UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO UNIDADE ACADÊMICA DE EDUCAÇÃO A DISTÂNCIA E TECNOLOGIA BACHARELADO EM SISTEMAS DE INFORMAÇÃO

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

**ALEXSANDRO MATIAS DE ALMEIDA**

**ANÁLISE DE DESEMPENHO APÓS OTIMIZAÇÕES NO BANCO DE DADOS MYSQL**

PALMARES, 2021

**ALEXSANDRO MATIAS DE ALMEIDA**

**ANÁLISE DE DESEMPENHO APÓS OTIMIZAÇÕES NO BANCO DE DADOS MYSQL**

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Bacharelado em Sistemas de informação, pelo aluno ALEXSANDRO MATIAS DE ALMEIDA, sob orientação da professora Dra. Juliana Regueira Basto Diniz, para conclusão do Curso de Sistemas de Informação.

PALMARES, 2021

**AGRADECIMENTOS**

Agradeço, primeiramente, a Deus pela oportunidade dessa formação e por ter me dado saúde e força para superar as dificuldades.

A esta faculdade, seu corpo docente, direção, administração e funcionários que sempre me serviram da melhor maneira possível nos momentos que precisei.

A minha mãe, irmãs e esposa, por todo amor, paciência, incentivo e apoio incondicional que me deram durante a minha jornada no curso.

E a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

**RESUMO**

O presente estudo mensura o desempenho quando são realizados ajustes nas variáveis de configuração do sistema de Banco de Dados MySQL utilizando o Sistema Operacional Linux Debian Buster. Para realização deste teste será utilizado um modelo internacional para avaliação de benchmark chamado TPC-H que serve para medição de carga de trabalho auxiliando no suporte à decisão. Para isso, serão criadas quatro bases de dados: As duas primeiras serão criadas seguindo o modelo de criação e população das tabelas indicado pelo TPC-H, tendo estas os tamanhos de 1 GB e outra de 10 GB. A partir deste momento serão realizadas as 22 consultas propostas pelo teste de Benchmark e realizadas as respectivas medições do tempo de médio de cada consulta. Já na próxima etapa, serão realizadas as alterações nos valores das variáveis de configuração do MySQL como memória de cache e memoria buffer com o intuito de tornar as consultas ao banco mais rápidas. A partir dos dados coletados, será descrito se houve ganho ou perda no tempo médio nas consultas nessas bases de dados supostamente otimizadas.

**Palavras-Chave:** MySQL, Desempenho, TPC-H.

**ABSTRACT**

The present study analyzes how much performance is gained when several configuration adjustments are made in the MySQL Database, a process called tuning. To perform this performance test, an international benchmark model called TPC-H will be used to measure workload, helping to support the decision.

**Keywords:** Relacional Databases, Performance.

**LISTA DE FIGURAS**

[Figura 1 - Esquema TPC-H 16](file:///C:\Users\pb7p\git\automatizacao-mysql\final-doc\AlexsandroMatias-final-doc.docx#_Toc53440151)

**LISTA DE TABELAS**

[Tabela 1 - Tempo Acumulado da Query 01 18](#_Toc53443949)

[Tabela 2 - Tempo Médio das Consultas da Base de Dados não Otimizada 19](#_Toc53443950)

**LISTA DE SIGLAS**

BD – Banco de Dados.

DDL - Data Definition Language.

DML - Data Manipulation Language.

SGBD – Sistema de Gerenciamento de Banco de Dados.

SGBDR – Sistema de Gerenciamento de Banco de Dados Relacional.

SQL – Structured Query Language.

TPC – Transaction Performance Council.

**SUMÁRIO**

[1. INTRODUÇÃO 7](#_Toc53444120)

[1.1 OBJETIVOS 7](#_Toc53444121)

[1.1.1 OBJETIVO GERAL 7](#_Toc53444122)

[1.1.2 OBJETIVOS ESPECÍFICOS 7](#_Toc53444123)

[1.2 JUSTIFICATIVA 8](#_Toc53444124)

[2. FUNDAMENTAÇÃO TEÓRICA 8](#_Toc53444125)

[2.1 DADOS 9](#_Toc53444126)

[2.2 INFORMAÇÃO 9](#_Toc53444127)

[2.3 BANCO DE DADOS 9](#_Toc53444128)

[2.4 SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS 9](#_Toc53444129)

[2.5 BANCO DE DADOS MySQL 10](#_Toc53444130)

[2.5.1 DDL DATA DEFINITION LANGUAGE 11](#_Toc53444131)

[2.5.2 DML DATA MANIPULATION LANGUAGE 11](#_Toc53444132)

[2.5.3 DCL DATA CONTROL LANGUAGE 11](#_Toc53444133)

[2.6 CHAVES 12](#_Toc53444134)

[2.6.1 CHAVE PRIMÁRIA 12](#_Toc53444135)

[2.6.2 CHAVE ESTRANGEIRA 12](#_Toc53444136)

[2.6.3 INTEGRIDADE REFERENCIAL 12](#_Toc53444137)

[2.6.4 ÍNDICES 12](#_Toc53444138)

[2.6.5 PARTICIONAMENTO DE TABELAS 14](#_Toc53444139)

[2.7 TPC-H 14](#_Toc53444140)

[2.7.1 ENTIDADES E RELACIONAMENTOS DO BANCO DE DADOS 16](#_Toc53444141)

[3. PROCEDIMENTOS METODOLÓGICOS 17](#_Toc53444142)

[3.1 METODOLOGIA 17](#_Toc53444143)

[3.2 AMBIENTE DE TESTES 17](#_Toc53444144)

[3.3 COLETA DE DADOS 18](#_Toc53444145)

[3.4 BASE DE DADOS NÃO OTIMIZADA 18](#_Toc53444146)

[4. ANÁLISE DOS DADOS DA PESQUISA (OBTIDOS) 20](#_Toc53444147)

[5. LIMITAÇÕES DO TESTE 20](#_Toc53444148)

[6. CONSIDERAÇÕES FINAIS 20](#_Toc53444149)

[7. REFERÊNCIAS 21](#_Toc53444150)

1. INTRODUÇÃO

A utilização dos bancos de dados é indispensável em sistemas de informação para aplicações no campo empresarial. Isso se torna visível quando cinco dos bancos dados mais utilizados no mercado nos últimos cinco anos, quatro deles são relacionais. Estes são representados em ordem crescente em Oracle, MySQL, Microsoft SQL Server, PostgreSQL e MongoDB (DB-ENGINES, 2020).

