**FACULDADES FACCAT**

**ADRIAN HIDEKI DOS SANTOS**

**SOFTWARE DE OTIMIZAÇÃO DE BANCO DE DADOS**

**TUPÃ**

**2020**

**ADRIAN HIDEKI DOS SANTOS**

**SOFTWARE DE OTIMIZAÇÃO DE BANCO DE DADOS**

Trabalho de conclusão de curso apresentado ao curso de Sistemas de Informação das Faculdades FACCAT, como requisito para o exame de defesa e obtenção do título de Bacharel em Sistemas de Informação. Orientador(a): Esp. Ewerton Silva.

**TUPÃ**

**2020**

**ADRIAN HIDEKI DOS SANTOS**

**SOFTWARE DE OTIMIZAÇÃO DE BANCO DE DADOS**

Trabalho de Conclusão de Curso apresentado à Faculdades FACCAT, como parte dos requisitos necessários para obtenção do título de Bacharel em Sistemas de Informação.

BANCA EXAMINADORA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof.º Adriano de Oliveira Cipriano

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof.º Ms. José Marcelo Pereira da Silva

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof.ª Dr.ª Patrícia da Silva Moreno e Souza (Orientadora)

**TUPÃ**

**2020**

**RESUMO**

**ABSTRACT**

**LISTA DE FIGURAS**

[Figura 1 – Tabelas de um Banco de Dados 13](#Figura!0|sequence)

[Figura 2 - Edgar Frank Codd 14](#Figura!1|sequence)

[Figura 3 - Página de dados SQL Server 22](#Figura!2|sequence)

[Figura 4 - Extensão de páginas 22](#Figura!3|sequence)

[Figura 5 - Sintaxe de criação de índices 24](#Figura!4|sequence)

**LISTA DE TABELAS**

[Tabela 1 - Exemplo de tabela antes da primeira forma 17](#Tabela!0|sequence)

[Tabela 2 - Exemplo de tabela com a primeira forma normal aplicada 17](#Tabela!1|sequence)

[Tabela 3 - Tabela de vendas antes da segunda forma 18](#Tabela!2|sequence)

[Tabela 4 - Tabela de Produtos segunda forma 18](#Tabela!3|sequence)

[Tabela 5 – Tabela de Venda x Produto 18](#Tabela!4|sequence)

**SUMÁRIO**

[1. Introdução 10](#__RefHeading___Toc1950_2792148385)

[1.1. Definição do problema 10](#__RefHeading___Toc1952_2792148385)

[1.2. Objetivos 11](#__RefHeading___Toc1954_2792148385)

[1.2.1. Objetivo Geral 11](#__RefHeading___Toc1956_2792148385)

[1.2.2. Objetivo Específico 11](#__RefHeading___Toc1958_2792148385)

[1.3. Justificativa 11](#__RefHeading___Toc1960_2792148385)

[1.4. Metodologia 12](#__RefHeading___Toc1962_2792148385)

[2. Fundamentação teórica 12](#__RefHeading___Toc1964_2792148385)

[2.1. Introdução ao banco de dados 12](#__RefHeading___Toc1966_2792148385)

[2.2. SGBD 15](#__RefHeading___Toc2126_2792148385)

[2.3. Melhores práticas de banco de dados 16](#__RefHeading___Toc1968_2792148385)

[2.4. Normalização de dados 16](#__RefHeading___Toc1970_2792148385)

[2.4.1. Primeira forma 17](#__RefHeading___Toc2128_2792148385)

[2.4.2. Segunda forma 18](#__RefHeading___Toc2130_2792148385)

[2.4.3. Terceira forma 19](#__RefHeading___Toc2132_2792148385)

[2.4.4. Quarta forma 19](#__RefHeading___Toc2134_2792148385)

[2.4.5. Quinta forma 20](#__RefHeading___Toc1972_2792148385)

[2.5. SQL Server 20](#__RefHeading___Toc1976_2792148385)

[2.6. Indexação de tabelas e estrutura de dados 21](#__RefHeading___Toc1978_2792148385)

[3. Desenvolvendo o projeto proposto 24](#__RefHeading___Toc1980_2792148385)

