PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

BOAS PRÁTICAS DE TUNING DE BANCO DE DADOS PARA OTIMIZAÇÃO DE UM BANCO DE DADOS RELACIONAL

GABRIEL TRISTAO DO COUTO MENDONÇA

GOIÂNIA
2021

GABRIEL TRISTÃO DO COUTO MENDONÇA

BOAS PRÁTICAS DE TUNING DE BANCO DE DADOS PARA OTIMIZAÇÃO DE UM BANCO DE DADOS RELACIONAL

Trabalho de Conclusão de Curso apresentado à Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.
Orientador(a):
Prof. Dr. Fábio Barbosa Rodrigues
Banca examinadora:

Prof. Me. Eugenio Júlio Messala de Caravalho
Prof. Me. Olegário Correa da Silva Neto

GOIÂNIA
2021

GABRIEL TRISTÃO DO COUTO MENDONÇA

BOAS PRÁTICAS DE TUNING DE BANCO DE DADOS PARA OTIMIZAÇÃO DE UM BANCO DE DADOS RELACIONAL

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Engenharia de Computação, em _____/_____/_____.

_____

Orientador(a): Dr. Fábio Barbosa Rodrigues

_____

Prof. Me. Nome do coordenador(a) de TCC
Coordenador(a) de Trabalho de Conclusão de Curso

GOIÂNIA
2021

DEDICATÓRIA

Dedico este trabalho a todos que contribuíram para meu crescimento intelectual e profissional, no decorrer desta jornada.

AGRADECIMENTOS

Agradeço a Deus primeiramente, pela minha saúde, que possibilitou continuar com meus estudos, logo depois aos meus Pais que me deram estrutura e oportunidade para chegar a onde estou hoje e por tudo que alcancei até agora no decorrer deste caminho.

“A verdadeira função do homem é viver, não existir. Eu não gastarei meus dias a tentar prolonga-los. Usarei o meu tempo.”

Jack London

Sumário

RESUMO

Com o avanço da tecnologia, entramos em uma era com um grande acumulo de dados, tais dados possuem diversas aplicações gerando seus próprios valores, para o controle e gerenciamento dos mesmos, contamos com estruturas e ferramentas, para disponibilizar a maior quantidade de dados no menor tempo possível, uma vez que há limites para determinadas ferramentas garantirem certo nível de desempenho, cabe a otimização manual continuar este processo, através de boas práticas bem definidas para melhorar cada vez mais o desempenho de um banco de dados relacional.

Tais boas práticas, são abordadas desde a construção de consultas em sql até no uso mais apropriado de meios de armazenamento, deixando de forma clara, como uso de boas práticas contribuem de forma significativa na otimização de um banco de dados relacional.

ABSTRACT

With the advancement of technology, we have entered an era with a large accumulation of data, such data have several applications generating their own values, for their control and management, we have structures and tools to make the greatest amount of data available in the smallest amount. possible time, since there are limits for certain tools to guarantee a certain level of performance, it is up to manual optimization to continue this process, through well-defined best practices to increasingly improve the performance of a relational database.

Such best practices are addressed from the construction of queries in sql to the most appropriate use of storage media, making it clear how the use of best practices significantly contributes to the optimization of a relational database.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

BD Banco de Dados

DCL *Data Control Language - Linguagem de Controle de Dados*

DDL *Data Definition Language – Linguagem de Definição de Dados*

DML *Data Manipulation Language – Linguagem de Manipulação de Dados*

DTL *Data Transaction Language – Linguagem de Transação de Dados*

ERP *Enterprise Resource Planning – Planejamento de Recursos Especiais*

HD *Hard disk – Disco Rígido*

SGBD Sistema Gerenciador de Banco de Dados

SQL *Structured Query Language – Linguagem de Consulta Estruturada*

SSD *Solid State Drive – Unidade de Armazenamento de Estado Solido*

TI Tecnologia da informação

MER Modelo Entidade Relacionamento

1 INTRODUÇÃO

Atualmente, com a constante informatização de dados realizadas por diversos setores no mundo, surge a necessidade de um armazenamento seguro, inteligente e bem estruturado, fazendo com que as empresas passem a investir nos sistemas gerenciadores de banco de dados (SGBD). (SEM AUTOR,2020) (SCUDERO,2017).

O SGBD atua como uma ponte entre o usuário e o banco de dados, facilitando a manutenção de pontos cruciais para uma boa segurança e fluxo de informação, nele podemos manipular os dados a um nível alto de abstração, facilitando a extração e manipulação de dados importantes, como por exemplo os dados que são usados para o suporte a tomada de decisão (OLIVEIRA, 2019).

Alguns dos SGBDs mais conhecidos: Oracle, MySQL, PostgreSQL, SQL Server. (SEM AUTOR,2020; SCUDERO,2017)

Com um banco de dados uma vez consolidado, devemos nos atentar a sua possibilidade de sofrer algumas que alterações inevitáveis, como no seu espaço de armazenamento, na velocidade atualização de informações e por fim nos novos tipos de dados quem podem ser armazenados. (OLIVEIRA, 2021)

Tais alterações mencionadas como volume, velocidade e variedade, podem ser resumidas ao termo “Big Data”, do qual indica de forma objetiva quais são os aspectos mais cruciais que devemos tratar em um banco de dados, para que ele possa fornecer um bom desempenho.

Atualmente há bons SGBDs que conseguem manter um banco de dados de forma automática, com um bom desempenho, porém sempre há limitações, justamente por nem todas as mudanças poderem ser previsíveis, podendo comprometer o desempenho do banco. (BRITO; LIFSCHITZ, 2018)

Normalmente os grandes causadores de perca de desempenho nos bancos de dados, estão ligados de 70% a 80% nos scripts/consultas, feitos de forma errada ou ineficiente, deixando o restante das causas para o hardware, SGBD’S e sistema operacional.

Segundo as pesquisas “Instruções SQL e índices são responsáveis por 60% a 90% dos problemas de desempenho de aplicações” (LECCO, 2003, apud

ANDRADE, 2005, p. 13); “Atividades relacionadas às consultas SQL consomem entre 70% e 90% dos recursos dos BDs” (LECCO, 2003, apud ANDRADE,2005, p. 14);

Dentro de tal contexto, entra a prática conhecida como *tuning* de banco de dados ou sintonia fina, do qual tal prática se caracteriza por ser um processo bem mais complexo e minucioso, levando a cada caso a ser tratado de forma personalizada, esta prática procura otimizar o desempenho de um banco de dados através de ajustes da infraestrutura até a suas consultas/scripts.

A sintonia fina ou *tuning* de banco de dados, pode proporcionar uma melhor vazão de dados, reduzindo gargalos no tempo de resposta e aumentando a capacidade com que o banco de dados possa executar determinados processos.

Fica evidente a importância de não somente ter um banco de dados ligado a um SGBD, mas sim de mantê-lo de organizado e bem implementado, para que o banco de dados e o SGBD trabalhem em sintonia, para proporcionar uma vazão de dados satisfatória.

Uma das áreas mais beneficiadas de um banco de dados bem otimizado, é setor de ciência de dados ligado ao suporte de decisões, podemos ver esse reflexo através do maior número de informações atualizadas e organizadas no menor tempo possível, do qual quanto menor o período, maior pode ser o ganho de uma empresa perante a concorrência de mercado. (DOMINICO, 2013) (RIGO, 2012)

Com a ciência de dados juntamente aos SGBDs, os investimentos na área de tecnologia da informação nas empresas, vem crescendo de forma considerável, segundo a pesquisa de mercado brasileiro, realizado pelo Centro de tecnologia de informação aplicada da escola de Administração de empresas de São Paulo da fundação Getúlio Vargas (FGV-EAESP), foi constatado que o investimento e gastos no setor de TI nas empresas foi de 8%, valor similar que o setor tem no PIB brasileiro. (Prof. Fernando S. Meirelles,FGV, 2020)

Fica claro, que hoje em dia o sucesso e futuro de uma empresa pode ser fortemente alavancado pelos próprios dados gerados, assim como os produzidos por sua concorrência, basta apenas ela realizar um uso correto destes dados em um tempo hábil. (DOMINICO, 2013)

Não somente ligado a área de ciência de dados, mas também o bom desempenho de um banco de dados, pode acabar refletindo no desempenho de outros sistemas, como os famosos softwares de ERP (*Enterprise Resource Planning*

traduzindo como planejamento de recursos de empresa), presentes em quase todas as empresas, estes softwares estão ligados diretamente a sua gestão englobando desde a um departamento de recursos humanos até ao controle de diferentes departamentos. (SCUDERO,2017)

Grande parte destas aplicações, podem ter seu nível de desempenho diretamente ligado ao banco de dados, tendo em vista ao grande número de consultas e requisições que podem ser realizadas por estes sistemas, sendo assim uma vez que o banco passa a entregar mais informações em um menor período, o software de ERP, pode passar a entregar resultados em seu funcionamento de forma mais rápida.

A sintonia fina de um banco de dados ou *tuning*, muitas vezes, pode acabar sendo bem complexa, dependendo do ambiente e da situação em que um banco de dados é diagnosticado, dentro de tal cenário, não podemos aplicar uma fórmula exata de implementação para a otimização de seu desempenho, porém há práticas valiosas que se bem aplicadas em determinadas situações, podem facilitar a implementação do *tuning*.

Dentro do contexto de *tuning* de banco de dados abordado neste trabalho, o mesmo irá se restringir à construção de scripts e à parte de hardware abordada de forma um pouco mais superficial, lembrando que o tuning de banco de dados engloba diversos aspectos mais profundos em sua composição.

O objetivo do trabalho é apresentar, um estudo sobre as principais boas práticas para implementação de *tuning* de banco de dados, procurando destacar, os ganhos de desempenho e o aumento da eficiência.

Não será somente apresentado a implementação de boas práticas de *tuning* de banco de dados e seus resultados, mas também será apresentado as vantagens, desvantagens e situações mais indicadas que cada boa prática possui.

O estudo será conduzido através da implementação de boas práticas, que serão abordadas no capitulo 4, em um banco de dados teste, onde cada boa prática será avaliada por seu comportamento e tempo de execução, sendo evidenciado seu desempenho pela comparação de tempo de execução de consultas ineficientes para a diferença de consultas que seguem boas práticas.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 banco de dados / banco de dados relacionais

Podemos definir banco de dados como um conjunto de vários pacotes de dados que possuem relação entre si, a fim de gerar um objetivo dentro do contexto em que é aplicado, normalmente se dá por uma composição bem-organizada dos dados, o que possibilita extração e manipulação de informações.

Dentro da composição de um banco de dados relacional, temos os metadados, os metadados são dados que contém informações complementares de outros dados, com o intuito de organizar, facilitar e simplificar os relacionamentos armazenados além de evidenciar os objetivos de cada informação.