Então, já que os banco MySQL está entre os cinco mais utilizados no mercado (DB-ENGINES, 2020), se torna interessante conhecimento mais específico quanto à performance do mesmo no que diz respeito ao tempo gasto nas operações de consulta dos dados quando aplicada determinada carga de dados no sistema.

Dito isto, na seção 2 deste trabalho serão apresentados os conceitos básicos, e aplicação desses bancos de dados, assim como serão explanados os comandos que viabilizam os testes. Na seção 3 será apresentada a metodologia e preparação do ambiente para o teste de carga, indicando quais os softwares (com suas respectivas versões) e hardware foram utilizados, além de tratar de algumas métricas preestabelecidas juntamente com as configurações especificas dos bancos de dados estudados. Já na seção 4 serão mostrados os resultados obtidos no teste de carga, para que se possa concluir qual o ganho percentual dessas otimizações.

* 1. OBJETIVOS

Serão apresentados os objetivos que nortearam este trabalho final de curso.

* + 1. OBJETIVO GERAL

Mensurar o desempenho após alterações no banco de dados MySQL utilizando como referência de Benchmark o padrão TPC-H.

* + 1. OBJETIVOS ESPECÍFICOS
* Realizar criação e população das tabelas nos bancos de dados MySQL utilizando o modelo TPC-H;
* Realizar medição de consultas aos registros sem otimizações;
* Analisar os resultados obtidos;
* Realizar as otimizações no mesmo banco de dados;
* Analisar o tempo gasto nessas consultas depois das otimizações.
  1. JUSTIFICATIVA

O aumento crescente da utilização dos bancos de dados é atribuído aos avanços nas tecnologias de sistemas de informação que consideram os dados de qualquer instituição como um bem intangível e extremamente valioso. Assim, com o crescimento exponencial, a disponibilidade, integridade, confidencialidade e forma de armazenamento desses dados são imprescindíveis neste processo de competitividade empresarial. Outra aplicação direta do uso de banco de dados é a manipulação de grandes volumes de dados visando também promover subsídios para a tomada de decisão.

1. FUNDAMENTAÇÃO TEÓRICA

Nesta seção são abordados conceitos básicos necessários para a compreensão dos principais temas abordados nesta pesquisa. De uma forma geral serão explanados os conceitos fundamentais em torno de Sistemas de Gerenciamento de Banco de Dados, os comandos SQL e suas particularidades.

* 1. SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS

Originalmente da sigla em inglês DBMS (Data Base Management System), ou Sistema de Gerenciamento de banco de dados (SGBD), é um pacote de softwares projetado cuja função é gerenciar uma base de dados. As vantagens do seu uso, de acordo com o que é proposto por Ramakrishnan & Gehrke (2009):

* Independência de Dados: Os programas aplicativos não devem, idealmente, ser expostos aos detalhes de representação e armazenamento de dados.
* Acesso Eficiente aos Dados: Um SGBD utiliza uma variedade de técnicas sofisticadas para armazenar e recuperar dados eficientemente.
* Integridade e Segurança dos Dados: Se os dados são sempre acessados através do SGBD, ele pode forçar restrições de integridade.
* Administração de Dados: Quando diversos usuários compartilham dados, centralizar a administração dos dados pode oferecer melhorias significativas.
* Acesso Concorrente e Recuperação de Falha: Um SGBD planeja o acesso concorrente aos dados de maneira tal que os usuários podem achar que os dados estão sendo acessados por apenas um único usuário de cada vez. Além disso, o SGBD protege os usuários dos efeitos de falhas de sistema.
* Tempo Reduzido de Desenvolvimento de Softwares: O SGBD suporta funções importantes que são comuns a várias linguagens de programação que acessam os dados no SGBD.

É importante ressaltar (RAMAKRISHNAN e GEHRKE, 2009) que algumas vezes, não se torna viável o uso de um SGBD. Isso se aplica já que se trata de um software complexo para executar determinadas aplicações, como por exemplo, responder a consultas complexas ou tratar várias requisições concorrentes. Por isso, seu desempenho pode não ser adequado para determinadas aplicações especificas. Entretanto, na maioria das situações em que é necessário gerenciamento de dados em grande escala, os SGBDs têm se tornado uma ferramenta indispensável.

* 1. BANCO DE DADOS MySQL

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, ou do inglês Structured Query Language) como interface. É atualmente um dos sistemas de gerenciamento de bancos de dados mais populares da Oracle Corporation, com mais de 10 milhões de instalações pelo mundo. Ele possui as seguintes características:

* É um Software Livre com base na GPL;
* Alta portabilidade já que suporta praticamente qualquer plataforma atual;
* Boa Compatibilidade com linguagens de programação sua variedade de drivers e módulos de interface para diversas linguagens de programação, como Delphi, Java, C/C++, C#, Visual Basic, Python, Perl, PHP, ASP e Ruby;
* Excelente desempenho e estabilidade;
* Pouco exigente quanto a recursos de novos hardwares;
* Facilidade no manuseio;
* Contempla a utilização de vários Storage Engines como MyISAM, InnoDB, Falcon, BDB, Archive, Federated, CSV, Solid;
* Suporta controle transacional;
* Suporta Triggers;
* Suporta Cursors (Non-Scrollable e Non-Updatable);
* Suporta Stored Procedures e Functions;
* Replicação facilmente configurável;

No próximo tópico serão expostos quais os comandos são utilizados nessa linguagem.