[3.1. Diagramas UML 24](#__RefHeading___Toc1982_2792148385)

[3.1.1. MER - Modelo de Entidade e Relacionamento 24](#__RefHeading___Toc1984_2792148385)

[3.1.2. Dicionário de dados 24](#__RefHeading___Toc1986_2792148385)

[3.1.3. Diagrama de Classes 25](#__RefHeading___Toc1988_2792148385)

[3.1.4. Descrição dos Casos de Uso 25](#__RefHeading___Toc1990_2792148385)

[3.1.4.1. … 25](#__RefHeading___Toc2136_2792148385)

[3.1.4.2. .... 25](#__RefHeading___Toc2138_2792148385)

[3.2. Requisitos não funcionais 25](#__RefHeading___Toc1992_2792148385)

[3.2.1. Manutenção 25](#__RefHeading___Toc1994_2792148385)

[3.2.2. Suporte 26](#__RefHeading___Toc1996_2792148385)

[3.2.3. Infraestrutura 26](#__RefHeading___Toc1998_2792148385)

[3.2.4. Segurança 26](#__RefHeading___Toc2000_2792148385)

[3.3. Métodos para controle de segurança do sistema 27](#__RefHeading___Toc2002_2792148385)

[3.3.1. Controle de Segurança Lógica 27](#__RefHeading___Toc2004_2792148385)

[3.3.2. Plano de Contingência 27](#__RefHeading___Toc2006_2792148385)

[3.4. Layout dos Relatórios 27](#__RefHeading___Toc2008_2792148385)

[3.4.1. Nome do relatório 27](#__RefHeading___Toc2010_2792148385)

[3.4.1.1. Tela do relatório 27](#__RefHeading___Toc2012_2792148385)

[3.4.1.2. Exemplo do relatório 27](#__RefHeading___Toc2014_2792148385)

[3.4.1.3. Instruções SQL 27](#__RefHeading___Toc2016_2792148385)

[3.4.1.4. Ferramenta utilizada para desenvolver e apresentar o relatório 27](#__RefHeading___Toc2018_2792148385)

[4. Implementações Futuras 27](#__RefHeading___Toc2020_2792148385)

[5. Conclusão 27](#__RefHeading___Toc2022_2792148385)

[6. Referências Bibliográficas 28](#__RefHeading___Toc2024_2792148385)

[7. Relatório de Participação dos alunos no estágio 29](#__RefHeading___Toc2026_2792148385)

# **1. I****ntrodução**

Atualmente muitas soluções possuem sua lógica de negócio completamente concentrada no banco de dados, ou seja, em regras de negócio e tecnologia, torna-se cada vez mais importante prezar pelo controle e performance do que acontece no banco de dados. Para que os clientes tenham maior fluidez ao utilizar o sistema e os usuários consigam fazer as operações normalmente com eficiência e praticidade.

Com os padrões definidos pela tecnologia, de melhores práticas, seja para estruturação e codificação, torna-se importante cada vez mais adequar os programas existentes para este padrão, para sistemas antigos acaba sendo necessário revisar os códigos existentes, gerando atividades que não seriam necessárias se fosse utilizado um padrão de desenvolvimento eficiente.

O SQL Server possui uma linguagem chamada T-SQL (Transact SQL), onde é possível de modo transacional, interagir com o banco de dados, fazendo consultas e atualizações no mesmo (MICROSOFT, 2020). Quando necessitamos processar as informações no banco de dados, é necessário utilizar das funções disponíveis na linguagem T-SQL, todavia deve sempre atentar-se com relação às operações no banco de dados que possuem filtros ou ordenações pois quando não existem índices e suporte uma consulta pode ficar muito lenta, consumindo recursos desnecessários do servidor.

Para a Microsoft (2020), os índices são objetos no banco de dados criados para otimizar operações no banco de dados para determinadas tabelas e visões, basicamente os índices ordenam os registros de uma tabela, a partir dos campos, ou seja, quando um índice é criado precisamos informar as colunas chave, criando o índice quando realizarmos uma consulta utilizando de filtros os campos do índices, onde temos um ganho de performance, pois não é necessário ler toda a tabela ou ordenar os dados para realizar a pesquisa.

