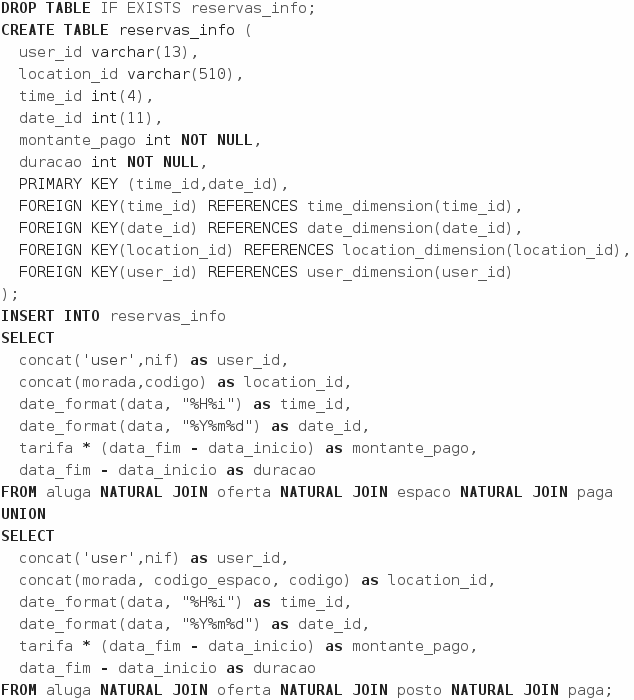
Location dimension

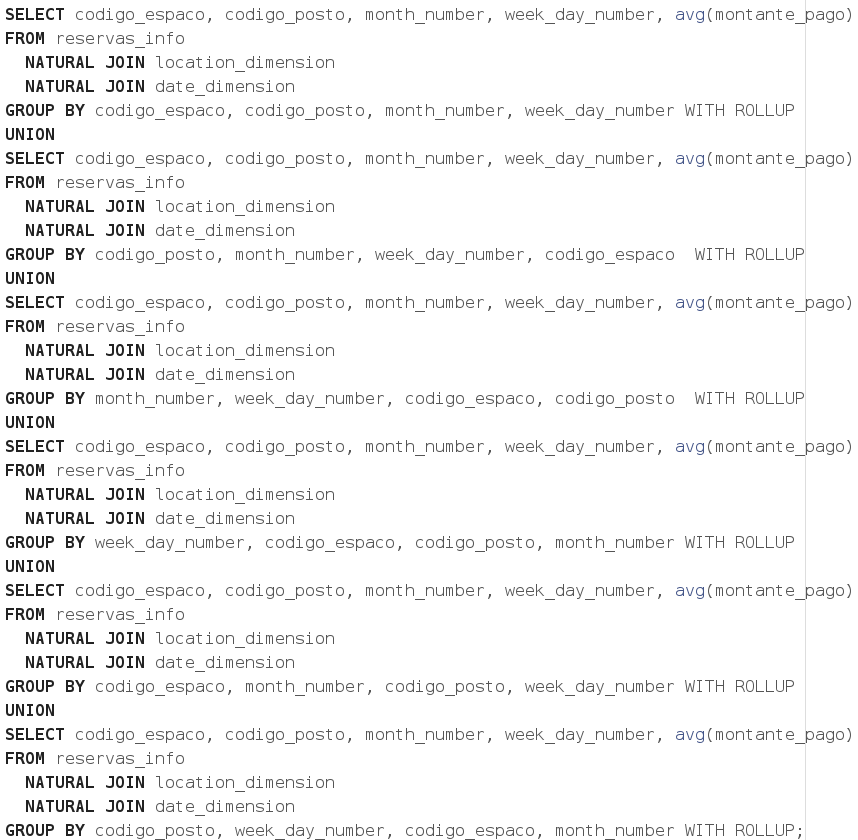


Date dimension

Time dimension

Reservas (Tabela de Factos)



Cube query