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**

**MEDIÇÃO DO GANHO DE DESEMPENHO APÓS OTIMIZAÇÕES NO BANCO DE DADOS MYSQL**

PALMARES, 2021

**ALEXSANDRO MATIAS DE ALMEIDA**

**MEDIÇÃO DO GANHO 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, irmã 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 analisa o quanto de desempenho é ganho quando realizados diversos ajustes de configuração no Banco de Dados MySQL, processo esse chamado de tuning. Para realização deste teste de performance será utilizado um modelo internacional para benchmark chamado TPC-H que serve para medição de carga de trabalho auxiliando no suporte à decisão. Para isso, serão criadas duas bases dados: A primeira apenas seguindo o modelo de criação e população das tabelas indicado pelo TPC-H. Já no segundo serão realizadas as configurações possíveis para as consultas ao banco sejam as mais performáticas possíveis. A partir dos dados coletados, serão descritos os valores dos ganhos percentuais nas consultas a essa base de dados otimizada.

**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. For that, two databases will be created: The first just following the model of creation and population of the tables indicated by TPC-H. In the second one, the possible configurations for queries to the bank will be made as well as possible. From the collected data, the values of the percentage gains in the consultations to this optimized database will be described

**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.

MER – Modelo Entidade Relacionamento

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 relacionais 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.

* 1. OBJETIVOS

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

* + 1. OBJETIVO GERAL

Mensurar o ganho de desempenho após otimizações 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;
* Recolher 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 o conceito de Big Data que vem se consolidando como a base para o desenvolvimento de novas tecnologias capazes de manipular grandes conjuntos de dados visando também promover subsídios para a tomada de decisão.

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. 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. Os conceitos em torno dos testes necessários para avaliação de desempenho dos dois Banco de Dados em estudo e testes estatísticos utilizados na análise dos resultados.

* 1. DADOS
  2. INFORMAÇÃO
  3. BANCO DE DADOS
  4. 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 software projetado cuja função é gerenciar uma base dados. As vantagens do seu uso são (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 & 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 determinados 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, 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.

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 pois existe drivers ODBC, JDBC e .NET 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, ou em tradução livre, Linguagem Estruturada de Consultas, que representa a linguagem usada nos SGBDs por padrão. No entanto, cada um tem suas particularidades dentro da própria linguagem, tendo implementações diferentes (Atanazio, 2019). 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 (restrição de chave primária) é uma ou mais colunas que identificam exclusivamente uma linha. Nenhuma das colunas que fazem parte da chave primária pode ser anulável. Uma tabela não deve ter mais do que uma chave primária (TPC, 2018).

* + 1. CHAVE ESTRANGEIRA

Uma chave estrangeira (restrição de chave estrangeira) é uma coluna ou combinação de colunas usada para estabelecer e impor um link entre os dados em duas tabelas. Um link é criado entre duas tabelas 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. ÍNDICES

Os índices são usados ​​para localizar linhas com valores de colunas específicas mais rapidamente. Sem um índice, o MySQL deve começar com a primeira linha e, em seguida, leia toda a tabela para encontrar o registro procurado (MySQL, 2020). Quanto maior a tabela, o mais isso custa. Se a tabela tiver um índice para as colunas em questão, o MySQL pode determinar rapidamente a posição para procurar no meio do arquivo de dados sem ter que olhar para todos os dados. Isso é muito mais rápido do que ler cada linha sequencialmente

A documentação oficial da linguagem (MySQL, 2020) indica para que haja um melhor desempenho do banco, os parâmetros que devem ser observados são:

* As consultas que contém a cláusula WHERE;
* Quando for utilizadas as funções MIN() ou MAX(), valores mínimo e máximo respectivamente, para uma coluna indexada específica key\_col. Isso é otimizado por um pré-processador que verifica se você está usando WHERE key\_part\_N = constante em todas as partes-chave que ocorrem antes de key\_col no índice. Neste caso, o MySQL faz uma única pesquisa de chave para cada MIN () ou MAX () expressão e a substitui por uma constante. Se todas as expressões forem substituídas por constantes, a consulta retorna de uma vez. Por exemplo: SELECT MIN (key\_part2), MAX (key\_part2) FROM nome\_tabela WHERE parte\_chave1 = 10;
* Aonde for usado alguma classificação ou agrupamento na tabela, ou seja, utilizando as funções ORDER BY e GROUP BY respectivamente
* Para eliminar linhas de consideração. Se houver uma escolha entre vários índices, o MySQL normalmente usa o índice que encontra o menor número de linhas (o índice mais seletivo).
* Se a tabela tiver um índice de várias colunas, qualquer prefixo mais à esquerda do índice pode ser usado pelo otimizador