O banco de dados relacional pode ser composto por coleções de tabelas, contendo diferentes conjuntos de dados relacionados a determinados objetos (SILVA, D, 2015; SCUDERO, 2017; REIS, F, 2019).

2.2 Definição de SGBD (Sistema Gerenciador de Banco de Dados)

Com um banco de dados, surge a necessidade de mantê-lo organizado, seguro e integro, aí entra o Sistema de Gerenciamento de banco de Dados (SGBD), que auxilia de forma efetiva e solida neste processo, atualmente existe vários tipos de SGBDs cada um com suas particularidades, que faz com que cada um tenha uma vantagem em determinada situação.

O SGBD atua como uma ponte entre o usuário e o banco de dados, diminuindo o nível de abstração para a operação de um banco de dados, é o SGBD que recebe e processa todas as solicitações feitas por aplicações ou pelo seu operador, que as interpreta em várias operações complexas para cumprimento destas requisições.

Com um sistema de gerenciamento de banco de dados, podemos obter controle sobre acesso, backups, scripts e outras funções cruciais para um bom funcionamento de um banco. (SCUDERO, 2017; SILVA, D, 2015)

Um SGBD deve possuir as seguintes características:

- Controle de Redundâncias;
- Compartilhamento de Dados;
- Controle de Acesso;
- Interfaceamento - Disponibilizar versões gráficas e não somente modo texto;
- Esquematização - Tornar compreensível as relações entre tabelas;
- Controle de Integridade;
- Cópias de Segurança

2.3 Linguagem SQL

Structured Query Language (SQL), em sua tradução para o português como linguagem estruturada de dados, é uma linguagem de programação que foi criada para a manipulação de dados dentro dos sistemas de gerenciamento de banco de dados. (OLIVEIRA, M, 2021;SANCHES, A, 2005)

Dentro da estrutura da linguagem SQL, temos as seguintes divisões de comandos:

DML – *Data Manipulation Language*: comandos usados para manipulação, alteração e atualização de dados.

DDL – *Data Definition Language*: comandos usados para o gerenciamento e estruturação de tabelas

DCL – *Data Control Language*: comandos usados para controle de permissões ao banco de dados.

DTL – *Data Transaction Language*: comandos usados para salvar alterações feitas.

2.4 Tabela Verdade

Tabela verdade é uma ferramenta usada no campo de estudo sobre a logica, onde em sua maior parte, envolve a análise de proposições e seus conectivos lógicos. (GOUVEIA, 2021.)

Na tabela verdade temos os seguintes conectivos lógicos:

NÃO(NOT) :Valor é falso quando a proposição for verdadeira e vice-versa.

E(AND): Valor é verdadeiro somente quando todas as preposições forem verdadeiras.

OU(OR):Para o valor ser verdadeiro, basta que ao menos uma proposição seja verdadeira.

SE(IF): Valor será falso quando a proposição antecedente for verdadeira e a em seguida for falsa.

2.5 Big Data

“Big data é definido por três V’s: Volume, Variedade e Velocidade” (Berman, 2013), o volume refere se a quantidade de dados, a variedade faz referência a diversidade de origens e tipos de onde um dado pode ser obtido e por fim, a velocidade é relacionada ao seu tempo de atualização, do qual um dado pode estar sob uma constante mudança através de dados externos e seus metadados (BERMAN, 2013).

2.6 Tuning de bancos de dados

A prática de *tuning* de banco de dados ou sintonia fina, surgiu da necessidade de diminuir o tempo de resposta dos bancos de dados, consequentemente melhorando a sua vazão de informações (*throughput*) e conseguindo encaixar o maior número de consultas executada em no menor tempo de execução possível, porém vale lembrar que existem diferenças entre os processos de otimização e sintonia fina no escopo de um SGBD.

A otimização de um banco de dados, normalmente ocorre dentro do SGBD de forma automática, onde ele converte comandos declarativos, normalmente em SQL, e os processa para uma linha de execução, especificando os operadores relacionados aos dados, ordenando e estruturando, por fim gerando uma otimização feita de forma automática.

A sintonia fina é uma boa prática que reúne uma série de procedimentos que podem ser implementados para a otimização dos bancos de dados, sendo um processo mais complexo e minucioso do qual nunca é estático, sendo sempre moldado de acordo com cada ambiente e situação, esta sintonia fina pode englobar quase todas as estruturas de um bando de dados, desde o hardware até as instruções SQL. (GOES, 2019; BRITO; LIFSCHITZ, 2018)

2.7 Benchmarks

Benchmark é um dos artefatos muito usados no estudo de concorrências, no qual por meio de um software, são aplicadas cargas de trabalho especificas, a fim de monitorar e estudar seu desempenho e comportamento durante as determinadas situações processando as cargas de trabalho.

A prática do benchmark é aplicada a diversas áreas, sendo uma delas para a avaliação e estudo dos SGBDs, nele são avaliados diversos parâmetros e comportamentos, sempre gerando e organizando seu histórico, para futuras comparações.

Através de um benchmark aplicado, podemos diagnosticar, testar e aprimorar um banco de dados, sendo um dos principais aliados da prática sintonia fina de banco de dados ou *tuning* de banco de dados. (BRITO; LIFSCHITZ, 2018)

2.8 HD e SSD

Dentro da parte de infraestrutura de banco de dados, temos várias opções de armazenamento, sendo duas mais usadas, são o HD e o SSD.

O HD (hard disk, traduzido disco rígido), é um disco mecânico para o armazenamento de dados, embora seja uma tecnologia antiga, ainda é muito usado, devido ao seu baixo custo e diversas opções de tamanho para armazenamento, porém por mais que que seja acessível , o mesmo apresenta um desempenho inferior quando se trata de taxa de transferência de dados, isto ocorre devido ao seu sistema de armazenamento, que não é de acesso direto, mas sim por blocos dentro do seu disco mecânico, do qual para acessar determinada informação em um bloco especifico, é necessário caminhar por todos seus antecessores.

O SSD (Solid State Drive, traduzindo unidade de armazenamento solido) é um dispositivo de armazenamento de acesso direto, bem semelhando aos cartões de memória, sendo assim dispensando o uso de qualquer componente mecânico, o que faz com que ele seja mais seguro, rápido e duradouro, todavia ainda o seu custo é bem maior do que é um HD, ainda mais quando tratamos de um SSD desenvolvido para ambiente de produção, onde o mesmo deve possuir uma vida útil maior em relação aos modelos domésticos. (LEANDRO, L, 2020)

2.9 Acesso de Dados SQL

Normalmente a linguagem SQL (*Structured Query Language*) apresenta dois padrões para acesso de dados, o primeiro e padrão é o *scan*, do qual começa do início da página até ao seu fim, de acordo com a query implementada.

O segundo tipo de acesso que a linguagem SQL pode realizar, são os acessos através de indicies, do qual pode se dizer que um índice é um campo ou um conjunto de campos pré-ordenados pelo banco de dados, referenciando cada informação contida nele.

Quando um índice é aplicado a determinada tabela, o banco de dados pode acessar a informação de forma mais rápida, tendo em vista que pelo índice fornecer a localização exata de cada dado, o banco não irá realizar o scan por todos os

dados na tabela, até achar o correto, mas sim acessá-lo de forma direta se orientando pela localização armazena no índice. (DAMIN,2019)

2.10 Modelo Entidade Relacionamento (MER)

O modelo entidade relacionamento é um modelo conceitual utilizado para descrever as entidades definidas em determinado escopo, juntamente com seus atributos e relacionamentos, fornecendo uma visão abstrata de como ficara o banco de dados relacional. (LUCICHART, 2019)

2.11 Erro relativo

Procedimentos repetitivos que são mais suscetíveis a possíveis erros, sendo assim, quando se tem a necessidade de repetição de um determinado experimento, as medições e cálculos passam a seguir a grandeza de comparações, tais comparações incluem diferenças entre diversas origens, podendo obter tais diferenças dentro de um valor real.

Um erro relativo é a diferença entre um valor base determinado para com o valor obtido, seguindo a seguinte formula: erro relativo = | valor base – valor obtido | / valor obtido * 100. (HAZEWINKEL, 2001) (STEEL, ROBERT GD, TORRIE, JAMES H, 1960)

3 DESCRIÇÃO DO AMBIENTE

3.1 Base de Dados Usada

Os estudos foram conduzidos em um ambiente local teste, do qual será utilizado uma base de dados com um total de 300.000 registros relacionados entre si, divididos em três tabelas contendo 100.000 registros cada, tal base de dados será manipulada através do sistema gerenciador de banco de dados MySQL.

O relacionamento da base de dados é estabelecido por uma chave primaria para com as chaves estrangeiras estabelecidas nas outras 2 tabelas.

A base de dados foi gerada através do site gratuito chamado *Fake Name Generator*: https://pt.fakenamegenerator.com/order.php, da qual pode gerar diversos campos, em arquivos csv ou mysql, com limite de até 100000 registros.

A base de dados gerada, foi ajustada para importação através da ferramenta *sublime text*, onde foram alterados os nomes padrão gerados pela ferramenta *Fake Name Generator.*

Como se trata de um ambiente de estudo e teste com intuito de apenas demonstrar consultas não otimizadas e otimizadas, a cardinalidade definida para a base de dados, foi de 1 para n, sem nenhuma restrição.

Abaixo segue o modelo relacional (Figura 1) da base de dados, onde os ícones com a imagem de uma lâmpada indicam uma chave primaria, os ícones azuis indicam os atributos e os ícones vermelhos indicam uma chave estrangeira.