## **1.1.** **Definição do problema**

A lentidão no banco de dados pode ser causada por diversos fatores, seja uma tabela muito grande, comandos utilizados de forma incorreta, a ausência de indexação ou a má indexação de uma e várias tabelas, onde podem afetar as funcionalidades do sistema tornando-as lentas impedindo o fluxo do usuário. Ainda, configurações indevidas, no banco, ou tabelas com uma grande volumetria, podem atrapalhar o funcionamento do ambiente, prejudicando todos os usuários do ambiente.

## **1.2.** **Objetivos**

### **1.2.1.** **Objetivo Geral**

Desenvolver um sistema de otimização de banco de dados, tornando possível gerar relatórios de pontos que possam ser melhorados no sistema, com relação ao banco de dados, sendo possível fazer análises de informações presentes na base e de rotinas sendo executadas em tempo real. A análise da indexação das tabelas e dos relacionamentos, assim como estatísticas que influenciam totalmente na performance de uma aplicação no banco de dados.

### **1.2.2.** **Objetivo Específico**

Identificar problemas que degradam a performance do banco de dados, criar e alterar índices em tabelas no banco de dados, de forma que facilite o trabalho de identificação de problemas e customização de índices na base de dados.

* Estudar e apresentar as melhores práticas de banco de dados;
* Desenvolver um sistema que automatize verificações realizadas manualmente;
* Exemplificar situações que podem afetar a performance de um sistema;

## **1.3.** **Justificativa**

Cada vez mais se torna necessário realizar análises nos servidores de banco de dados, devido a quantidade de itens que podem afetar o desempenho do servidor, por isso torna-se necessário economizar tempo e identificar o problema de forma automática e que o resultado seja compreendido, considerando sempre as melhores práticas de configuração e programação no banco de dados. Com isso, é possível evidenciar para os clientes os problemas que afetam o servidor e o sistema em si.

## **1.4.** **Metodologia**

Será consultada a documentação do produto SQL Server atualizada, assim como sites de profissionais nesta tecnologia para que seja sempre levando em consideração os melhores padrões de configuração e desenvolvimento. O software será desenvolvido em C#, utilizando os recursos mais atualizados do .net framework.

# **2.** **Fundamentação teórica**

## **2.1.** **Introdução ao banco de dados**

O banco de dados é um aplicativo que armazena os dados em formato de tabelas, contendo linhas e colunas. Quando se fala de banco de dados, temos que saber a diferença entre dados e informações, para Silva (2015):

Os dados são os fatos brutos, em sua forma primária, e podem não fazer nenhum sentido quando estão isolados; já as informações são o agrupamento de dados organizados, de forma que façam sentido e gerem algum conhecimento.

Um banco de dados pode ter diversas tabelas, assim como uma tabela pode ter várias linhas. Sempre que criamos uma tabela, temos o objetivo de armazenar um conjunto de dados sobre uma entidade específica, ou seja, se criarmos uma tabela de pessoas, criaremos campos que armazenam dados de uma pessoa, como nome, data de nascimento, peso. Segundo Silva (2015):

Uma das definições de banco de dados afirma que se trata de uma coleção de informações que se relacionam de modo que criem algum sentido, isto é, é uma estrutura bem organizada de dados que permite a extração de informações. Assim, são muito importantes para empresas e tornaram-se a principal peça dos sistemas de informação.

Com esta afirmação, conseguimos entender que através de um banco de dados é possível que sistemas processem informações, e que elas sejam utilizadas por uma organização ou indivíduo, por este motivo é importante ter uma estrutura muito bem organizada ao se criar um banco de dados, definindo as colunas, os tipos das colunas, as nomenclaturas e um dos itens mais importantes do banco de dados que é o relacionamento das tabelas (a figura 1 é uma representação visual da estrutura de uma tabela e seus relacionamentos), que garante a integridade do banco de dados e da regra de negócio da aplicação em si.