A linguagem utilizada no MySQL é a Structured Query Language que representa a linguagem usada nos SGBDs por padrão. No entanto, cada um tem suas particularidades dentro da própria linguagem, tendo implementações diferentes. A linguagem SQL tem algumas divisões, que facilitam o entendimento da mesma, categorizando seus comandos. Sendo que as mais conhecidas, que serão explicadas a seguir, são: DDL, DML e DCL.

* + 1. DDL DATA DEFINITION LANGUAGE

Linguagem de Definição de Dados, é a parte da Linguagem SQL que trata, como o próprio nome diz, da definição da estrutura dos dados, cujos efeitos se dão sobre objetos. Esses comandos são utilizados para a criação de bancos de dados, tabelas, views, triggers (ATANAZIO, 2019). Exemplos de comandos: CREATE (criação), ALTER (alteração), DROP (remoção).

* + 1. DML DATA MANIPULATION LANGUAGE

Linguagem de Manipulação de Dados, é a parte da Linguagem SQL que não altera a estrutura, mas sim os registros de uma base de dados, cujos efeitos se darão sobre registros (ATANAZIO, 2019). São comandos que fazem consultas, inserem, alteram ou apagam registros. Exemplos de comandos: SELECT (consulta), INSERT (inserção), UPDATE (alteração), DELETE (remoção).

* + 1. DCL DATA CONTROL LANGUAGE

Linguagem de Controle de Dados, é a parte da linguagem SQL referente ao controle de acesso a objetos por usuários e seus respectivos privilégios. Os principais comandos são:

* GRANT: Garante (permite) acesso dado a um usuário;
* REVOKE: Revoga (retira) direitos dados a um usuário. Os direitos dados a um usuário podem ser: ALL, CREATE, EXECUTE, REFERENCES, SELECT, TRIGGER, USAGE, CONNECT, DELETE, INSERT, RULE, TEMPORARY, UPDATE, etc.
  1. CHAVES
     1. CHAVE PRIMÁRIA

Uma chave primária é uma ou mais colunas que identificam exclusivamente uma linha. Nenhuma das colunas que fazem parte da chave primária pode ser anulável, ou seja, com ausência de algum valor. Uma tabela não deve ter mais do que uma chave primária (TPC, 2018).

* + 1. CHAVE ESTRANGEIRA

Uma chave estrangeira é uma coluna ou combinação de colunas usada para estabelecer a relação entre os dados em duas tabelas. Isso é possível adicionando a coluna ou colunas que contêm os valores de chave primária de uma tabela à outra tabela. Esta coluna se torna uma chave estrangeira na segunda tabela. Também pode ser referido como uma restrição de chave estrangeira (TPC, 2018).

* + 1. INTEGRIDADE REFERENCIAL

Integridade referencial é uma propriedade de dados pela qual uma chave estrangeira em uma tabela tem uma chave primária correspondente em uma tabela diferente (TPC, 2018).

* 1. Inno

Tipos de tradução

Tradução de textos

Texto de origem

4368 / 5000

Resultados de tradução

O que é um pool de buffer do InnoDB?

O pool de buffer do InnoDB é o espaço de memória que contém muitas estruturas de dados em memória do InnoDB, buffers, caches, índices e até mesmo dados de linha. innodb\_buffer\_pool\_size é o parâmetro de configuração do MySQL que especifica a quantidade de memória alocada para o pool de buffer InnoDB pelo MySQL. Esta é uma das configurações mais importantes na configuração de hospedagem MySQL e deve ser definida com base na RAM do sistema disponível.

Neste artigo, vamos guiá-lo por duas abordagens para definir o valor do tamanho do buffer pool do InnoDB, examinar os prós e contras dessas práticas e também propor um método exclusivo para chegar a um valor ideal com base no tamanho da RAM do seu sistema .

Abordagem 1. Método da regra prática

A prática mais comumente seguida é definir esse valor em 70% - 80% da RAM do sistema. Embora funcione bem na maioria dos casos, esse método pode não ser ideal em todas as configurações. Vejamos o exemplo de um sistema com 192 GB de RAM. Com base no método acima, chegamos a cerca de 150 GB para o tamanho do buffer pool. No entanto, este não é realmente um número ideal, pois não aproveita totalmente o grande tamanho de RAM que está disponível no sistema e deixa para trás cerca de 40 GB de memória. Essa diferença pode ser ainda mais significativa à medida que mudamos para sistemas com configurações maiores, onde deveríamos estar utilizando a RAM disponível em maior extensão.

Abordagem 2. Uma abordagem mais diferenciada

Essa abordagem é baseada em uma compreensão mais detalhada das partes internas do pool de buffers do InnoDB e suas interações, que é muito bem descrito no livro High Performance MySQL.

Vejamos o seguinte método para calcular o tamanho do buffer pool do InnoDB.

Comece com RAM total disponível.

Subtraia a quantidade adequada para as necessidades do sistema operacional.

Subtraia a quantidade adequada para todas as necessidades do MySQL (como vários buffers do MySQL, tabelas temporárias, pools de conexão e buffers relacionados à replicação).

Divida o resultado por 105%, que é uma aproximação da sobrecarga necessária para gerenciar o próprio buffer pool.

Por exemplo, vamos olhar para um sistema com 192 GB de RAM usando apenas InnoDB e tendo um tamanho total de arquivo de log de cerca de 4 GB. Podemos usar uma regra como "máximo de 2 GB ou 5% do total de RAM" para a alocação das necessidades do sistema operacional, conforme recomendado no livro acima, que chega a cerca de 9,6 GB. Então, também vamos alocar cerca de 4 GB para outras necessidades do MySQL, principalmente levando em consideração o tamanho do arquivo de log. Este método resulta em cerca de 170 GB para o tamanho do pool de buffer InnoDB, que é cerca de 88,5% de utilização do tamanho de RAM disponível.

Embora tenhamos usado a regra 'máximo de 2 GB ou 5% de RAM total' para calcular nossa alocação de memória para as necessidades do SO acima, a mesma regra não funciona muito bem em todos os casos, especificamente para sistemas com RAMs de tamanho médio entre 2 GB e 32 GB . Por exemplo, em um sistema com 3 GB de RAM, alocar 2 GB para as necessidades do sistema operacional não deixa muito para o buffer pool do InnoDB, ao passo que alocar 5% da RAM é muito pouco para as necessidades do nosso sistema operacional.