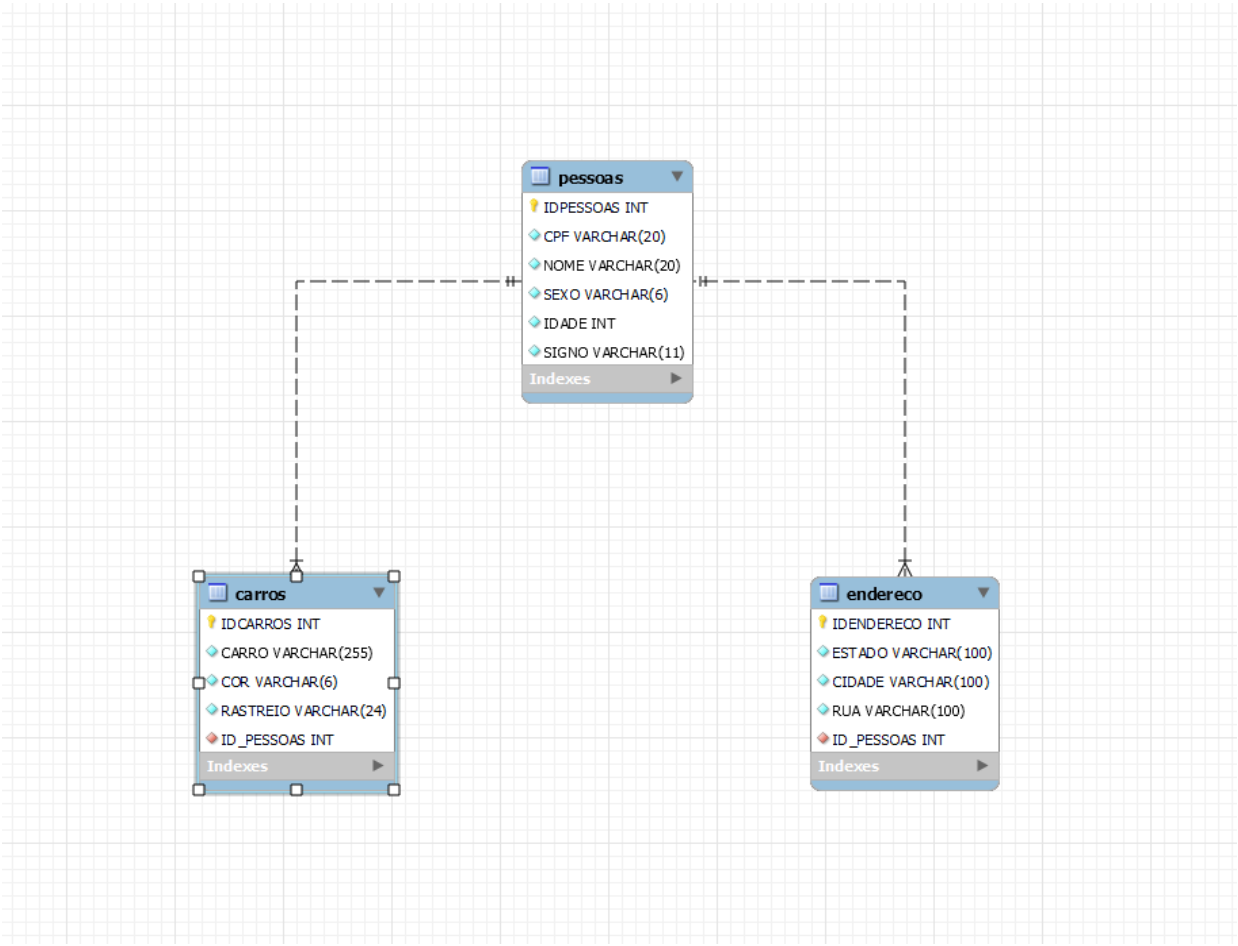


Figura 1: Modelo relacional da base de dados gerado por engenharia reversa através do MySQL workbench (autoria própria), 2021

3.2 Versão do SGBD e Hardware

A versão utilizada do SGBD é a MySQL Server Community 8.0.26 arquitetura 64 bits, a mesma irá rodar na máquina Acer Predator Helios 300, processador i7 9750h com 6 núcleos de processamento e 12 threads, 32gb de memória RAM e 256 gb de SSD M2 NVME de armazenamento interno.

3.3 Benchmarks

Os benchmarks serão realizados através de duas medidas de tempo de execução, por script executado.

A primeira medida será feita pela ferramenta mysql workbench que retornara o tempo de duração (tempo de execução) e o tempo de busca (tempo de retorno dos dados) em segundos.

A segunda medida será feita através da função **microsecond(CURTIME(4))**, implementada aos scripts desenvolvidos no decorrer do estudo, tal função retorna o tempo de execução de cada script em microssegundos.

3.4 Analise e Interpretação de Dados

O processo de benchmark utilizado possuirá algumas limitações, devido ao ambiente estar implementado em uma escala bem menor em relação aos grandes ambientes de produção que são utilizados massivamente.

A primeira ressalva que influenciara nos testes, é tamanho da base de dados, que pode influenciar em algumas otimizações por apresentar um tempo de execução muito pequeno ou quase nulo, tão limitação está liga ao gerador de dados, em fornecer um limite máximo de 100000 registros.

A segunda ressalva se da pelo uso isolado da base de dados comparado a um ambiente empresarial em seu dia a dia, fazendo com que muitas vezes não seja possível apresentar percas de desempenho significativas.

A ultima e terceira ressalva, se da pelo processo de otimização interno do próprio sistema gerenciador de banco de dados, que pode acabar impactando na leitura de desempenho utilizada pela ferramenta mysql workbench, do qual em alguns casos de teste será possível realizar somente a medida através da função microsecond.

Outro impacto causado por este otimizador automático, é a variação de resultados de consultas para consultas, fazendo com que fosse adotado a moda entre dez execuções sobre o mesmo script, escolhendo o resultado que mais se repete neste período de dez tentativas por caso de teste.

A apresentação dos resultados será feita pela diferença dos scripts menos eficientes para os mais otimizados, assim como também será apresentado a porcentagem do ganho de desempenho através do calculo do erro relativo entre as duas medidas.

O erro relativo é calculado através da seguinte formula:

$$ErroRelativo = \frac{(|(TempoMenosEficiente) - (TempoOtimizado)|)}{TempoOtimizado} * 100$$

4 PRINCIPAIS BOAS PRATICAS

4.1 PLANEJAMENTO

Por trás de todo banco de dados bem consolidado e otimizado, há um projeto sistematicamente bem planejado, um banco de dados bem planejado pode não somente entregar um bom desempenho a um cliente, mas também acaba fornecendo uma alta capacidade de adaptabilidade a mudanças não planejadas. (ALVES, 2017)

Um banco de dados planejado de forma incorreta, pode acabar acarretando percas de desempenhos significativas no futuro, ou até mesmo levando a inviabilidade de um projeto comprometendo a entrega dos requisitos acordados pelo cliente. Para a implementação de um planejamento eficaz para um banco de dados relacional, há boas práticas que podem ser seguidas no decorrer do processo de construção, como a análise de requisitos, modelo conceitual, modelo logico, modelo físico, sintaxe de scripts e regras de segurança. (ALVES, 2017; BELEM, 2010)

Na prática de análise de requisitos deve se compreender as reais necessidades do cliente, é nesta etapa que se deve compreender as regras de negócio envolvidas, relacionamentos, escopos do projeto, ambiente e o que deve ser alterado ou mantido, assim consolidando a ideia inicial do projeto.

Uma vez, os requisitos levantados, se tem uma visão mais clara e objetiva sobre o que deve ser feito, possibilitando um desenho de modelo mais próximo possível da real realidade do cliente, facilitando e garantindo o sucesso para as próximas etapas de construção do planejamento do banco de dados.

Sendo assim, será mais simples avaliar a média de registros que serão cadastrados a cada tabela, as consultas que serão elaboradas com mais frequência,

informações sobre certos dados para evitar redundâncias, levantamento de futuros cenários de mudança, reduzindo a imprevisibilidade das mesmas causando maior acurácia na entrega de componentes do projeto.

Na prática de desenho conceitual do projeto, são estabelecidas as principais entidades e relacionamentos que atuaram no escopo do projeto, tal desenho conceitual será representado através de uma prática muito usada, chamada de Modelo entidade relacionamento.

Esta etapa é de grande importância para construção de um planejamento de um banco de dados, por ser na mesma em que é definida a cardinalidade entre as entidades envolvidas, assim como o fluxo de seus respectivos relacionamentos, com base nas regras de negócio estabelecidas no levantamento de requisitos.

Na prática de modelo logico defini se os atributos, tipos e tamanhos das entidades definidas no modelo conceitual, onde será composta a estrutura física do banco de dados relacional, passando a ganhar forma através da conversão e construção dos respectivos itens em tabelas.

Essa prática é crucial para a acurácia do modelo, pois para cada erro cometido nesta etapa, ele pode acabar requerendo a necessidade de alterações para um projeto que já está em desenvolvimento ou em produção, provocando impacto diretamente nos prazos, custos, desempenho e acurácia do produto final.

Como parte final do planejamento de um banco de dados relacional, é definido os aspectos técnicos e físicos de um banco de dados, do qual será abordado itens de extrema importância como armazenamento de dados, geração de scripts, SGBD, sintaxe de scripts e sistema operacional.

Também incluso nesta etapa irão ser abordados assuntos financeiros como orçamentos e planos de viabilidade e assuntos de segurança como ambientes, protocolos e rotinas que irão ser implementadas. (ALVES, 2017; BELEM, 2010; DOMINICO, 2013)

4.2 SCRIPTS EM SQL

4.2.1 Operadores Lógicos

Os operadores lógicos dentro dos scripts SQL, podem acabar causando grandes problemas de desempenho se não usados de forma correta e analítica.

Todavia com boas práticas aplicadas adequadamente em sua estrutura, se tem operações e consultas executas em um menor período e tempo, juntamente com uma vazão de dados maior, causando um reflexo positivo ao desempenho geral de um banco de dados relacional.

A linguagem SQL apresenta os seguintes operadores lógicos: OR, AND e NOT, antes de aplica-los a qualquer script, deve se levar em consideração a ordem com que as entidades em questão, estão sendo manipuladas, juntamente com suas características como sua contagem geral.

Tais detalhes analisados e levantados em relação as entidades em uso, serão verificados através de uma prática extremamente conhecida, chamada tabela verdade.

Através da tabela verdade se tem uma visão completamente analítica e logica de como a execução do script irá ocorrer com os conectivos lógicos, fornecendo uma visão clara o suficiente para determinar a ordem com que cada elemento será manipulado dentro do banco.

Uma vez conseguindo determinar uma ordem de verificação, consequentemente conseguimos realizar menos verificações redundantes dentro de um script, assim diminuindo o seu tempo de execução.

Começando pelas consultas mais simples, iremos atender uma requisição de consulta, que solicita uma listagem de pessoas do sexo masculino ou feminino.

O conectivo logico “ou” tem o mesmo conceito logico que é aplicado a uma tabela verdade, da qual para uma proposição ser verdadeira, basta apenas que uma das condições seja verdadeira.

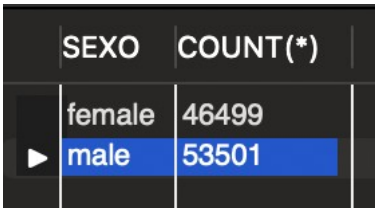
Logo afim de obter menos verificações redundantes dentro de uma consulta, é realizado uma contagem do número de elementos de determinado atributo de interesse em um banco de dados.

Como no exemplo abaixo da figura 1 e 2 o atributo a ser analisado é o sexo e signo, sendo que uma vez que se tem a contagem determinada de cada elemento, podemos determinar a ordem com que cada atributo será verificado no script de acordo com cada tipo de demanda.

Selecionando os indivíduos que sejam do sexo masculino **ou** do signo de capricórnio, iremos começar pelo levantamento de dados, começando pela contagem de cada atributo.