Figura 1 – Tabelas de um Banco de Dados

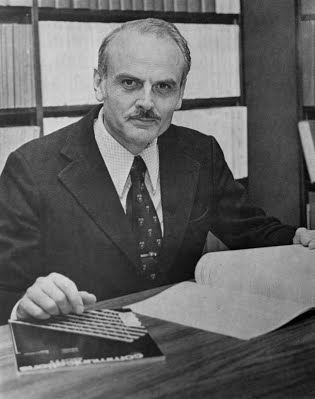


Fonte: Site Estudo Pratico. Disponível em <<https://www.estudopratico.com.br/banco-de-dados/>>.

Atualmente os bancos de dados são disponibilizados com recursos modernos que facilitam a criação, manutenção e monitoramento do banco de dados, todavia os bancos de dados nem sempre foram aplicações tão robustas, mas sim aplicações que evoluíram conforme as necessidades do mercado e da comunidade.

A história do banco de dados, conforme explica Rezende (2020), tem seu início com os softwares de armazenamento de informações (aplicações como Clipper, Dbase 2, Fox Pro, COBOL), os mesmos disponibilizavam métodos e funcionalidades para armazenar informações, selecionar e editar as mesmas. Para Rezende (2020), durante os anos de 1970 e 1972 surgiu o modelo de banco de dados relacional, proposto por Edgar Frank Codd (figura 2), isso fez com que o modo em que os dados são armazenados não estivessem relacionados com a estrutura lógica do banco de dados, criando-se assim uma modelagem que buscava relacionar entidades do banco de dados, evitando uma modelagem que possa conter erros ou dados duplicados em uma mesma tabela.

Figura 2 - Edgar Frank Codd



Fonte: Site Pioneiros. Disponível em <<https://sites.google.com/site/pioneiroscomputacao/p-1970-89/p61>>.

O banco de dados relacional proporcionou muitas melhorias em projetos de banco de dados, de forma que não ocorram redundâncias e campos desnecessários nas tabelas, ganhando armazenamento e melhorando a performance. Em 1976 o Dr. Peter Chen propôs o modelo de entidades e seus relacionamentos (conhecido atualmente como MER), sendo uma técnica de criação de projetos de banco de dados, capaz de descrever a estrutura inteira do banco e seus respectivos relacionamentos, sendo fácil, intuitivo e muito prático em casos de dúvidas na fase de desenvolvimento e manutenção do produto. Em 1980 segundo Rezende (2020):

A Linguagem Estruturada de Consulta – SQL (Structured Query Language) se torna um padrão mundial. A IBM transforma o DB2 como carro chefe da empresa em produtos para BD. Os modelos em rede e hierárquico passam a ficar em segundo plano praticamente sem desenvolvimentos utilizando seus conceitos, porém vários sistemas legados continuam em uso. O desenvolvimento do IBM PC desperta muitas empresas e produtos de BD como: RIM, RBASE 5000, PARADOX, OS/2 Database Manager, Dbase III e IV (mais tarde transformado em FoxBase e mais tarde ainda como Visual FoxPro), Watcom SQL, entre outros.

Em 1990, um modelo de aplicação começou a ganhar espaço no mercado, de acordo com Rezende (2020), “O modelo cliente-servidor (client-server) passa a ser uma regra para futuras decisões de negócio e vemos o desenvolvimento de ferramentas de produtividade como Excel/Access (Microsoft) e ODBC”.

Na metade dos anos 90, os processos de transação em tempo real atingem uma estabilidade com seu intenso uso em pontos de venda (REZENDE, 2020). A partir dos anos 2000, os sistemas gerenciadores de banco de dados evoluíram muito, tendo capacidade de armazenar grande quantidade de dados e aumentaram os recursos fornecidos para que se torne cada vez mais fácil trabalhar com as informações no banco de dados.

Atualmente os bancos de dados seguem com a tendência de não ser somente um repositório de dados, mas sim uma fonte de informações que torna-se possível extrair estatísticas e assim fornecer uma informação que seja possível utilizar em uma tomada de decisão, temos diversos bancos de dados que suportam capacidade superiores a Terabytes de informações, e tecnologias que conseguem processar as mesmas de acordo com as necessidades de negócio e níveis estratégicos da empresa (REZENDE, 2020). Recentemente as empresas têm investido muito em bancos de dados potentes, apostando em sistemas relacionais e não relacionais, visando sempre o melhor desempenho e praticidade em suas aplicações.