Portanto, vamos ajustar a regra de alocação de SO acima e examinar o método de computação InnoDB em várias configurações de RAM:

Para sistemas com RAM de tamanho pequeno (<= 1 GB)

Para sistemas rodando com menos de 1 GB de RAM, é melhor ir com o valor de configuração padrão do MySQL de 128 MB para o tamanho do buffer pool do InnoDB.

Para sistemas com RAM de tamanho médio (1 GB - 32 GB)

Considerando o caso de sistemas com um tamanho de RAM de 1 GB - 32 GB, podemos calcular as necessidades do sistema operacional usando esta heurística aproximada:

256 MB + 256 \* log2 (tamanho da RAM em GB)

A racionalização aqui é que, para configurações de RAM baixa, começamos com um valor base de 256 MB para as necessidades do SO e aumentamos essa alocação em uma escala logarítmica conforme a quantidade de RAM aumenta. Dessa forma, podemos criar uma fórmula determinística para alocar RAM para nossas necessidades de sistema operacional. Também alocaremos a mesma quantidade de memória para nossas outras necessidades do MySQL. Por exemplo, em um sistema com 3 GB de RAM, faríamos uma alocação justa de 660 MB para as necessidades do sistema operacional e outros 660 MB para outras necessidades do MySQL, resultando em um valor de cerca de 1,6 GB para o tamanho do nosso buffer pool InnoDB.

Para sistemas com RAM de maior tamanho (> 32 GB)

Para sistemas com tamanhos de RAM maiores que 32 GB, voltaríamos a calcular as necessidades do sistema operacional como 5% do tamanho da RAM do nosso sistema e a mesma quantidade para outras necessidades do MySQL. Portanto, para um sistema com um tamanho de RAM de 192 GB, nosso método pousará em cerca de 165 GB para o tamanho do buffer pool do InnoDB, que é, novamente, um valor ideal a ser usado.

Mais sobre o texto original

É necessário fornecer o texto original para ver mais informações sobre a tradução

Enviar feedback

Painéis laterais

* + 1. INNODB\_BUFFER\_POOL\_SIZE

O tamanho em bytes do buffer pool, a área de memória onde o InnoDB armazena em cache a tabela e o índice dados. O valor padrão é 134217728 bytes (128 MB). O valor máximo depende da CPU arquitetura; o máximo é 4294967295 (232-1) em sistemas de 32 bits e 18446744073709551615 (264-1) em sistemas de 64 bits. Em sistemas de 32 bits, a arquitetura da CPU e o sistema operacional podem impor um tamanho máximo prático inferior ao máximo declarado. Quando o tamanho do buffer pool é maior que 1 GB, definindo innodb\_buffer\_pool\_instances para um valor maior que 1 pode melhorar a escalabilidade em um servidor ocupado (MYSQL, 2020).

Um buffer pool maior requer menos E / S de disco para acessar os mesmos dados da tabela mais de uma vez. Com um servidor de banco de dados dedicado, você pode definir o tamanho do buffer pool para 80% do tamanho físico da máquina Tamanho da memória. Esteja ciente dos seguintes problemas potenciais ao configurar o tamanho do buffer pool e seja preparado para reduzir o tamanho do buffer pool, se necessário.

* A competição pela memória física pode causar paginação no sistema operacional;
* InnoDB reserva memória adicional para buffers e estruturas de controle, de modo que o total alocado o espaço é aproximadamente 10% maior que o tamanho do buffer pool especificado;
* O espaço de endereço do buffer pool deve ser contíguo, o que pode ser um problema no Windows sistemas com DLLs que carregam em endereços específicos;
* O tempo para inicializar o buffer pool é aproximadamente proporcional ao seu tamanho. Em instâncias com grande pools de buffer, o tempo de inicialização pode ser significativo. Para reduzir o período de inicialização, você pode salvar o estado do buffer pool no desligamento do servidor e restaure-o na inicialização do servidor. Consulte a Seção 15.8.3.6, “Salvando e restaurando o estado do buffer pool”.

Ao aumentar ou diminuir o tamanho do buffer pool, a operação é executada em blocos. O tamanho do pedaço é definido pela variável innodb\_buffer\_pool\_chunk\_size, que possui um padrão de 128 MB. O tamanho do buffer pool deve ser sempre igual ou múltiplo de innodb\_buffer\_pool\_chunk\_size \* innodb\_buffer\_pool\_instances. Se você alterar o tamanho do buffer pool para um valor que não seja igual para ou um múltiplo de innodb\_buffer\_pool\_chunk\_size \* innodb\_buffer\_pool\_instances, o tamanho do buffer pool é ajustado automaticamente para um valor igual ou múltiplo de innodb\_buffer\_pool\_chunk\_size \* innodb\_buffer\_pool\_instances.

innodb\_buffer\_pool\_size pode ser definido dinamicamente, o que permite redimensionar o buffer pool sem reiniciar o servidor. Os relatórios de variável de status Innodb\_buffer\_pool\_resize\_status o status das operações de redimensionamento do buffer pool online. Veja a Seção 15.8.3.1, “Configurando o Buffer InnoDB Tamanho da piscina ”para mais informações.

Se innodb\_dedicated\_server estiver habilitado, o valor innodb\_buffer\_pool\_size é configurada automaticamente se não for definida explicitamente. Para obter mais informações, consulte a Seção 15.8.12, “Habilitando a configuração automática para um servidor MySQL dedicado”.

* + 1. INNODB\_BUFFER\_POOL\_INSTANCES

O número de regiões nas quais o buffer pool do InnoDB é dividido. Para sistemas com buffer pools em o intervalo de vários gigabytes, dividindo o pool de buffer em instâncias separadas pode melhorar a simultaneidade, reduzindo a contenção à medida que diferentes threads leem e gravam nas páginas em cache. Cada página que é armazenada em ou lido do buffer pool é atribuído a uma das instâncias do buffer pool aleatoriamente, usando uma função de hash. Cada buffer pool gerencia suas próprias listas gratuitas, listas de liberação, LRUs e todos os outros dados estruturas conectadas a um pool de buffer e são protegidas por seu próprio mutex de pool de buffer.