Para isso é usado os seguintes scripts:

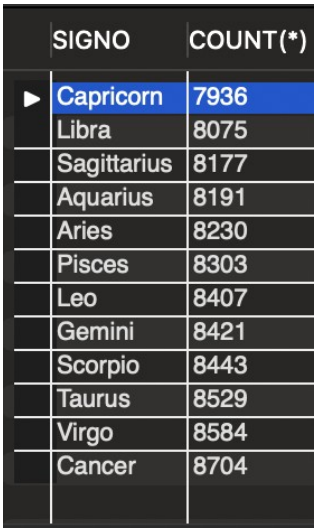
```
SELECT SEXO, COUNT(*)
FROM PESSOAS
GROUP BY SEXO
ORDER BY 2;
```



SEXO	COUNT(*)
female	46499
male	53501

Figura 2: contagem de sexo (autoria própria), 2021

```
SELECT SIGNO, COUNT(*)
FROM PESSOAS
GROUP BY SIGNO
ORDER BY 2;
```



SIGNO	COUNT(*)
Capricorn	7936
Libra	8075
Sagittarius	8177
Aquarius	8191
Aries	8230
Pisces	8303
Leo	8407
Gemini	8421
Scorpio	8443
Taurus	8529
Virgo	8584
Cancer	8704

Figura 3: contagem de signo (autoria própria), 2021

Com a seguinte contagem, observasse que de 100.000 pessoas 46.499 são mulheres e 53.501 são homens, representando 53,501% do total de pessoas, já para a relação da quantidade de pessoas por signos temos 7.936 pessoas do signo de capricórnio, representando 7,936% do total de pessoas.

Com base nos dados levantados, a boa prática indica a prioridade do atributo ‘sexo’ na construção do script, tal escolha se baseia na logica do conectivo logico ***or***(ou), que necessita de apenas uma condição verdadeira para que toda proposição também seja.

Logo devemos verificar com prioridade, o atributo que possui a maior quantidade na tabela do qual é o atributo ‘sexo’ que compõe 53,501% dos registros, sendo que o atributo signo de capricórnio possui apenas 7,936%.

Uma vez a condição sendo verdadeira, a segunda condição não é verificada, consumindo menos tempo, processamento e aumentando a vazão de dados, devido a diminuição do numero de verificações na tabela.

Realizando a consulta das pessoas do sexo masculino ou do signo de capricórnio através do seguinte script.

```
SELECT NOME, SEXO, SIGNO
FROM PESSOAS
WHERE  SEXO = 'MALE' OR  SIGNO =  'CAPRICORN';
```

Como resultado temos um resultado significativo do tempo de execução, comparado ao que se teria se priorizássemos o atributo de menor quantidade primeiro, aumento o número de verificações.

Script desenvolvido de forma menos eficiente, juntamente com seu tempo de execução.

```
SELECT NOME, SEXO, SIGNO, microsecond(CURTIME(4))
FROM PESSOAS
WHERE SIGNO =  'CAPRICORN' OR SEXO = 'MALE';
```

Tempo de execução 1: 227500 microssegundos

Figura 4: tempo de execução 2 do script menos eficiente usando conectivo logico ou (autoria própria), 2021

Script desenvolvido de forma eficiente com o atributo correto priorizado em sua construção.

```
SELECT NOME, SEXO, SIGNO,  microsecond(CURTIME(4))
FROM PESSOAS
WHERE SEXO = 'MALE' OR SIGNO =  'CAPRICORN';
```

Tempo de execução 1: 128600 microssegundos

Figura 5: tempo de execução 2 do script eficiente usando conectivo logico ou (autoria própria), 2021

Temos uma otimização 98900 microssegundos na primeira medida.

Na segunda medida utilizando o workbench MySQL, temos **0,03 seg / 0,009 seg** de duração e tempo de busca a menos respectivamente.

Calculando o erro relativo das medidas 1 e 2, temos um ganho de desempenho de 76.90% e 1500.0% / 15.78% (tempo de duração/ tempo de busca) respectivamente.

Para que uma proposição com conectivo logico ***AND*** (E) seja verdadeira, todas as condições devem ser verdadeiras, logo durante a execução de um script, basta que apenas a primeira condição de o valor seja falsa, para que a segunda condição não seja verificada.

Considerando este comportamento de verificação para o conectivo logico AND, é de boa prática priorizarmos na verificação do script, o atributo que possuir a menor quantidade, pois sendo assim, se aumenta as chances de um resultado falso na primeira verificação, logo anulando a segunda.

Usando o mesmo exemplo utilizado anteriormente, iremos construir um script para listar informações de pessoas que sejam do sexo masculino e que sejam do signo de capricórnio.

Como os dados já estão levantados, iremos dar prioridade na verificação para o atributo signo, pois segundo os dados levantados, o signo de capricórnio compões apenas 7,936% do total de registros na tabela.

Logo priorizando o mesmo nas verificações, a probabilidade da verificação, dar um valor que não atenda as condições é maior, assim reduzindo o numero de verificações e diminuindo o tempo de execução, aumentando o seu desempenho.

Script desenvolvido de forma menos eficiente, juntamente com seu tempo de execução.

```
SELECT NOME, SEXO, SIGNO
FROM PESSOAS
WHERE  SEXO = 'MALE' AND SIGNO ='CAPRICORN'  ;
```

Tempo de execução 1: 265400 microssegundos

Figura 6: tempo de execução 2 do script menos eficiente usando conectivo logico E (autoria própria), 2021

Script desenvolvido de forma eficiente, juntamente com seu tempo de execução.

```
SELECT NOME, SEXO, SIGNO
FROM PESSOAS
WHERE  SIGNO =  'CAPRICORN' AND SEXO = 'MALE';
```

Tempo de execução 1: 234500 microssegundos

Figura 7: tempo de execução 2 do script eficiente usando conectivo logico E (autoria própria), 2021

Temos uma otimização 30900 microssegundos na primeira medida.

Na segunda medida utilizando o workbench MySQL, temos **0,0027 seg / 0,012 seg** de duração e tempo de busca a menos respectivamente.

Calculando o erro relativo das medidas 1 e 2, temos um ganho de desempenho de 13,17% e 52,94% / 41,37% (tempo de duração/ tempo de busca) respectivamente.

Lidando com ambos operadores AND e OR, deve se analisar cada condição separadamente afim de obter um valor logico, para que a mesma seja novamente analisada com a segunda condição.

Realizando a listagem de todas as pessoas que seja do sexo feminino **e** sejam do signo de capricórnio **ou** de câncer, deve se montar o script em partes, começando pela primeira condição de verificação, seguindo a mesma lógica da tabela verdade tanto para o conectivo AND quanto para OR.

Conforme os dados que temos levantados, temos 46.499 pessoas do sexo feminino, 7936 pessoas do signo de capricórnio e 8704 do signo de câncer, montando primeira condição com o conectivo AND.

Priorizando o atributo com a menor quantidade, deve se verificar primeiro o signo de capricórnio, logo o script será construído da seguinte forma:

(SIGNO = 'capricorn' AND SEXO = 'female')

Por sequência, na segunda condição priorizamos a verificação do segundo atributo com a menor quantidade de registros.

(SIGNO = 'cancer' AND SEXO = 'female');

Com as condições elaboradas separadamente, deve se priorizar a condição que possui as maiores chances de um resultado verdadeiro, devido a inserção do conectivo logico OR.

```
SELECT NOME, SEXO, SIGNO
FROM PESSOAS
WHERE
(SIGNO = 'capricorn' AND SEXO = 'female')
OR
(SIGNO = 'cancer' AND SEXO = 'female');
```

Tempo de execução 1: 313700 microssegundos

Logo das duas condições elaboradas com o conectivo AND, a que possui mais chances de uma saída verdadeira em relação a segunda condição, é a condição que prioriza o signo de capricórnio.

Uma vez retornando mais valores verdadeiros, a segunda condição é anulada pelo conectivo logico OR, resultando em uma otimização de tempo de execução conforme a figura 7.

Figura 8: tempo de execução 2 do script eficiente usando conectivo logico E e OU(autoria própria), 2021

Script elaborado de forma incorreta, priorizando a verificação de condições na ordem errada.

```
SELECT NOME, SEXO, SIGNO
FROM PESSOAS
WHERE
(SIGNO = 'cancer' AND SEXO = 'female')
OR
(SIGNO = 'capricorn' AND SEXO = 'female');
```

Tempo de execução 1: 384400 microssegundos

Figura 9: tempo de execução 2 do script menos eficiente usando conectivo logico E e OU(autoria própria), 2021

Temos uma otimização 70700 microssegundos na primeira medida.

Na segunda medida utilizando o workbench MySQL, temos **0,0001 seg / 0,003 seg** de duração e tempo de busca a menos respectivamente.

Calculando o erro relativo das medidas 1 e 2, temos um ganho de desempenho de 22,53% e 1,21% / 6,52% (tempo de duração/ tempo de busca) respectivamente.

4.2.2 Clausulas de Junções

As junções entre tabelas podem ser realizadas por diversas ocasiões, desde para obter informações através de um cruzamento de dados entre várias tabelas, ou até mesmo para geração de relatórios mais complexos.

Um dos meios mais comuns e usados para junções de tabelas, é através da associação da chave primaria da primeira tabela com as chaves estrangeiras de outras tabelas.

Para executarmos junções entre tabelas, temos alguns comandos classificados por clausulas, as mesmas possibilitam implementar alguns fundamentos da teoria dos conjuntos, algumas das principais clausulas e mais usadas são as INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN e CROSS JOIN.

Cada clausula mencionada, possui suas respectivas particularidades de execução e implementação, gerando a necessidade de análise do comportamento de cada uma, conforme seu desempenho para cada situação de forma particular, que determinará o uso da clausula que melhor atenderá determinada situação.

Embora o uso de cada clausula de junção varie conforme a sua necessidade e situação, há boas práticas podem ser aplicadas na construção todas as clausulas afim de otimizar seu desempenho. (ARAUJO, 2017; PRASS, 2019; LOPES, 2013)

Ao construirmos um script que trará informações de várias tabelas, deve-se usar as cláusulas de junções para construir a correlação de dados entre as mesmas, como no seguinte exemplo 1 abaixo:

1) Traga o nome, sexo, estado, cidade, carro e numero de rastreio do carro de cada usuário cadastrado no banco.

```
SELECT PESSOAS.NOME, PESSOAS.SEXO, ENDERECO.ESTADO,
ENDERECO.CIDADE, CARROS.CARRO, CARROS.RASTREIO
FROM PESSOAS
INNER JOIN ENDERECO
ON PESSOAS.IDPESSOAS = ENDERECO.ID_PESSOAS
INNER JOIN CARROS
ON PESSOAS.IDPESSOAS = CARROS.ID_PESSOAS;
```

Tempo de execução 1: 631400 microssegundos

Figura 10: tempo de execução 2 do script menos eficiente com uso de ponteiramento por atributos (autoria própria), 2021

Nota se que ao usar a clausula de junção INNER JOIN, os atributos selecionados, devem estar apontados para as suas respectivas origens, afim de evitar a ambiguidade e erro durante sua execução.

