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 measures performance when adjustments are made to the configuration variables of the MySQL Database system using the Debian Buster Linux Operating System. To perform this test, an international model for benchmark evaluation called TPC-H will be used, which serves to measure workload, helping to support the decision. For this, four databases will be created: The first two will be created following the model of creation and population of the tables indicated by TPC-H, these having the sizes of 1 GB and another of 10 GB. From this moment on, the 22 queries proposed by the Benchmark test will be carried out and the respective measurements of the average time of each consultation will be carried out. In the next step, changes will be made to the values of the MySQL configuration variables such as cache memory and buffer memory in order to make queries to the bank faster. From the collected data, it will be described if there was a gain or loss in the average time in the consultations in these supposedly optimized databases.

**Keywords:** MySQL, Performance, TPC-H.

# LISTA DE FIGURAS

# LISTA DE TABELAS

# 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

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.

## 1.1 OBJETIVOS

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

### 1.1.1 OBJETIVO GERAL

Mensurar a variação de desempenho após alterações dos valores das variáveis de configuração no banco de dados MySQL utilizando como referência de Benchmark o padrão TPC-H.

### 1.1.2 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 alterações nas variáveis de sistema no mesmo banco de dados;
- Analisar o tempo gasto nessas consultas depois das otimizações.

## 1.2 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.

Visando a otimização ao acesso dessa informação, os profissionais que trabalham com Banco de Dados procuram realizar configurações rápidas que já impactam positivamente no desempenho desta tecnologia, dentre elas, a alteração das variáveis do sistema como por exemplo, memória de acesso ao disco e memória compartilhada entre o sistema de Gerenciamento de Banco de Dados e o sistema operacional.

# 2. 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.

## 2.1 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.

## 2.2 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 (CABRAL e MURPHY, 2009):

- 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;

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.

### 2.2.1 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).

### 2.2.2 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).

### 2.2.3 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 (ATANAZIO, 2019). 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.

## 2.3 CHAVES

### 2.3.1 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).

### 2.3.2 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).

### 2.3.3 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).

## 2.4 VARIÁVEIS DE SISTEMA DO SERVIDOR MYSQL

O servidor MySQL mantém muitas variáveis de sistema que configuram sua operação. Cada variável do sistema possui um valor padrão. As variáveis do sistema podem ser definidas na inicialização do servidor usando opções na linha de comando ou em um arquivo de opções (MYSQL, 2020). A maioria deles pode ser alterada dinamicamente em tempo de execução

usando a instrução SET que permite modificar a operação do servidor sem precisar interrompê-lo e reiniciá-lo. Também é possível realizar a alteração dos valores dessas variáveis através ou de um arquivo de configuração no Debian localizado <**devo ou não colocar a localização**>. Porém neste último caso, é necessário reiniciar o servidor (serviço em execução do MySQL).

Como está havendo uma alteração que pode comprometer a integridade do Banco de Dados e/ou do sistema operacional, por motivos de segurança, para que sejam alterados esses valores é indispensável privilégios especiais ou de superusuário. Já para definir um valor de variável de tempo de execução do sistema de sessão normalmente não requer privilégios especiais e pode ser feito por qualquer usuário, embora haja exceções (MYSQL, 2020).

### 2.4.1 INNODB_BUFFER_POOL_SIZE

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 (SCALEGRID, 2018).

### 2.4.2 INNODB_BUFFER_POOL_INSTANCES

Essa variável representa 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 (MYSQL, 2020).

### 2.4.3 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 bloco 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:

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:

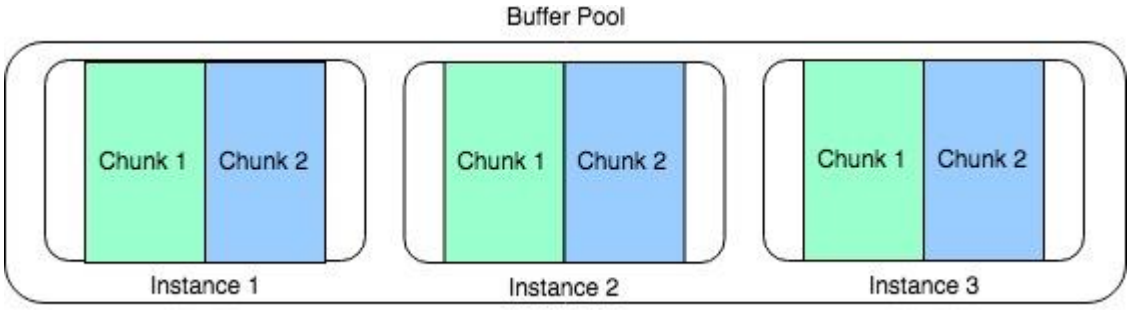


*Figura 1 - Modelo simplificado Memória Buffer*

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.