Esta opção só tem efeito ao definir innodb\_buffer\_pool\_size para 1 GB ou mais. O total O tamanho do buffer pool é dividido entre todos os buffer pools. Para melhor eficiência, especifique uma combinação de innodb\_buffer\_pool\_instances e innodb\_buffer\_pool\_size para que cada buffer pool instância é de pelo menos 1 GB.

O valor padrão em sistemas Windows de 32 bits depende do valor de innodb\_buffer\_pool\_size, conforme descrito abaixo:

• Se innodb\_buffer\_pool\_size for maior que 1,3 GB, o padrão para innodb\_buffer\_pool\_instances é innodb\_buffer\_pool\_size / 128 MB, com solicitações de alocação de memória para cada bloco. 1,3 GB foi escolhido como o limite no qual há risco significativo para o Windows de 32 bits ser incapaz de alocar o espaço de endereço contíguo necessário para um único buffer pool.

• Caso contrário, o padrão é 1.

Em todas as outras plataformas, o valor padrão é 8 quando innodb\_buffer\_pool\_size é maior que ou igual a 1 GB. Caso contrário, o padrão é 1. Para obter informações relacionadas, consulte a Seção 15.8.3.1, “Configurando o tamanho do pool de buffers do InnoDB”.

* + 1. INNODB\_BUFFER\_POOL\_CHUNK\_SIZE

innodb\_buffer\_pool\_chunk\_size define o tamanho do bloco para redimensionamento do buffer pool do InnoDB operações.

Para evitar a cópia de todas as páginas do buffer pool durante as operações de redimensionamento, a operação é executada em “pedaços”. Por padrão, innodb\_buffer\_pool\_chunk\_size é 128 MB (134217728 bytes).

O número de páginas contidas em um chunk depende do valor de innodb\_page\_size. innodb\_buffer\_pool\_chunk\_size pode ser aumentado ou diminuído em unidades de 1 MB (1048576 bytes). As seguintes condições se aplicam ao alterar o valor innodb\_buffer\_pool\_chunk\_size:

• Se innodb\_buffer\_pool\_chunk\_size \* innodb\_buffer\_pool\_instances é maior do que o tamanho do buffer pool atual quando o buffer pool é inicializado, innodb\_buffer\_pool\_chunk\_size é truncado para innodb\_buffer\_pool\_size/ innodb\_buffer\_pool\_instances.

• O tamanho do buffer pool deve ser sempre igual ou múltiplo de innodb\_buffer\_pool\_chunk\_size \* innodb\_buffer\_pool\_instances. Se você alterar innodb\_buffer\_pool\_chunk\_size, innodb\_buffer\_pool\_size é automaticamente arredondado para um valor igual ou múltiplo de innodb\_buffer\_pool\_chunk\_size \* innodb\_buffer\_pool\_instances. O ajuste ocorre quando o buffer pool é inicializado. Importante Deve-se ter cuidado ao alterar innodb\_buffer\_pool\_chunk\_size, pois a alteração desse valor pode aumentar automaticamente o tamanho do buffer pool.

Antes de alterar innodb\_buffer\_pool\_chunk\_size, calcule seu efeito em innodb\_buffer\_pool\_size para garantir que o tamanho do buffer pool resultante é aceitável.

Para evitar possíveis problemas de desempenho, o número de pedaços (innodb\_buffer\_pool\_size / innodb\_buffer\_pool\_chunk\_size) não deve exceder 1000.

A variável innodb\_buffer\_pool\_size é dinâmica, o que permite redimensionar o buffer pool enquanto o servidor está online. No entanto, o tamanho do buffer pool deve ser igual ou múltiplo de innodb\_buffer\_pool\_chunk\_size \* innodb\_buffer\_pool\_instances, e mudando dessas configurações de variáveis ​​requer reiniciar o servidor.

Desde o MySQL 5.7.5, podemos redimensionar dinamicamente o pool de buffers do InnoDB. Esse novo recurso também introduziu uma nova variável innodb\_buffer\_pool\_chunk\_size - que define o tamanho do bloco pelo qual o buffer pool é ampliado ou reduzido. Esta variável não é dinâmica e se for configurada incorretamente, pode levar a situações indesejadas. A representação da interação entre as o innodb\_buffer\_pool\_size, innodb\_buffer\_pool\_instances e innodb\_buffer\_pool\_chunk\_size interagem:



O buffer pool pode conter várias instâncias e cada instância é dividida em partes. Existem algumas informações que devemos levar em consideração: o número de instâncias pode ir de 1 a 64 e a quantidade total de blocos não deve exceder 1000.

Portanto, para um servidor com 3 GB de RAM, um pool de buffer de 2 GB com 8 instâncias e blocos no valor padrão (128 MB), obteremos 2 blocos por instância:

De acordo com a figura, pode-se observar que haverá 16 blocos.

Não vou explicar os benefícios de ter várias instâncias, vou me concentrar nas operações de redimensionamento. Por que você deseja redimensionar o pool de buffer? Bem, existem vários motivos, tais como:

* em um servidor virtual, você pode adicionar mais memória dinamicamente;
* para um servidor físico, você pode querer reduzir o uso de memória do banco de dados para abrir caminho para outros processos;
* em sistemas onde o tamanho do banco de dados é menor do que a RAM disponível
* se você espera um grande crescimento e deseja aumentar o buffer pool sob demanda
  + 1. THREAD\_CACHE\_SIZE

Quantos threads o servidor deve armazenar em cache para reutilização. Quando um cliente se desconecta, o cliente encadeamentos são colocados no cache se houver menos de encadeamentos thread\_cache\_size lá. solicitações de para encadeamentos são satisfeitos reutilizando encadeamentos retirados do cache, se possível, e somente quando o cache is empty é um novo thread criado. Esta variável pode ser aumentada para melhorar o desempenho se você tiver muitas novas conexões. Normalmente, isso não fornece um desempenho notável melhoria se você tiver uma boa implementação de thread. No entanto, se o seu servidor vê centenas de conexões por segundo, você normalmente deve definir thread\_cache\_size alto o suficiente para que a maioria novas conexões usam threads em cache. Ao examinar a diferença entre as conexões e Variáveis de status Threads\_created, você pode ver o quão eficiente é o cache de threads. Para detalhes, veja Seção 5.1.10, “Variáveis de status do servidor”.