O uso do ponteiro para especificação da origem de cada atributo, faz com que o banco de dados realize uma busca por todos os atributos mencionados, a cada comparação.

Sendo assim durante as junções de determinadas tabelas, há verificações redundantes no processo de junção, como no exemplo 1 acima, ao realizar a junção entre as tabelas PESSOAS e ENDERECO, a busca também será realizada de forma redundante sobre a tabela CARROS, que possui atributos mencionados no script.

Observando se a construção e o uso dos ponteiros no script acima, nota se que por haver verificações redundantes, consequentemente há uma perca de desempenho.

Sendo assim é de boa prática, fazer o uso de ponteiramento através de nomenclaturas ou apelidos especificando a origem através de cada tabela envolvida ao invés de cada atributo.

Tal boa prática leva o banco de dados, a realizar menos verificações, uma vez que a origem de cada atributo está especificada através de cada tabela,

apontando para seus respectivos atributos, realizando somente as verificações necessárias conforme cada operação de junção.

Realizando a construção do script para atender o exemplo 1, utilizando as boas práticas, temos:

```
SELECT P.NOME, P.SEXO, E.ESTADO, E.CIDADE, C.CARRO, C.RASTREIO, microsecond (CURTIME(4))
FROM PESSOAS P
INNER JOIN ENDERECO E
ON P. IDPESSOAS = E.ID_PESSOAS
INNER JOIN CARROS C
ON P. IDPESSOAS = C.ID_PESSOAS;
```

Tempo de execução 1: 488100 microssegundos

Figura 11: tempo de execução 2 do script eficiente com uso de ponteiramento por tabelas (autoria própria), 2021

Temos uma otimização 143300 microssegundos na primeira medida.

Na segunda medida utilizando o workbench MySQL, temos **0,001 seg / 0,016 seg** de duração e tempo de busca a menos respectivamente.

Calculando o erro relativo das medidas 1 e 2, temos um ganho de desempenho de 29,30% e 6,67% / 2% (tempo de duração/ tempo de busca) respectivamente.

Outra boa prática que pode ser aplicada a qualquer clausula de junção, é a priorização das tabelas com menores quantidades de registro na construção do script, priorizando as menores tabelas para serem verificados primeiro, o numero de verificações vai diminuindo respectivamente a cada junção, fazendo com que ao chegar na maior tabela, a maior parte das junções já tenham sido processadas, evitando redundâncias no processo e otimizando o mesmo.

Como o banco implementado possui todas as tabelas com as mesmas quantidades de registros, segue o exemplo teórico de com funcionar este processo de otimização.

Exemplo 2:

Tabela_A = 150000 registros

Tabela_B = 100000 registros
Tabela_C = 50000 registros

Construindo um script de forma menos otimizada tem se a seguinte estrutura:

```
SELECT A.ITEM_A, A.ITEM2_A, B.ITEM_B, B.ITEM2_B, C.ITEM_C, C.ITEM2_C
FROM Tabela_A A
INNER JOIN  Tabela_B B
ON A.IDTabela_A = B.IDTabela_B
INNER JOIN Tabela_C C
ON A.IDTabela_A = C.IDTabela_C;
```

Construindo um script de forma otimizada tem se a seguinte estrutura:

```
SELECT C.ITEM_C, C.ITEM2_C , B.ITEM_B, B.ITEM2_B,  A.ITEM_A, A.ITEM2_A
FROM Tabela_C C
INNER JOIN  Tabela_B B
ON C.Tabela_C = B.IDTabela_B
INNER JOIN Tabela_A A
ON C.Tabela_C = A.IDTabela_A;
```

Priorizando as tabelas menores no segundo script a cima, as mesmas irão ser armazenadas na memória mais efetivamente, fazendo com que INNER JOIN seja executado de forma mais rápida, devido a menor quantidade de filtros para as tabelas maiores.

Levando em consideração que também o script SQL seja processado da esquerda para direita, sendo assim, se a tabela menor estiver priorizada na ordem, ela será processada primeiro utilizando memória de forma mais efetiva.

Demonstrando também como o uso das cláusulas de junções dependem muito da situação em que são aplicadas, como neste caso sendo favorável o uso da clausula LEFT JOIN, caso o SGBD realizasse da execução da esquerda para direita, mostrando mais uma vez que cada ambiente deve ser analisado antes das aplicações de cada clausula.

4.2.3 Bom Uso de Índices

Em um banco de dados, é comum termos determinadas tabelas, que requerem acessos diariamente, tanto para consultas ou até outras operações como alterações de um determinado registro, fazendo com que quanto maior o tamanho da tabela, maior será o gasto computacional para realizar determinadas operações, muitas vezes levando um tempo considerável para a conclusão de determinada operação.

Diante de tal situação, há vários métodos e boas práticas que podem ser aplicados para otimização do tempo de resposta com que acessamos determinadas tabelas, um dos recursos mais usados nesta situação, é o uso de índices.

Um índice é uma estrutura em disco associada a uma tabela ou view, que otimiza a busca por linhas, o mesmo contém chaves criadas de uma ou mais colunas, das quais são armazenadas em uma estrutura (árvore B) que habilita o SGBD a localizar a linha ou as linhas associadas aos valores de chave de forma otimizada. (RESENDE,2015; LOPES, 2013)

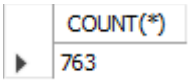
O uso de índices para otimização possui algumas ressalvas que devem ser levadas em consideração, pois se usados de forma inadequada, pode comprometer o desempenho do banco de dados de forma significativa.

Algumas das ressalvas de seu uso são:

- Alto custo computacional em sua criação, pois o mesmo irá criar uma chave para cada registro contido na tabela, recomendando sua criação em fora de horário de pico.
- Alto custo computacional para atualização de registros, pois ao atualizar determinado registro, todos os outros índices criados, serão atualizados também.

Voltando ao banco de dados implementado, vamos realizar a contagem sem o uso do índice, das pessoas que se chamam ‘Alice’.

```
SELECT COUNT(*)
FROM PESSOAS
WHERE NOME = 'Alice';
```



COUNT(*)
763

Figura 12: contagem do numero de registros com nome ‘Alice’

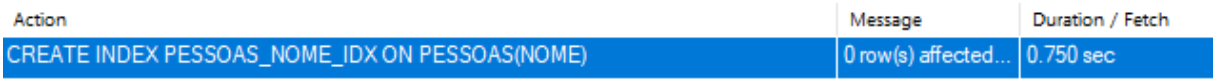
Tempo de execução 1: 850800 microssegundos

SELECT COUNT(*), microsecond(CURTIME(4)) FROM PESSOAS WHERE NOME ...	1 row(s) returned	0.031 sec / 0.000 sec

Figura 12: Tempo de execução 2 contagens de elementos sem uso de índice.

Agora aplicando índice ao atributo nome, teremos o seguinte script e tempo de execução.

```
CREATE INDEX PESSOAS_NOME_IDX
ON PESSOAS(NOME);
```



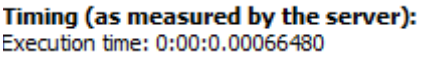
Action	Message	Duration / Fetch
CREATE INDEX PESSOAS_NOME_IDX ON PESSOAS(NOME)	0 row(s) affected...	0.750 sec

Figura 13: Tempo de execução para aplicação dos índices.

Realizando novamente a contagem de pessoas com o nome ‘Alice’, porem com uso de índices, temos o seguinte script e tempo de execução:

```
SELECT COUNT(*)
FROM PESSOAS
WHERE NOME = 'Alice';
```

Tempo de execução 1: 355000 microssegundos



Timing (as measured by the server):
Execution time: 0:00:0.00066480

Figura 14: Tempo de execução 2 contagens de elementos com uso de índice

Temos uma otimização 495800 microssegundos na primeira medida.

Na segunda medida utilizando o workbench MySQL, temos **0,03034seg** de tempo de execução.

Calculando o erro relativo das medidas 1 e 2, temos um ganho de desempenho de 139.66% e 4596.96% (tempo de duração/ tempo de busca) respectivamente.

Há alguns fatores devem ser levados em consideração para que haja o aproveitamento correto, do ganho de desempenho que o uso dos índices pode oferecer, alguns deles são:

- Conversão de dados: Se da utilização incorreta das tipagens definidas na construção de uma tabela, por exemplo utilizar um valor inteiro como parâmetro, entre aspas como um varchar, para um atributo que foi definido como inteiro, tal prática resultara com que o índice criado para determinada coluna, não seja usado, desperdiçando uma oportunidade de otimização.

  Exemplo 1:

```
SELECT NOME
FROM PESSOAS
WHERE IDPESSOAS = ‘20’;
```

  Neste caso o atributo IDPESSOAS foi criado como inteiro e chave primaria, da qual automaticamente já gera um índice por natureza, do qual não será usado nesta consulta, devido a passagem incorreta de um parâmetro com a tipagem errada.

- Operadores aritméticos: Ao modicar se determinado dado de uma coluna com operações aritméticas de forma direta na consulta, resulta na anulação do uso do índice na execução da consulta.

  Exemplo 2:

```
SELECT NOME, IDADE
FROM PESSOAS
WHERE IDADE-18 >= 0;
```

  Para descobrirmos as pessoas que são maiores de 18 anos, é possível também realizar a mesma consulta utilizando o parâmetro >= diretamente com a idade definida, desse jeito, não realizando nenhuma modificação direta no atributo idade.

  Exemplo 3:

```
SELECT NOME, IDADE
FROM PESSOAS
WHERE IDADE >= 18;
```

- Valores Nulos: Os índices não possuem referência para valores do tipo null, isto é, pelos valores null, poderem ser recuperados somente por uma leitura sequencial completa. (LOPES, 2013)

4.2.4 Uso de filtros de pesquisa

Há diversas maneiras de se construir uma consulta ao banco de dados, para trazer determinadas informações, porem há algumas boas práticas que são recomendadas para determinadas situações de busca ou seleção.

A primeira situação, se encontra no uso da clausula SELECT, da qual deve se analisar quais atributos devem ser retornados, afim de evitar redundâncias e consequentemente causando a otimização do processo.

A ideia principal desta boa prática de otimização é restringir os itens a serem retornados à somente as informações realmente necessárias, tomando muita atenção aos atributos que são pedidos, ou até mesmo aos elementos especificados, dos quais podem ter filtrados através da clausula WHERE.