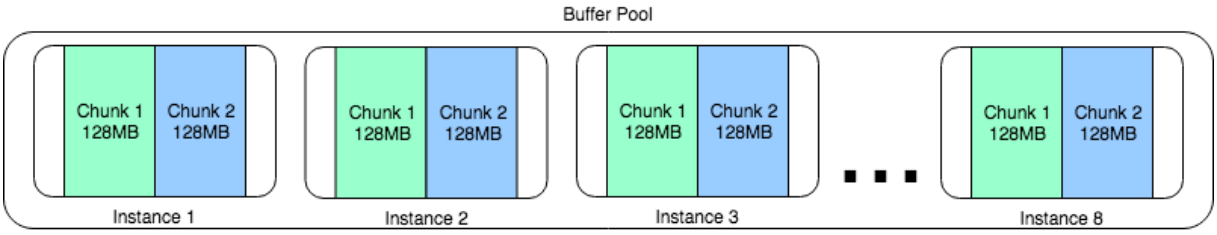


*Figura 2 - Modelo detalhado da Memória Buffer*

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, que acarreta, de acordo com o exemplo, 16 blocos de memória.

Dentre os vários benefícios de ter várias instâncias são (DAVID DUCOS, 2018):

- Em um servidor virtual, é possível adicionar memória dinamicamente;
- Para um servidor físico, você pode querer reduzir o uso de memória do banco de dados para abrir caminho para outros processos;

### 2.4.4 THREAD_CACHE_SIZE

Esta variável dimensiona quantos threads o servidor deve armazenar em cache para reutilização de comandos. Ela 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.

## 2.5 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.

### 2.5.1 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 **()** abaixo (TPC, 2018).

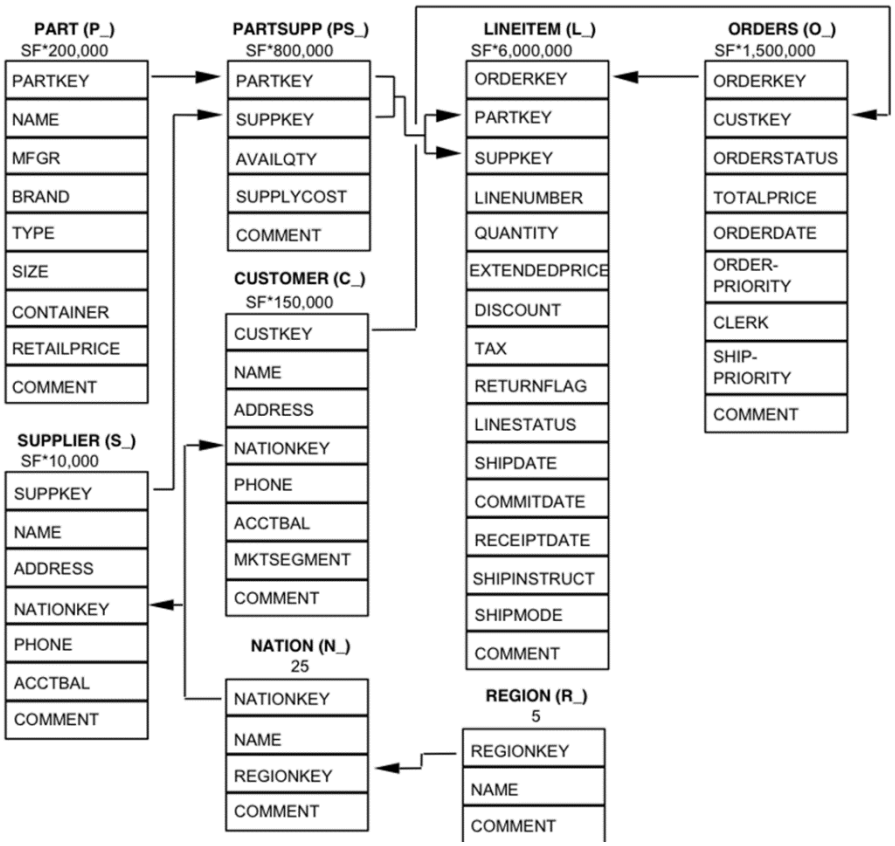


*Figura 3 - Esquema TPC-H – fonte da imagem*

### 2.5.2 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 3 (TPC, 2018).

Neste momento, com a base totalmente criada e populada, já é possível a realização de medição dos tempos das consultas.

# 3. PROCEDIMENTOS METODOLÓGICOS

## 3.1 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 utilizando as configurações que vêm por padrão no MySQL depois de instalado;
- Etapa 5: Alterada os valores das variáveis da Pool;
- Etapa 6: Execução novamente das consultas propostas pelo benchmark TPC-H;
- Etapa 7: Coleta e análise de resultados.