O valor padrão é baseado na seguinte fórmula, limitada a um limite de 100:

8 + (max\_connections / 100)

* + 1. KEY\_BUFFER\_SIZE

Blocos de índice para tabelas MyISAM são armazenados em buffer e compartilhados por todas as threads. key\_buffer\_size é o tamanho do buffer usado para blocos de índice. O buffer de chave também é conhecido como cache de chave.

A configuração máxima permitida para key\_buffer\_size é 4 GB − 1 em plataformas de 32 bits. Maior os valores são permitidos para plataformas de 64 bits. O tamanho máximo efetivo pode ser menor, dependendo em sua RAM física disponível e limites de RAM por processo impostos pelo seu sistema operacional ou plataforma de hardware. O valor desta variável indica a quantidade de memória solicitada. Internamente, o servidor aloca o máximo de memória possível até essa quantidade, mas a alocação real pode ser menos.

Você pode aumentar o valor para obter melhor tratamento do índice para todas as leituras e várias gravações; em um sistema cuja função principal é executar o MySQL usando o mecanismo de armazenamento MyISAM, 25% da máquina memória total é um valor aceitável para esta variável. No entanto, você deve estar ciente de que, se você tornar o valor muito grande (por exemplo, mais de 50% da memória total da máquina), seu sistema pode começar a paginar e se tornar extremamente lento. Isso ocorre porque o MySQL depende da operação sistema para executar o cache do sistema de arquivos para leituras de dados, então você deve deixar algum spaço para o arquivo cache do sistema. Você também deve considerar os requisitos de memória de quaisquer outros mecanismos de armazenamento que você pode usar além do MyISAM.

Para obter ainda mais velocidade ao escrever muitas linhas ao mesmo tempo, use LOCK TABLES. Vejo Seção 8.2.5.1, “Otimizando instruções INSERT”. Você pode verificar o desempenho do buffer de chave emitindo uma instrução SHOW STATUS e examinando Key\_read\_requests, Key\_reads, Key\_write\_requests e Variáveis ​​de status de key\_writes. (Consulte a Seção 13.7.7, “Demonstrações SHOW”.) Key\_reads / A proporção key\_read\_requests normalmente deve ser menor que 0,01. The Key\_writes / A proporção de key\_write\_requests é geralmente próxima de 1 se você estiver usando principalmente atualizações e exclusões, mas pode ser muito menor se você tende a fazer atualizações que afetam muitas linhas ao mesmo tempo ou se você estiver usando a opção de tabela DELAY\_KEY\_WRITE.

A fração do buffer de chave em uso pode ser determinada usando key\_buffer\_size em conjunto com a variável de status Key\_blocks\_unused e o tamanho do bloco de buffer, que está disponível no variável de sistema key\_cache\_block\_size:

1 - ((Key\_blocks\_unused \* key\_cache\_block\_size) / key\_buffer\_size)

Este valor é uma aproximação porque algum espaço no buffer de chave é alocado internamente para estruturas administrativas. Fatores que influenciam a quantidade de sobrecarga para essas estruturas inclui o tamanho do bloco e o tamanho do ponteiro. Conforme o tamanho do bloco aumenta, a porcentagem do buffer de chave perdida para a sobrecarga tende a diminuir. Blocos maiores resultam em um número menor de operações de leitura (porque mais chaves são obtidas por leitura), mas, inversamente, um aumento nas leituras de chaves que não são examinadas (se nem todas as chaves em um bloco forem relevantes para uma consulta).

É possível criar vários caches de chaves MyISAM. O limite de tamanho de 4 GB se aplica a cada cache individualmente, não como um grupo. Consulte a Seção 8.10.2, “Cache de chave MyISAM”.

* 1. TPC-H

O Benchmark TPC-H é um benchmark de suporte à decisão. Ele consiste em um conjunto de consultas orientadas para os negócios e modificações de dados simultâneas (TPC, 2018). As consultas e os dados que povoam o banco de dados foram escolhidos para ter ampla relevância em todo o setor, mantendo um grau suficiente de facilidade na sua implementação. Alguns tópicos que ilustram esse sistema são (TPC, 2018):

* Examinar grandes volumes de dados;
* Executar consultas com alto grau de complexidade;
* Dar respostas a perguntas de negócios do mundo real;
* Inclui uma ampla variedade de operadores e restrições de seletividade;
* Gerar atividade intensiva por parte do componente servidor de banco de dados do sistema em teste;
* São executados em um banco de dados em conformidade com a população específica e requisitos de escala;

O TPC acredita que as comparações dos resultados do TPC-H medidos em relação a diferentes tamanhos de banco de dados são enganosas e desencoraja tais comparações.

O banco de dados TPC-H deve ser implementado usando um sistema de gerenciamento de banco de dados (DBMS) disponível comercialmente e as consultas executadas por meio de uma interface usando SQL dinâmico. A especificação fornece variantes de SQL, pois os implementadores não precisam ter implementado um padrão SQL específico por completo.

O TPC-H usa terminologia e métricas semelhantes a outros benchmarks, originados pelo TPC e outros. Tal similaridade na terminologia não implica de forma alguma que os resultados do TPC-H sejam comparáveis ​​a outros benchmarks. Os únicos resultados de benchmark comparáveis ​​ao TPC-H são outros resultados do TPC-H compatíveis com a mesma revisão.