## **2.2. SGBD**

Os SGBD nada mais são do que os Sistemas de gerenciamento de banco de dados, os mesmos são aplicações que tem o intuito de gerenciar o banco de dados, sendo responsável por fazer os processamentos, armazenar, atualizar, apagar e selecionar informações no banco de dados que está sendo gerenciado, segundo Sanches (2005):

Os sistemas de banco de dados são projetados para gerenciar grandes grupos de informações. O gerenciamento de dados envolve a definição de estruturas para armazenamento de informação e o fornecimento de mecanismos para manipulá-las. Além disso, o sistema de banco de dados precisa fornecer segurança das informações armazenadas, caso o sistema dê problema, ou contra tentativas de acesso não-autorizado. Se os dados devem ser divididos entre diversos usuários, o sistema precisa evitar possíveis resultados anômalos [...] A importância das informações na maioria das organizações e o consequente valor dos bancos de dados têm orientado o desenvolvimento de um grande corpo de conceitos e técnicas para o gerenciamento eficiente dos dados.

O sistema gerenciador do banco de dados deve ser uma aplicação completa, onde seja possível garantir a integridade dos dados e que seja possível também realizar atividades de contingência, onde caso ocorra algum imprevisto, seja possível recuperar as informações, sendo um mecanismo conhecido pelo termo de *backup*.

## **2.3.** **Melhores práticas de banco de dados**

As boas práticas de programação existem também quando falamos de banco de dados, muitos programas de banco de dados possuem uma linguagem de programação onde é possível criar rotinas com validações lógicas, manipulação de dados e estruturas de repetição. Com todos estes recursos disponíveis em um banco de dados, deve-se sempre utilizar as melhores práticas de desenvolvimento voltado para banco de dados, principalmente na estruturação de tabelas, envolvendo princípios básicos de modelagem de banco de dados e indexação de dados.

## **2.4.** **Normalização de dados**

A normalização do banco de dados é uma das técnicas mais importantes utilizadas na modelagem do banco de dados e na criação de suas tabelas, com a utilização da mesma, temos diversos aproveitamentos do banco de dados em si, observando o entendimento do sistema, custo de armazenamento e otimização do mesmo, segundo Machado (2015):

Normalização é o processo de modelar o banco de dados projetando a forma como as informações serão armazenadas a fim de eliminar, ou pelo menos minimizar, a redundância no banco. Tal procedimento é feito a partir da identificação de uma anomalia em uma relação, decompondo-as em relações melhor estruturadas.

Normalmente precisamos remover uma ou mais colunas da tabela, dependendo da anomalia identificada e criar uma segunda tabela, obviamente com suas próprias chaves primárias e relacionarmos a primeira com a segunda para assim tentarmos evitar a redundância de informações.

Ao se utilizar a normalização de dados em um banco de dados, temos cinco formas para aplicar e assim normalizar um banco de dados, de forma simples, o objetivo da normalização do banco de dados é eliminar a duplicidade de informações e também evitar que informações não obrigatórias sejam armazenadas na mesma tabela, analisando todas as formas normais cada forma faz com que os dados sejam mais sintéticos e íntegros. Para considerar um banco de dados como normalizado não é necessário aplicar todas as formas normais, aplicando até a terceira forma já é possível considerar como normalizada (MARCHI, 2013).

### **2.4.1. Primeira forma**

A primeira forma consiste em remover campos que possuem mais de um valor informado. Nesse caso é necessário separar os as informações de um único campo em vários campos. Outro quesito da primeira forma normal é identificar a chave primária da tabela, sendo uma chave natural ou artificial (MELLO, 2016).

Tabela 1 - Exemplo de tabela antes da primeira forma

|  |  |  |
| --- | --- | --- |
| Código | Nome | Endereço |
| 10 | Adrian Hideki | Rua Cherentes, 60, CEP: 17600-000 |
| 11 | Alanis Mayumi | Rua Tapajós, 110, 17600-030 |
| 12 | Luciane Mitiko | Rua Carijós, 180, 17601-611 |
| 14 | Marcos Oliveira | Avenida Lélio Pizza, 180, 17605-655 |

Fonte: Autoria própria.