Um erro muito comum é o uso de caracteres coringa como passagem de parâmetros, para trazerem todas as colunas(atributos) das tabelas, que acaba sendo um causador de gargalos, por além de fazer uma busca completa por todos os registros, o mesmo irá trazer todos os dados contidos na tabela, exigindo mais memória e processamento. (ARAUJO, 2017; LOPES, 2013)

Exemplo 1 – traga o nome e signo de cada usuário cadastrado no banco.

Construindo o script de forma incorreta utilizando caractere coringa

SELECT * FROM PESSOAS;

Tempo de execução: 342900 microssegundos

Construindo o script de forma correta, filtrando somente os atributos necessários como nome e signo.

SELECT NOME, SIGNO FROM PESSOAS;

Tempo de execução:198700 microssegundos

Nesta situação otimizando 72,5%

A segunda situação recomenda se substituir o uso da clausula HAVING pela clausula WHERE, isto é, pelo funcionamento da clausula HAVING, que é destinada para filtrar um grupo de linhas somente depois que as mesmas forem agrupadas.

O que gera o aumento de tempo e processamento para a execução da mesma, fazendo com que seja indicado primeiro, uso da clausula WHERE na maior parte das vezes, pela mesma filtrar somente as linhas, assim evitando o tempo e processamento que seria executado no agrupamento das linhas filtradas.

Exemplo 2 – Traga o nome e idade das pessoas que possuam 50 anos ou menos.

Construindo o script utilizando a clausula HAVING.

```
SELECT NOME, IDADE
FROM PESSOAS
HAVING IDADE <= 50;
```

Tempo de execução: 470100 microssegundos

Construindo o script utilizando a clausula WHERE.

```
SELECT NOME, IDADE, microsecond(CURTIME(4))
FROM PESSOAS
WHERE IDADE <= 50;
```

Tempo de execução: 309200 microssegundos

Nesta situação otimizando 52,03%

4.3 UTILIZAÇÃO DE MEIOS DE ARMAZENAMENTO

Atualmente com o avanço da tecnologia voltado para os meios de armazenamento de dados, acaba que por sua vez possibilitando diversos meios para sua aplicação na área de otimização de desempenho de banco de dados.

Os meios mais aplicados para banco de dados, são os seguintes dispositivos de hardwares: Disco rígido (HD), Unidade de armazenamento de estado solido (SSD) e pôr fim a memória de acesso aleatório (RAM).

4.3.1 Discos rígidos (HDs)

Segundo o artigo publicado pelo autor Mauro Pichiliani, no site Devmedia, os discos rígidos (HDs) mecânicos foram os primeiros dispositivos de armazenamento em massa empregados na computação e até hoje seus princípios são utilizados.

Esse tipo de hardware de armazenamento é composto por discos magnéticos (ou discos rígidos) que armazenam as informações, onde a leitura e gravação dos dados são efetuadas por uma agulha mecânica. (PIONTKOSKI, 2017; BRITO, 2021)

Os discos rígidos mecânicos, embora sejam relativamente antigos, são amplamente usados no mercado de trabalho ate nos dias atuais, devido ao seu baixo custo, acesso sequencial aos dados gravados e por fim a sua vida útil.

Embora os mesmos, ainda sejam utilizados em larga escala, por serem compostos por uma estrutura com partes mecânicas, os mesmos devem apresentar um menor desempenho, devido a fatores de latência rotacional e tempo de busca. (GEHRKE; RAMAKRISHNAN,2008)

Tendo em vista as limitações físicas dos discos rígidos (HDs), os memos podem apresentar uma velocidade máxima de ate 200mb/s, do qual está velocidade máxima pode ser encontrada somente nos discos de alta velocidade, já para os discos padrão podem alcançar velocidades de transferência de até 150mb/s.

O que geralmente diferencia na velocidade de um disco rígidos, são as rotações por minuto, que interferem diretamente na velocidade com que são feitos os acessos aos setores do disco.

Os discos rígidos domésticos possuem uma velocidade de 5400rpm podendo chegar até 7200rpm.

Os discos de alta velocidade, são mais usados em ambiente corporativos, onde demandam mais leituras e acessos, podendo chegar a altas velocidades de até 15000rpm, porem sofrendo alteração nas suas interfaces de leitura, para fornecer mais velocidade e controle de erros.

Figura 14: HD demonstrado
Fonte: (BRITO.F,2021)

4.3.2 Solid State Drives (SSDs)

As unidades de armazenamento de estado solido (tradução de Solid State Drives), não são compostas de partes mecânicas, que por sua vez já não são suscetíveis a possíveis falhas mecânicas, já que não há nada a ser movimentado em seu interior.

Em geral os SSDs possuem dois componentes em sua construção, a memoria flash e o controlador.

A memoria flash tem a função de armazenar todos os arquivos e dados, enquanto os controladores tem a função de gerenciar a troca de dados entre a memoria e o dispositivo em que a mesma está instalada.

Devido à ausência dessas peças mecânicas, a troca de informações se dá através de cargas elétricas, o que torna as operações de leitura e escrita, muito mais rápidas em comparação ao seu antecessor HD, pelo fato de haver um acesso direto

a informação pretendida, que antes em antecessor, para se acessar determinado dado, havia a necessidade de realizar uma leitura sequencial por diversos setores.

Um ssd pode alcançar velocidades de transferência de ate 3gbs/s, podendo ser ultrapassada, depende do uso da tecnologia que for usada.(PIONTKOSKI, 2017; SALLES,2021)

Figura 15: SSD padrão SATA e M2
Fonte: (SALLES.F,2021)

4.3.3 Memória RAM

A memoria RAM é uma memória de acesso aleatório, ou seja, seu armazenamento não é feito de maneira sequencial, ela usada para armazenar arquivos ou dados de forma temporária, diferente do hd e ssd que pode ser armazenado de forma fixa.

A mesma é responsável pela leitura de dados ou arquivos, quando requeridos por algum programa ou processo.

A memória ram possui duas características relevantes, para o fornecimento de um bom desempenho:

- a primeira é a largura do seu barramento, que diz a respeito do numero de bits que podem ser enviados a cpu simultaneamente,

caracterizando sua velocidade pelo numero de vezes em que um grupo de bits pode ser enviado a cada segundo.

- A segunda característica se dá por sua velocidade em frequência, e latência, representando respectivamente o numero de vezes com que a mesma consegue enviar os bits ao processador e o seu intervalo com que leva para fazer a transferência de informação.

Tendo em mente o funcionamento de uma memória ram, a mesma possui grande aplicabilidade no funcionamento e desempenho de um banco de dados relacional.

Levando o seu uso para o lado de otimização de desempenho, o banco naturalmente já carrega parte de seus dados para a memória ram, o que pode ser considerado como memora cache do SGBD.

Alguns SGBDs permitem optar por quais tabelas serão armazenadas em cache, assim evitando o carregamento de um banco de dados todo para a memória, otimizando o tempo e aumentado a vazão.

O armazenamento de dados em memória cache, é muito útil em situações onde há uma base de dados muito extensa, pois, seguindo a boa prática para otimização, é recomendável priorizar o armazenamento das tabelas que são consultadas com mais frequência, assim evitando o acesso ao banco todo de forma frequente.

Vale ressalvar que não é indicado fazer o armazenamento de tabelas dinâmicas com campos que sempre estejam em constante alteração, pois a cada alteração, a tabela ou informação terá que ser carregada novamente na memória.

Segundo informações da ORACLE (2015) os bancos de dados em memória se diferenciam armazenagem de dados em linhas, porém quando ocupam a parte in memory da SGA (System Global Area) os mesmos são agrupados por blocos de colunas, que otimizam o retorno das consultas e essas podem ser varridas rapidamente, através de técnicas como o SIMD processamento de vetor.(ORACLE, 2015; PIONTKOSKI, 2017)

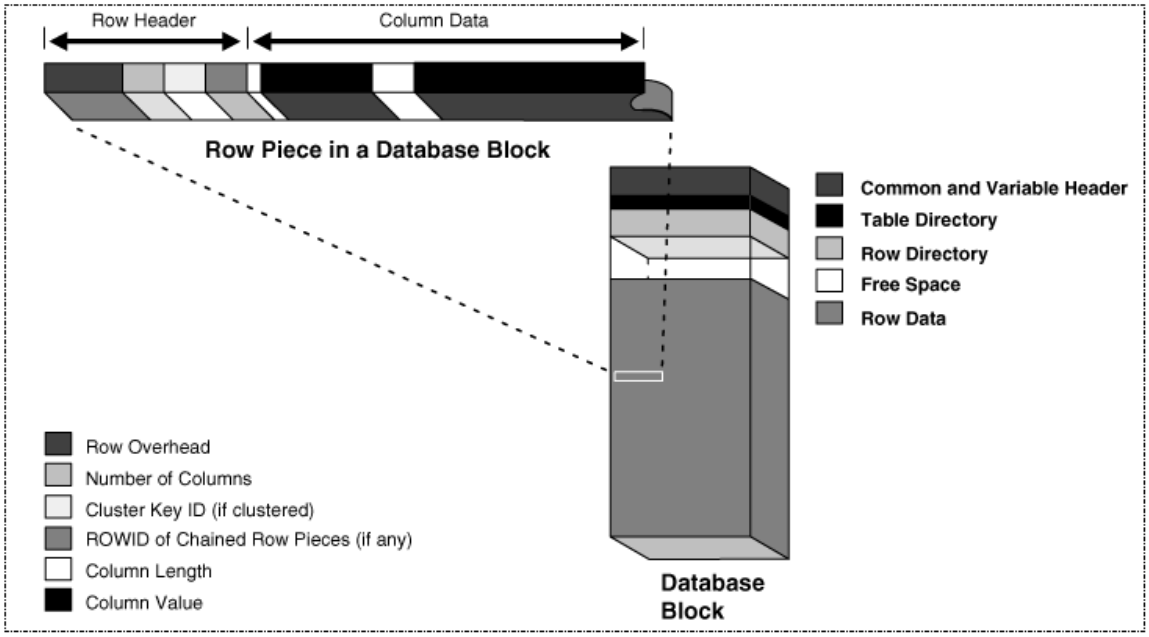


Figura 16: Funcionamento do banco de dados em memoria cache
Fonte: (ORACLE, 2015)

4.3.4 Analise de desempenho entre meios de armazenamento

Segundo o estudo realizado pelo autor Gilvano Piontkoski, sobre o tempo de execução de um milhão de linhas, tem se os gráficos representando o desempenho do banco de dados, conforme o meio de armazenamento usado para o HD, SSD, e memória RAM.

Para cada agrupamento de linhas, são exibidos dois gráficos, um apresentando o tempo de execução de cada consulta e outro apresentando o tempo médio de execução por cada meio de armazenamento, interpretando o menor valor como o melhor desempenho.