Apesar do fato de que este benchmark oferece um ambiente rico representativo de muitos sistemas de suporte à decisão, este benchmark não reflete toda a gama de requisitos de suporte à decisão. Além disso, até que ponto um cliente pode alcançar os resultados relatados por um fornecedor depende muito da proximidade da TPC-H com o aplicativo do cliente. O desempenho relativo dos sistemas derivados deste benchmark não se aplica necessariamente a outras cargas de trabalho ou ambientes. Extrapolações para qualquer outro ambiente não são recomendadas.

Os resultados do benchmark são altamente dependentes da carga de trabalho, requisitos de aplicativos específicos e projeto e implementação de sistemas. O desempenho relativo do sistema varia como resultado desses e de outros fatores. Portanto, o TPC-H não deve ser usado como um substituto para um benchmarking de aplicativo de cliente específico quando o planejamento de capacidade crítica e / ou decisões de avaliação de produto são contemplados.

* + 1. ENTIDADES E RELACIONAMENTOS DO BANCO DE DADOS

As tabelas e relacionamentos que compõem o esquema do banco de dados do benchmark TPC-H são representados por CUSTOMER, NATION, PART, PARTSUPP, REGION, SUPPLIER, ORDERS e LINEITEM. Estas tabelas, assim como os seus relacionamentos são ilustrados na figura 1 abaixo (TPC, 2018).



Figura 1 - Esquema TPC-H – fonte da imagem

* + 1. GERAÇÃO DOS DADOS E POPULAÇÃO DAS TABELAS

A geração dos dados para popular a base do benchmark TPC-H, é realizada através da ferramenta escrita em ANSI C denominada de DBGEN (Database Generator). Este programa tem a função de automatizar a criação dos dados das tabelas utilizando comando em linhas de texto com intuito de gerar a base de dados com o tamanho determinado no momento da compilação e execução do programa em C.

Fatores de escala usados ​​para o banco de dados de teste devem ser escolhidos a partir do conjunto de fatores de escala fixa definidos a seguir: 1, 10, 30, 100, 300, 1000, 3000, 10000, 30000 ou 100000.

O tamanho do banco de dados é definido com referência ao fator de escala 1 (ou seja, SF = 1; aproximadamente 1 GB), que indica o tamanho mínimo necessário para um banco de dados de teste. Assim, respectivamente, os espaços de armazenamento aproximados dos fatores de escala anteriormente mencionados são 1 GB, 10 GB, 30 GB, 100 GB, 300 GB, 1000 GB, 3000 GB, 10.000 GB, 30000 GB ou 100000 GB.

Com ele é possível passar como parâmetro, chamado de fator de escala (Scale Factor) para geração dos dados que pode ser 1GB, 10GB, 30GB, 100GB, 300GB, 1000GB, 3000GB, 10000GB, 30000GB e 100000GB. A menor escala a ser definida para gerar uma base de dados para testes é de 1GB (TPC, 2018). O fator de escala escolhido para a geração do banco de dados, é multiplicado pelo tamanho de cada tabela, como se pode ver na tabela 1.

Tabela 1 - Tamanho Estimado Banco de Dados Fator Escala 10GB (em tuplas)

|  |  |  |
| --- | --- | --- |
| TABELA | TUPLAS | TUPLAS \* FS |
| SUPPLIER | 10000 | 100000 |
| PART | 200000 | 2000000 |
| PARTSUPP | 800000 | 8000000 |
| CUSTOMER | 150000 | 1500000 |
| ORDERS | 1500000 | 15000000 |
| LINEITEM | 6001215 | 59986052 |
| NATION | 25 | 25 |
| REGION | 5 | 5 |
| TOTAL | 8661245 | 86586132 |

Os valores do resultante do produto TUPLAS \* FS podem ser encontrados na parte superior de cada tabela, de acordo com a figura 1. – **fonte da tabela**

**Conclusão antes do término de cada capítulo**

1. PROCEDIMENTOS METODOLÓGICOS
   1. METODOLOGIA

A metodologia de elaboração deste trabalho está dividida nas seguintes etapas:

* Etapa 1: Utilização do Modelo Entidade Relacionamento (MER) indicado pelo padrão TPC-H;
* Etapa 2: Instalação e configuração dos bancos de dados;
* Etapa 3: Criação e população das tabelas;
* Etapa 4: Execução das consultas, e posteriormente, a medição do tempo correspondente ao banco de dados não sofreu nenhuma otimização.
* Etapa 5:
* Etapa 6: Coleta e análise de resultados.
  1. AMBIENTE DE TESTES

Para os testes foi utilizado o sistema com as seguintes características:

* OS: Debian GNU/Linux 10 (Buster) x86\_64;
* Kernel: 4.19.0-10-amd64;
* Packages: 1426 (dpkg);
* Shell: bash 5.0.3;
* DE: Xfce4;
* CPU: AMD Ryzen 3 2200G (4) @ 3.500GHz;
* GPU: AMD ATI Radeon Vega Series / Radeon Vega Mobile Seri;
* Memory: 1795MiB / 15026MiB;
* MySQL: 8.0.21 for Linux on x86\_64 (MySQL Community Server - GPL).
* Engine utilizado nas tabelas: InnoDB;
  1. COLETA DE DADOS

Para medição do tempo médio de cada consulta foi implementado um script (arquivo) .sql que automatiza a repetição de uma determinada consulta. Como retorno deste script se obtém o tempo acumulado das trinta consultas. Então, para se determinar o tempo de cada comando, deve-se subtrair o tempo atual com o valor da iteração anterior.