Quando se deseja uma melhor performance para se encontrar linhas de outras tabelas ao realizar junções. MySQL pode usar índices em colunas mais de forma eficiente se forem declarados como do mesmo tipo e tamanho. Neste contexto, VARCHAR e CHAR são considerados iguais se forem declarados do mesmo tamanho. Por exemplo, VARCHAR (10) e CHAR (10) são do mesmo tamanho, mas VARCHAR (10) e CHAR (15) não são (MySQL, 2020).

Para comparações entre colunas de string não binárias, ambas as colunas devem usar o mesmo conjunto de caracteres. Por exemplo, comparar uma coluna utf8 com uma coluna latin1 impede o uso de um índice.

Seção 8.3.13, “Índices descendentes”.

• Em alguns casos, uma consulta pode ser otimizada para recuperar valores sem consultar as linhas de dados. (Um índice que fornece todos os resultados necessários para uma consulta é chamado de índice de cobertura.) Se uma consulta usa de uma tabela apenas colunas que estão incluídas em algum índice, os valores selecionados podem ser recuperados da árvore de índice para maior velocidade.

Os índices são menos importantes para consultas em tabelas pequenas ou grandes tabelas onde as consultas de relatório são mais processadas ou todas as linhas. Quando uma consulta precisa acessar a maioria das linhas, ler sequencialmente é mais rápido do que trabalhar por meio de um índice. As leituras sequenciais minimizam as buscas no disco, mesmo que nem todas as linhas sejam necessárias para a consulta.

* + 1. PARTICIONAMENTO DE TABELAS
  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;

A métrica de desempenho relatada pelo TPC-H é chamada de Métrica de Desempenho de Consulta por Hora Composto TPC-H (QphH @ Size) e reflete vários aspectos da capacidade do sistema de processar consultas. Esses aspectos incluem o tamanho do banco de dados selecionado em relação ao qual as consultas são executadas, o poder de processamento da consulta quando as consultas são enviadas por um único fluxo e o rendimento da consulta quando as consultas são enviadas por vários usuários simultâneos. A métrica Preço / Desempenho TPC-H é expressa como $ / QphH @ Size. Para estar em conformidade com o padrão TPC-H, todas as referências aos resultados TPC-H para uma determinada configuração devem incluir todos os componentes de relatório necessários (ver Cláusula

5.4.6). 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 o seus relacionamentos são ilustrados na figura abaixo (TPC, 2018).