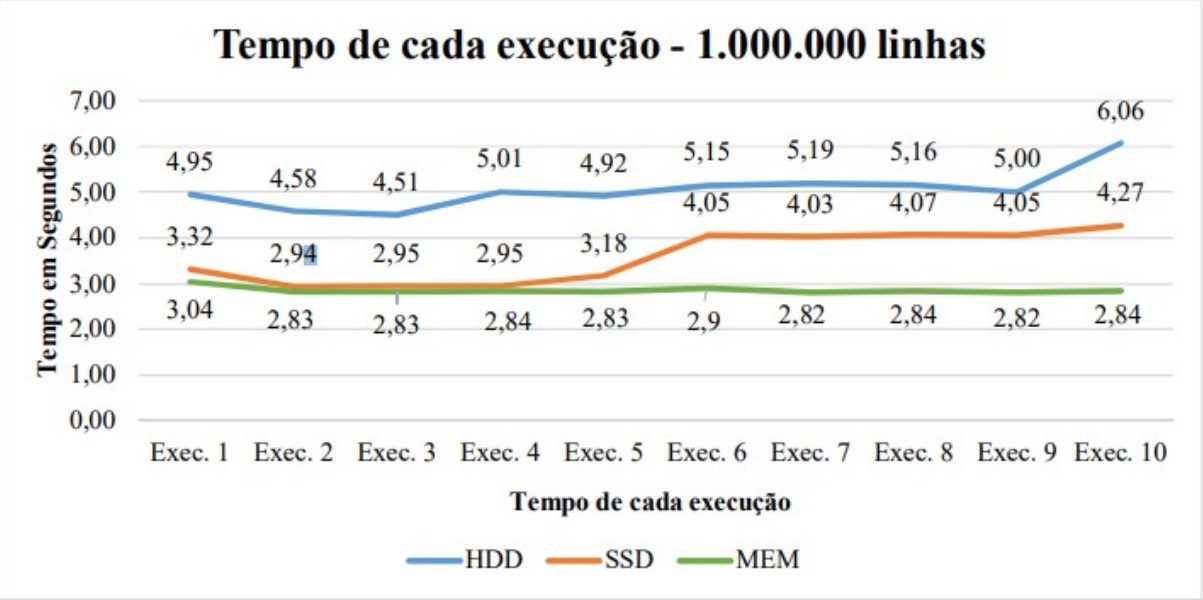


Figura 17: gráfico medindo tempo de execução por cada meio de armazenamento
Fonte: (PIONTKOSKI, 2017)

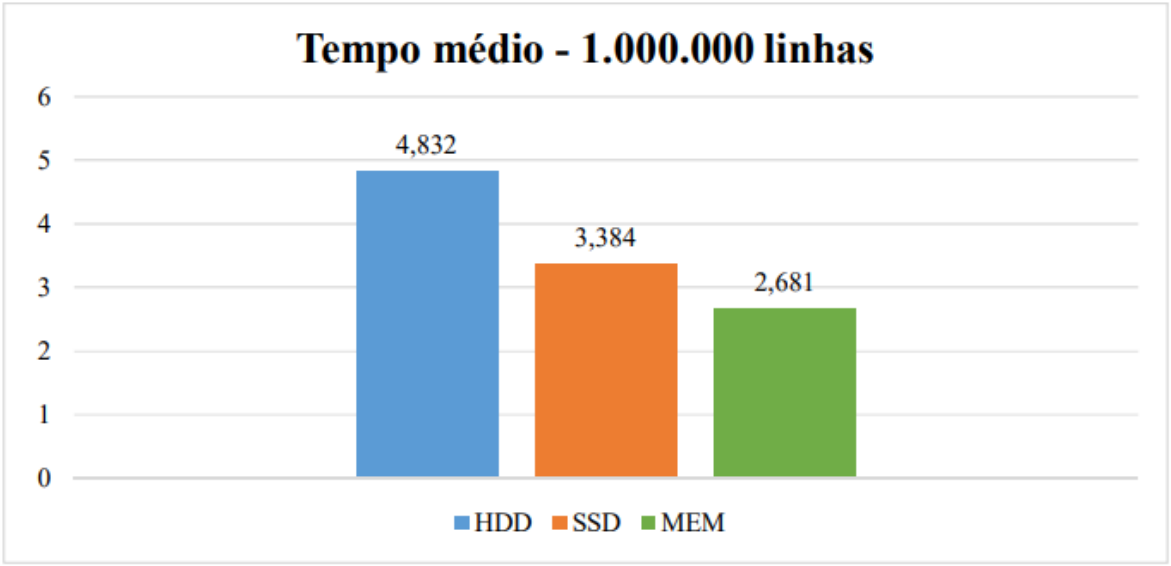


Figura 18: gráfico medindo tempo médio de execução por cada meio de armazenamento
Fonte: (PIONTKOSKI, 2017)

fica claro a vantagem sobre os meios de armazenamento que não utilizam recursos mecânicos, possibilitando um acesso de forma mais direta aos dados desejados (PIONTKOSKI, 2017).

Destacando também o alto desempenho do uso da memória cache através da memoria RAM, comprovando o uso desta boa prática, quando respeitada as suas particularidades, e destacando um bom meio para uma otimização além das boas práticas em cima das elaborações de consultas.

4.4 ANALISE DE AMBIENTE DE PRODUÇÃO

Em um banco de dados em produção, e de boa prática prestar atenção ao seu comportamento durante os horários de uso dele, para que assim ao projetar uma consulta ou script, seja levado em consideração o horário ou dia mais propício para execução dos mesmos.

Tal prática pode vir a ajudar a evitar gargalos em horários de pico, do qual são os momentos em que o banco de dados está trabalhando com uma maior carga de demanda. Para facilitar o processo é recomendado a elaboração de uma matriz de rastreabilidade, tornando mais visíveis os impactos que cada componente pode causar sobre os outros componentes e os sistemas juntamente envolvidos.

5 ANALISE DE IMPACTO DAS BOAS PRÁTICAS NO DESEMPENHO DE CONSULTAS SQL

Consolidando as otimizações realizadas no decorrer do estudo, temos o seguinte gráfico (Figura 18), demonstrando a diferença do uso das boas práticas na elaboração de consultas de um banco de dados relacional.

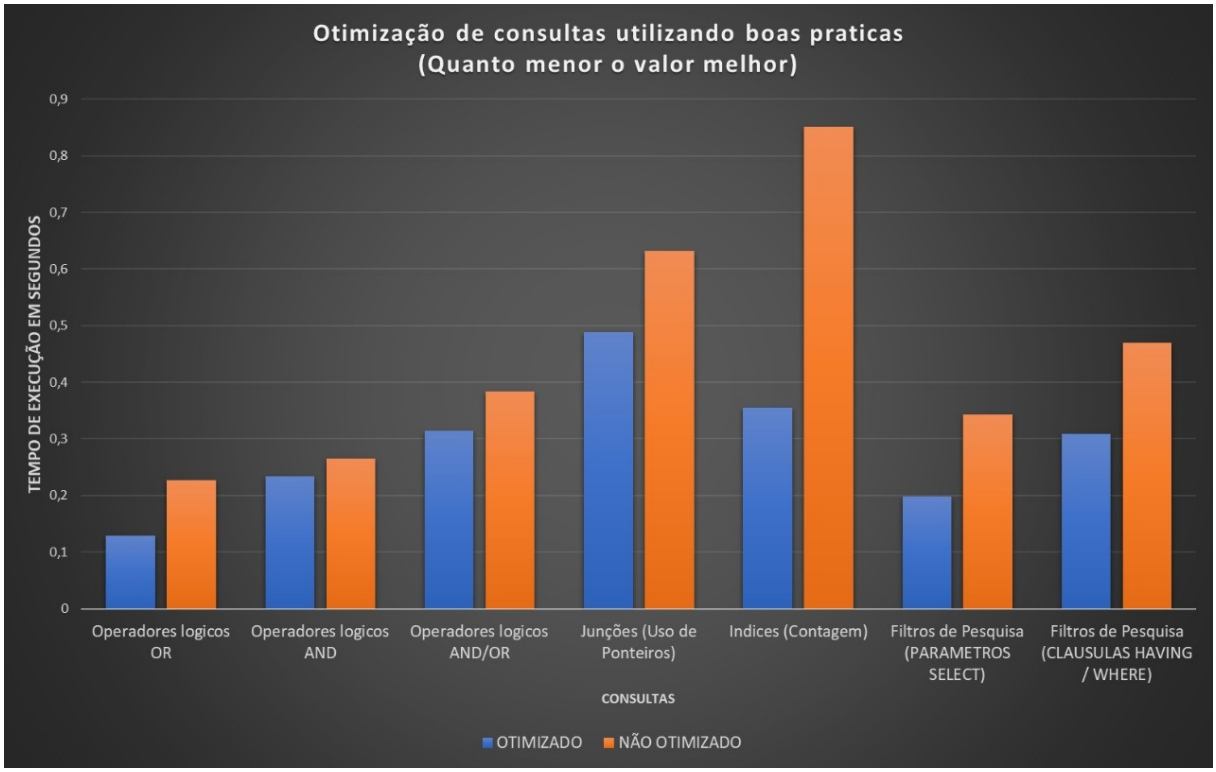


Figura 19: Gráfico de otimização utilizando boas práticas (autoria própria), 2021, 2021.

Nota se que ao aplicar boas na elaboração das consultas, temos um ganho significativo de desempenho na otimização do tempo de execução das mesmas, podendo refletir positivamente na maior vazão de dados no menor tempo possível.

Na análise da aplicação dos índices, foi usado apenas o exemplo da contagem, devido as outras boas práticas em relação ao mesmo, atentarem somente na consequência de desativação de seu uso, caso não seguidas, toda via tal diferença demonstra também o grande impacto que o mesmo pode causar, se não levadas em consideração as ressalvas mencionadas no estudo.

Embora em algumas medidas de desempenho, a diferença não tenha sido tão significativa, vale lembra que o banco de dados implementado, não consegue simular todas as situações de um ambiente de produção.

Por fim ao analisarmos o gráfico de forma geral, observasse que se aplicarmos as boas práticas desde o princípio, podemos entregar um bom desempenho aos clientes, sistemas e usuários, através de boas práticas simples de serem implementadas, e extremamente benéficas ao bom funcionamento de um banco de dados relacional.

6 CONCLUSÃO

A otimização de um banco de dados relacional, é uma tarefa complexa, que deve ser bem elaborado desde o seu projeto de ascensão, influenciando em toda sua estrutura, desde tecnologias que serão aplicadas, como por exemplo a nível de hardware como um dispositivo de armazenamento ou até mesmo a nível de software decidindo qual sistema gerenciador de banco de dados usar.

Por mais que se tenha boas ferramentas para se usar no decorrer do processo, deve se atentar a pequenos detalhes que possam acarretar grande impacto no futuro, como as simples elaborações de consultas ou até mesmo na elaboração de uma base de dados. Para facilitar o processo de otimização, podemos contar parcialmente com o sistema gerenciador de banco de dados, que possui processos integrados de otimização e facilitam a comunicação do administrador para com o banco de dados.