* 1. BASE DE DADOS NÃO OTIMIZADA

Do ponto de vista estatístico, quanto maior o número de testes mais próximo da realidade o benchmark se torna. Porém, este número de repetições (iterações) não pode ser alto, uma vez que comprometeria o prazo de entrega deste teste. Desta forma, foram realizadas 30 (trinta) repetições da mesma consulta objetivando obter o tempo médio de determinada consulta. Como exemplo, os valores de retorno da primeira consulta (Query 01) são apresentados na tabela abaixo:

Tabela 2 - Tempo Acumulado da Query 01

|  |  |  |  |
| --- | --- | --- | --- |
| Iteração | Tempo Acumulado (segundos) | Tempo Absoluto (segundos) | Tempo Absoluto (minutos) |
| 1 | 138,45 | 138,45 | 2,31 |
| 2 | 277,03 | 138,58 | 2,31 |
| 3 | 415,54 | 138,51 | 2,31 |
| 4 | 554,01 | 138,47 | 2,31 |
| 5 | 692,57 | 138,56 | 2,31 |
| 6 | 831,18 | 138,61 | 2,31 |
| 7 | 969,78 | 138,60 | 2,31 |
| 8 | 1.108,32 | 138,54 | 2,31 |
| 9 | 1.246,98 | 138,66 | 2,31 |
| 10 | 1.385,55 | 138,57 | 2,31 |
| 11 | 1.524,24 | 138,69 | 2,31 |
| 12 | 1.662,76 | 138,52 | 2,31 |
| 13 | 1.801,35 | 138,59 | 2,31 |
| 14 | 1.939,96 | 138,61 | 2,31 |
| 15 | 2.078,59 | 138,63 | 2,31 |
| 16 | 2.217,34 | 138,75 | 2,31 |
| 17 | 2.355,86 | 138,52 | 2,31 |
| 18 | 2.494,44 | 138,58 | 2,31 |
| 19 | 2.633,06 | 138,62 | 2,31 |
| 20 | 2.771,71 | 138,65 | 2,31 |
| 21 | 2.910,31 | 138,60 | 2,31 |
| 22 | 3.048,89 | 138,58 | 2,31 |
| 23 | 3.187,48 | 138,59 | 2,31 |
| 24 | 3.326,09 | 138,61 | 2,31 |
| 25 | 3.464,60 | 138,51 | 2,31 |
| 26 | 3.603,12 | 138,52 | 2,31 |
| 27 | 3.741,71 | 138,59 | 2,31 |
| 28 | 3.880,45 | 138,74 | 2,31 |
| 29 | 4.018,96 | 138,51 | 2,31 |
| 30 | 4.157,68 | 138,72 | 2,31 |

Os tempos gastos das consultas no banco de dados não otimizado são representados na tabela abaixo:

Tabela 3 - Tempo Médio das Consultas da Base de Dados não Otimizada

|  |  |  |  |
| --- | --- | --- | --- |
| Consulta | Média do Tempo Acumulado (segundos) | Média do Tempo Absoluto (s) | Média do Tempo Absoluto (minutos) |
| 1 | 2.147,93 | 138,59 | 2,31 |
| 2 | 2.145,27 | 138,41 | 2,31 |
| 3 | 1.002,43 | 57,55 | 0,96 |
| 4 | 247,00 | 12,93 | 0,21 |
| 5 | 624,00 | 37,32 | 0,62 |
| 6 | 341,48 | 21,37 | 0,35 |
| 7 | 362,16 | 21,02 | 0,35 |
| 8 | 359,68 | 21,85 | 0,36 |
| 9 | 4.375,18 | 269,60 | 4,49 |
| 10 | 359,68 | 21,85 | 0,36 |
| 11 | 75,88 | 4,91 | 0,08 |
| 12 | 599,28 | 34,83 | 0,18 |
| 13 | 2.433,88 | 157,14 | 2,62 |
| 14 | 521,64 | 31,47 | 0,18 |
| 15 | 717,24 | 46,29 | 0,15 |
| 16 | 89,39 | 5,78 | 0,10 |
| 17 | 180,83 | 11,87 | 0,14 |
| 18 | 550,82 | 33,10 | 0,55 |
| 19 | 66,86 | 3,85 | 0,07 |
| 20 | 201,54 | 11,12 | 0,19 |
| 21 | 1.257,41 | 78,84 | 1,31 |
| 22 | 37,01 | 2,16 | 0,03 |

1. ANÁLISE DOS DADOS DA PESQUISA (OBTIDOS)

1. LIMITAÇÕES DO TESTE

1. CONSIDERAÇÕES FINAIS
2. REFERÊNCIAS

ATANAZIO, J. **PostgreSQL - SQL Básico**. [S.l.]: [s.n.], 2019. Disponivel em: <https://github.com/juliano777/pgsql\_fs2w/blob/master/postgresql\_sql\_basico.pdf>. Acesso em: 10 Outubro 2020.

CABRAL, S.; MURPHY, K. **MySQL Administrator’s Bible**. Indianapolis: Wiley, 2009.

DAVID DUCOS. Percona. **InnoDB Buffer Pool Resizing:** Chunk Change, 2018. Disponivel em: <https://www.percona.com/blog/2018/06/19/chunk-change-innodb-buffer-pool-resizing/>. Acesso em: 08 Janeiro 2021.

DB-ENGINES. **DB-Engines Ranking**, 14 Junho 2020. Disponivel em: <https://db-engines.com/en/ranking>. Acesso em: 14 Junho 2020.

MYSQL. **MySQL 8.0 Reference Manual**, 09 Setembro 2020. Disponivel em: <https://downloads.mysql.com/docs/refman-8.0-en.pdf>. Acesso em: 2020.

RAMAKRISHNAN, R.; GEHRKE, J. **Sistemas de Gerenciamento Sistemas de Gerenciamento de Banco de Dados**. São Paulo: McGraw-Hill, 2009.

SCHWARTZ, B.; TKACHENKO, ; ZAITSEV,. **High Performance MySQL**. Third Edition. ed. Sebastopol: O’Reilly Media, 2012.

TARGETTRUST. **Performance e Otimização de Banco de Dados MySQL**, 2017. Disponivel em: <http://materiais.targettrust.com.br/ebook-otimizacao-banco-de-dados-mysql-lp>. Acesso em: 19 Setembro 2020.

TPC. **BENCHMARK (Decision Support) Standard Specification Revision**, 2018. Disponivel em: <http://www.tpc.org/tpc\_documents\_current\_versions/pdf/tpc-h\_v2.18.0.pdf>.

WIKIPEDIA. **David DeWitt**, 14 Julho 2020. Disponivel em: <https://en.wikipedia.org/wiki/David\_DeWitt>. Acesso em: 01 Setembro 2020.