## 3.2 AMBIENTE DE TESTES

Para a realização dos testes foi utilizado o sistema com as seguintes características no que tange aos softwares utilizados:

- OS: Debian GNU/Linux 10 (Buster) x86_64;
- Kernel: 4.19.0-10-amd64;
- MySQL: 8.0.21 for Linux on x86_64 (MySQL Community Server - GPL);
- Packages: 1426 (dpkg);
- Shell: bash 5.0.3;
- DE: XFCE4;

Já no que se refere ao hardware, o teste foi executado nas seguintes configurações:

- CPU: AMD Ryzen 3 2200G (4) @ 3.500GHz;
- GPU: AMD ATI Radeon Vega Series / Radeon Vega Mobile Seri;
- Memória do Sistema: 15026MiB;
- Engine utilizado nas tabelas: InnoDB;

## 3.3 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.

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. O resultado ao final da execução do script, é obtido o resultado dos tempos acumulados da consultada executada. Assim, para se determinar o tempo de cada consulta, deve-se subtrair o tempo atual com o valor da iteração anterior.

### 3.3.1 BASE DE DADOS NÃO OTIMIZADA DE 10 GB

Como exemplo, os valores de retorno da primeira consulta (Query 01) quando executadas na base de dados de tamanho de 10 GB 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 |

### 3.3.2 ANÁLISE DOS TEMPOS DA BASE DE DADOS NÃO OTIMIZADA DE 10 GB

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 |

### 3.3.3 MÉDIAS DE TODAS AS BASES DE DADOS

Depois de todos os testes, os valores obtidos estão representados na tabela abaixo:

*Tabela 4 - Média de todas as bases de dados*

| | 1GB | | | 10GB | | |
|---|---|---|---|---|---|---|
| Consulta | Media do Tempo (segundos) – Sem otimização | Media do Tempo (segundos) – Depois da Otimização | Percentual de Ganho ou perda | Media do Tempo (segundos) – Sem otimização | Media do Tempo (segundos) – Depois da Otimização | Percentual de Ganho ou perda |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 13,83 | 13,5657 | -1,91% | 138,5893 | 135 | -2,59% |
| 2 | 0,106 | 0,105 | -0,94% | 138,4093 | 1,368 | -99,01% |
| 3 | 4,4657 | 1,1135 | -75,06% | 57,549 | 89,5033 | 55,53% |
| 4 | 0,58 | 0,5827 | 0,46% | 12,9313 | 7,305 | -43,51% |
| 5 | 2,7257 | 0,8626 | -68,35% | 37,322 | 11,4407 | -69,35% |
| 6 | 2,08 | 1,7807 | -14,39% | 21,37 | 20,2969 | -5,02% |
| 7 | 1,7817 | 0,7714 | -56,70% | 21,0213 | 9,077 | -56,82% |
| 8 | 6,5713 | 2,3937 | -63,57% | 21,8473 | 30,7483 | 40,74% |
| 9 | 7,5917 | 1,6473 | -78,30% | 269,6007 | 377,461 | 40,01% |
| 10 | 1,762 | 1,152 | -34,62% | 21,8473 | 14,1997 | -35,01% |
| 11 | 0,4513 | 0,1237 | -72,60% | 4,9117 | 1,9287 | -60,73% |
| 12 | 3,015 | 2,5993 | -13,79% | 34,8333 | 132,0593 | 279,12% |
| 13 | 1,515 | 2,8777 | 89,94% | 34,8333 | 35,5354 | 2,02% |
| 14 | 1,5233 | 1,7743 | 16,48% | 31,4677 | 105,8877 | 236,50% |
| 15 | 4,4237 | 3,7787 | -14,58% | 46,294 | 39,0483 | -15,65% |

| | | | | | | |
|---|---|---|---|---|---|---|
| 16 | 0,5923 | 0,315 | -46,82% | 5,7837 | 3,6267 | -37,29% |
| 17 | 1,2233 | 0,266 | -78,26% | 11,867 | 2,3441 | -80,25% |
| 18 | 3,009 | 2,6623 | -11,52% | 33,1013 | 129,46 | 291,10% |
| 19 | 0,3039 | 0,172 | -43,40% | 3,8523 | 2,284 | -40,71% |
| 20 | 0,81 | 0,2207 | -72,76% | 11,1193 | 6,9877 | -37,16% |
| 21 | 0,81 | 5,199 | 541,85% | 78,8357 | 151,268 | 91,88% |
| 22 | 0,2087 | 0,1633 | -21,73% | 2,162 | 1,76 | -18,59% |

# 4. ANÁLISE DOS DADOS DA PESQUISA (OBTIDOS)

Com base nos valores médios dos tempos de consultas, pode-se observar que uma base de dados menor pode sofrer uma maior interferência quando ajustados os valores das variáveis deste estudo. Mesmo os valores de tempo médio de algumas consultas ter aumentado, na grande maioria dos valores obtidos se mostraram com ganhos no tempo de consulta.

# 5. CONSIDERAÇÕES FINAIS

O estudo se tornou bastante frutífero, pois foi capaz explicitar que não se torna determinante o ganho de performance nas consultas apenas aumentando a memória disponível para o SGBD.

# 6. 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.

SCALEGRID. **What is an InnoDB Buffer Pool?**, 2018. Disponivel em: <https://scalegrid.io/blog/calculating-innodb-buffer-pool-size-for-your-mysql-server/>. Acesso em: 08 Janeiro 2020.

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.