Toda via, o sistema gerenciador de banco de dados possui limitações, fazendo que seja necessária a intervenção manual, para a continuação do processo de otimização.

No processo de otimização manual há boas práticas que podem ser seguidas não somente após a implantação do SGBD, mas também desde a criação do banco de dados.

Tais boas práticas colaboram de forma significativa desde a construção do banco até na elaboração dos scripts, com elas podemos entender a um alto nível de abstração, a forma com que o banco de dados se comporta a cada operação, possibilitando o ajuste fino, que um SGBD estaria incapaz de realizar.

Percebe se que a elaboração de consultas pode possuir grande grau de relação com o desempenho de um banco de dados, do qual as mesmas devem ser devidamente analisadas e estudas durante sua elaboração.

Foi possível observar no desenvolvimento de algumas consultas, alguns recursos que podem ser ajustados de forma manual no SGBD, como o uso de memória cachem e o uso do paralelismo para processamento de dados, todavia tais ajustes dependem de um conhecimento profundo para cada tipo de SGBD, sendo interessante propor futuros estudos explorando estes recursos para otimização.

Por fim, fica claro o impacto que boas práticas abordadas podem fornecer, para o processo de otimização, embora os estudos tenham sido feitos em um ambiente local isolado, as mesmas práticas aplicadas a qualquer ambiente de produção de uso intensivo, que pode vir a replicar o seu ganho de desempenho, em uma escala maior comparada as que foram abordadas no estudo.

REFERÊNCIAS

ALMEIDA, AC. HAEUSLER. E. H. LIFSCHITZ, S. OLIVEIRA, R. P. SCHWABE, D. Outer-*Tuning*: Sintonia Fina automática baseada em ontologia. Rio de Janeiro: UERJ, PUC-Rio, 2018.

ALVES, G, 2017. Como criar um projeto de banco de dados. Disponível em : < https://dicasdeprogramacao.com.br/como-criar-um-projeto-de-banco-de-dados/ >Acesso em 21 jun 2021.

ARAUJO, M. 2017. Otimização de legibilidade e tempo de execução em consultas SQL no SGBD Oracle 11g. IMASTERS, 2017. Disponível em: < https://imasters.com.br/data/otimizacao-de-legibilidade-e-tempo-de-execucao-em-consultas-sql-no-sgbd-oracle-11g > Acesso em 10 set 2021.

BELEM, T, 2010. Planejamento e modelagem de um banco de dados para e-Commerce. Disponível em: < https://www.devmedia.com.br/planejamento-e-modelagem-de-um-banco-de-dados-para-e-commerce/17185 > Acesso em 21 jun 2021.

BERMAN, J. J. *Principles of Big Data: Preparing, Sharing, and Analyzing complex Information*. 2013.

BINI, T. A. Análise da Aplicabilidade das Regras de Ouro ao *Tuning* de Sistemas Gerenciadores de Bancos de Dados Relacionais em Ambientes de Computação em Nuvem. Curitiba: UFP, 2014.

BRITO, AC. A. LIFSCHITZ, S. (Auto) Sintonia-Fina em Sistemas de Bancos de Dados nas Organizações. Rio de Janeiro: UERJ, PUCRJ, 2018.

BRITO, F. O que é um HD. ago2021.ZOOM. Disponível em :<https://www.zoom.com.br/pc-computador/deumzoom/o-que-e-um-hd >Aceesso em 24 nov 2021.

CARNEIRO, A. P. MOREIRA, J. L. FREITAS, AL. C. *TUNING* – Tecnicas de Otimizaçao de Banco de Dados: Um estudo comparativo Mysql e Postgresql. Rio Grande do Sul: FURG, 2011.

DAMIN, D. Linguagem SQL: Capitulo 5 – Boas práticas de configuração de um Histórico de Consulta. Elipse Knowledgebase, 25 mar 2019. Disponível em: < https://kb.elipse.com.br/linguagem-sql-capitulo-5-boas-práticapráticas-de-configuracao-de-um-historico-e-uma-consulta/> Acesso em 12 fev 2021.

DOMINICO, S. *TUNING*: Um estudo sobre a otimização de desempenho de sistemas gerenciadores de banco de dados relacionais sob carga de trabalho de suporte a tomada de decisão. GUARAPUAVA: UTFPR, 2013.

GEHRKE, Johannes; RAMAKRISHNAN,Raghu. Sistemas de Gerenciamento de Banco de Dados. São Paulo: McGraw-Hill, 2008.

GOES, W. P. *Tuning* de Índices em sistemas gerenciadores de Banco de Dados Relacionais, utilizando Sistemas Classificadores. Curitiba: PUCPR, 2019.

GOUVEIA, R. Tabela Verdade. Toda Matéria, jan 2019. Disponível em: <https://www.todamateria.com.br/tabela-verdade/> Acesso em 25 de fev 2021.
Hazewinkel, Michiel, ed. (2001). "Teoria dos Erros." Enciclopédia de Matemática . Springer Science + Business Media BV / Kluwer Academic Publishers. ISBN 978-1-55608-010-4.

LEANDRO, L. SSD ou HD: Veja prós e contras de cada um e saiba qual usar no seu PC. Techtudo, 08 set 2020.
Disponível em: <https://www.techtudo.com.br/noticias/2020/09/ssd-ou-hd-veja-pros-e-contras-de-cada-um-e-saiba-qual-usar-no-seu-pc.ghtml> Acesso em 25 de fev 2021

LOPES, G. Otimização de Consultas SQL: Uma análise de Desempenho. Americana, SP, 2013. 51p. (Monografia de Analise e Desenvolvimento de Sistemas) – Fatec Americana, sp.

LUCIDCHART,2019. O que é um modelo de banco de dados. Disponível em: < https://www.lucidchart.com/pages/pt/o-que-e-um-modelo-de-banco-de-dados>
Acesso em 22 jun 2021.

OLIVEIRA, M. O que é SGBD: Vamos ver conceitos e dicas SQL. Terminal Root, 30 ago 2019. Disponível em: <https://terminalroot.com.br/2019/08/o-que-e-sgbd.html> Acesso em 12 fev 2021.

ORACLE, 2015. Oracle Database 12c: In-Memory (Parte I). Disponível em:< https://www.oracle.com/br/technical-resources/articles/database-performance/oracle-db-12c-in-memory.html > Acesso em 20 nov 2021.

PICHILIANI, M. Testando o desempenho do HD com banco de dados. 2016. Disponível em: < https://www.devmedia.com.br/testando-o-desempenho-do-hd-com-banco-de-dados/37429 > Acesso em 24 nove 2021

PIONTKOSKI, D. Comparativo De Desempenho De Consultas Sql Entre Banco De Dados Em Memória Ram E Em Ssd. Pato Branco, 2017. 36p. (Monograafia de especialização em Banco de Dados) - Universidade Tecnológica Federal do Paraná, campus Pato Branco.

PRASS, F. Otimização de Consultas SQL. DEVMEDIA, 2015. Disponível em: <https://www.devmedia.com.br/otimizacao-de-consultas-sql/33485 >Acesso em 12 fev 2021.

REIS, F. O que é um Banco de Dados Relacional. Boson treinamentos, 16 ago 2019. Disponível em: < http://www.bosontreinamentos.com.br/bancos-de-dados/o-que-e-um-banco-de-dados-relacional/> Acesso em 12 fev 2021.

RESENDE, D. Entendendo o funcionamento dos índices no SQL Server. [2015]. Disponível em: < https://www.dirceuresende.com/blog/entendendo-o-funcionamento-dos-indices-no-sql-server/. Acesso em: 20 nov 2021.> 2015.

RIGO, V. F. Estudo da metodologia de *Tuning* em Banco de Dados Oracle. Assis: IMESA, FEMA, 2012.

ROB, P. CORONEL, C. Sistemas de Banco de Dados: Projeto, Implementação e Gerenciamento. 2003.
Disponível em: < https://www.ime.usp.br/~andrers/aulas/bd2005-1/aula13.html#:~:text=Em%20SQL%2C%20nessa%20cl%C3%A1usula%20podem,igual%20a%20algum%20outro%20valor.> Acesso em 25 de fev 2021.

SALLES, F. O que é SSD e como ele funciona: Saiba tudo sobre esse componente. ago 2021. ZOOM. Disponível em:https://www.zoom.com.br/notebook/deumzoom/o-que-e-ssd Aceesso em 24 nov 2021.

SANCHES, A. Linguagem SQL. USP, 09 jun 2005.

SANTOS, J. C. SILVA, A. P. S. Otimização e Performance de Banco de Dados utilizando sql *tuning*. Paranavai: Unipar, 2013.

SCUDERO, E. TOP 10 principais SGBDs do mercado global. Becode, 2017. Disponível em: < https://becode.com.br/principais-sgbds/> Acesso em 12 fev 2021.

SEM AUTOR. O que é SGBD: O que é e o que você precisa saber. TiFlux, 2020. Disponível em: < https://www.tiflux.com.br/blog/sgbd-o-que-e-e-o-que-voce-precisa-saber/> Acesso em 12 fev 2021.

SILVA, D. Banco de Dados. [2015]. Disponível em: <www.estudopratico.com.br/banco-de-dados/. Acesso em: 20 mar 2021.>
SOUZA, AP. S. CAMPOS, B. F. DIAS, C. G. G. ALVES, M. B. VIEIRA, C. E. C. CARELLI, F. C. SÁ, L. F. C. *Tuning* em Banco de Dados. Volta Redonda: UniFOA, 2009.

STEEL, ROBERT GD; Torrie, James H. (1960). Princípios e Procedimentos de Estatística, com Referência Especial às Ciências Biológicas . McGraw-Hill.

VOLACO, E. B. Otimização de comandos SQL. Revista SQL Magazine. Out 2007. Ano 01. Edição 01. São Paulo: DevMedia. 2006. Disponível em: . Acesso em: 10 nov 2021.

APÊNDICE

PUC GOIÁS

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 • Setor Universitário
Caixa Postal 86 • CEP 74605-010
Goiânia • Goiás • Brasil
Fone: (62) 3946-3081
www.pucgoias.edu.br • reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Gabriel Tristão do Couto Mendonça
do Curso de Ciencias da Computação, matrícula 2018100280223,
telefone: 62996090660 e-mail gabriel.tristao@outlook.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Boas práticas de tuning de banco de dado para otimização de um banco de dados relacional, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto (PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 09 de dezembro de 2021.

Assinatura do(s) autor(es): _____

Nome completo do autor: Gabriel Tristão do Couto Mendonça

Assinatura do professor-orientador: _____

Nome completo do professor-orientador: Fabio Barbosa Rodrigues