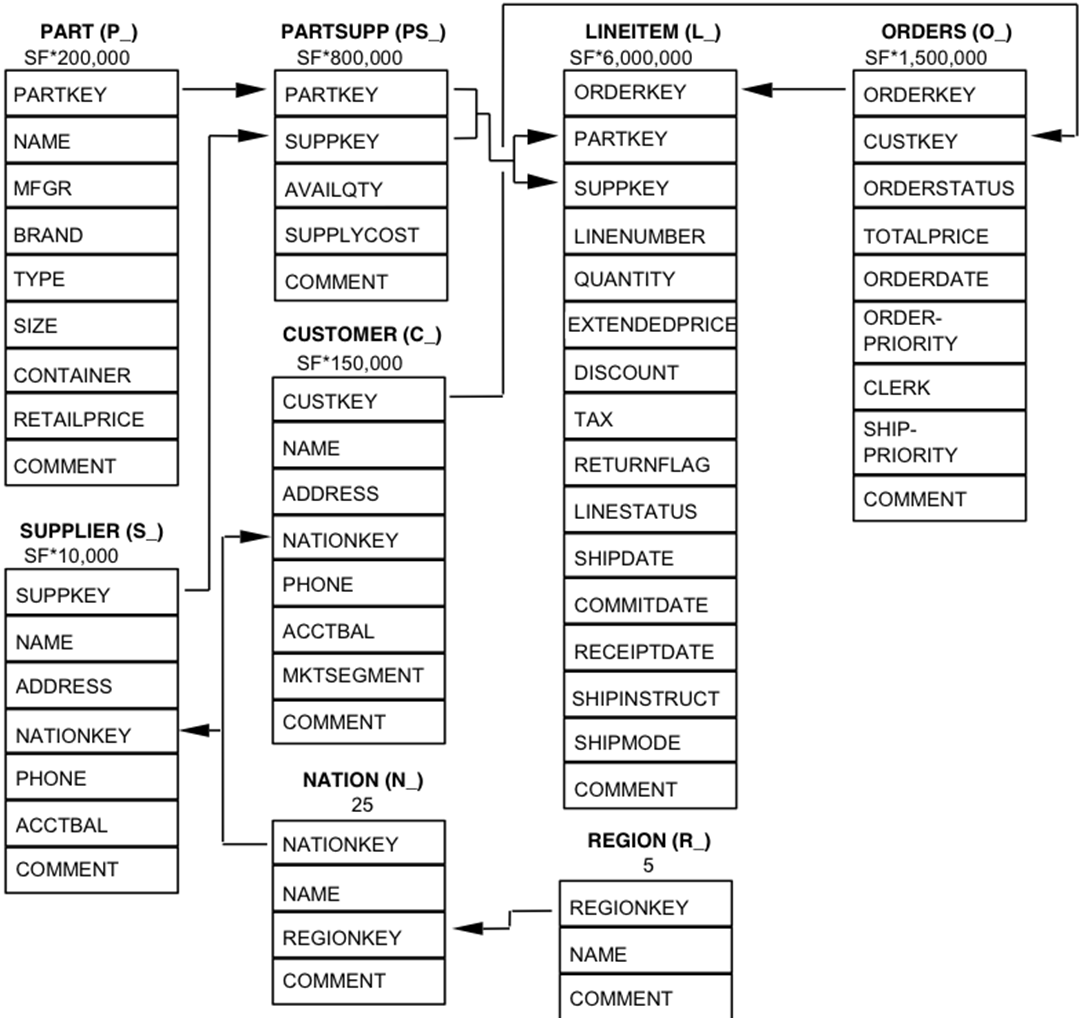


Figura 1 - Esquema TPC-H

* + 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.

Comentário 1: Embora o tamanho mínimo do banco de dados de teste para um teste de desempenho válido seja de 1 GB (ou seja, SF = 1), um banco de dados de teste de 3 GB (ou seja, SF = 3) não é permitido. Este requisito tem o objetivo de encorajar a comparabilidade dos resultados na extremidade inferior e garantir uma diferença real substancial nos tamanhos do banco de dados de teste.

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 Figura 1.

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 1 - 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 2 - 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)
2. LIMITAÇÕES DO TESTE
3. CONSIDERAÇÕES FINAIS
4. REFERÊNCIAS

Cabral, S., & Murphy, K. (2009). *MySQL Administrator’s Bible.* Indianapolis, Indiana: Wiley.

DB-Engines. (14 de Junho de 2020). Acesso em 14 de Junho de 2020, disponível em DB-Engines Ranking: https://db-engines.com/en/ranking

MySQL. (09 de Setembro de 2020). Acesso em 2020, disponível em MySQL 8.0 Reference Manual: https://downloads.mysql.com/docs/refman-8.0-en.pdf

Ramakrishnan, R., & Gehrke, J. (2009). *Sistemas de Gerenciamento Sistemas de Gerenciamento de Banco de Dados.* São Paulo: McGraw-Hill.

Schwartz, B., Tkachenko, V., & Zaitsev, P. (2012). *High Performance MySQL* (Third Edition ed.). Sebastopol: O’Reilly Media.

Targettrust. (2017). Acesso em 19 de Setembro de 2020, disponível em Performance e Otimização de Banco de Dados MySQL: http://materiais.targettrust.com.br/ebook-otimizacao-banco-de-dados-mysql-lp

TPC. (2018). Fonte: BENCHMARK (Decision Support) Standard Specification Revision: http://www.tpc.org/tpc\_documents\_current\_versions/pdf/tpc-h\_v2.18.0.pdf

Wikipedia. (14 de Julho de 2020). Acesso em 01 de Setembro de 2020, disponível em David DeWitt: https://en.wikipedia.org/wiki/David\_DeWitt