Tabela 2 - Exemplo de tabela com a primeira forma normal aplicada

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Código (PK) | Nome | Endereço | Número | CEP |
| 10 | Adrian Hideki | Rua Cherentes | 60 | 17600-000 |
| 11 | Alanis Mayumi | Rua Tapajós | 110 | 17600-030 |
| 12 | Luciane Mitiko | Rua Carijós | 180 | 17601-611 |
| 14 | Marcos Oliveira | Avenida Lélio Pizza | 180 | 17605-655 |

Fonte: Autoria própria.

Analisando as tabelas apresentadas, é possível identificar que o campo endereço (da tabela 1) foi separado em um total de três campos, sendo eles Endereço, Número e CEP (na tabela 2), mostrando assim uma tabela antes da primeira forma normal e após.

### **2.4.2. Segunda forma**

A segunda forma normal pode ser utilizada quando a tabela em si já está na primeira forma normal, tendo o objetivo de identificar campos que não dizem respeito a chave primária da tabela completamente, sendo necessário gerar uma nova tabela com esses dados (MELLO, 2016). Sendo assim podemos assumir um exemplo da tabela de vendas (tabela 3), onde temos o código da venda e do produto vendido, porém nem todos os campos dependem completamente da chave primária desta tabela:

Tabela 3 - Tabela de vendas antes da segunda forma

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Código Venda | Código produto | Nome Produto | Quantidade | Valor Produto |
| 1 | 1 | Caneca | 10 | 1,50 |
| 2 | 2 | Lápis | 30 | 0,80 |
| 3 | 3 | Borracha | 15 | 1,50 |

Fonte: Autoria própria.

Ao aplicar a segunda forma normal é gerada duas tabelas, uma de produtos (tabela 4) e outra de vendas x produtos (tabela 5):

Tabela 4 - Tabela de Produtos segunda forma

|  |  |  |
| --- | --- | --- |
| Código Produto (PK) | Nome Produto | Valor Produto |
| 1 | Caneca | 1.50 |
| 2 | Lápis | 0,80 |
| 3 | Borracha | 1,50 |

Fonte: Autoria Própria.

Tabela 5 – Tabela de Venda x Produto

|  |  |  |
| --- | --- | --- |
| Código Venda (PK) | Código Produto (FK, PK) | Quantidade |
| 1 | 1 | 10 |
| 2 | 2 | 30 |
| 3 | 3 | 15 |

Fonte: Autoria Própria.

Note que agora eu não tenho mais o nome do produto na tabela de vendas, fazendo com que todos os campos da tabela de vendas dependam da chave primaria nos campos Código Venda e Código Produto, sendo uma chave composta.

### **2.4.3. Terceira forma**

Segundo Mello (2016), para a terceira forma, deve-se “Identificar todos os atributos que são funcionalmente dependentes de outros atributos não chave”, nesse caso devemos remover esses campos.

### **2.4.4. Quarta forma**

Para Mello (2016), uma tabela estar na quarta forma normal, ela deve estar na terceira forma e não existir dependências multivaloradas. Nesse caso devemos dividir a tabela em várias, evitando assim redundância de dados. Abaixo conseguimos observar uma tabela antes (tabela 6) e depois da quarta forma normal (tabelas 7 e 8):

Tabela 6 – Tabela antes da quarta forma normal

|  |  |  |
| --- | --- | --- |
| Código Aula | Aluno | Sala |
| A001 | E001 | S001 |
| A001 | E002 | S001 |
| A001 | E001 | S002 |
| A001 | E002 | S002 |

Fonte: Autoria Própria.

Tabela 7 - Tabela de Aula x Alunos

|  |  |
| --- | --- |
| Código Aula | Aluno |
| A001 | E001 |
| A001 | E002 |

Fonte: Autoria própria.

Tabela 8 - Tabela de Aula x Sala

|  |  |
| --- | --- |
| Código Aula | Sala |
| A001 | S001 |
| A001 | S002 |

Fonte: Autoria própria.

Nesta forma normal, foram geradas duas tabelas, sendo uma com a relação das aulas e alunos, e na outra a relação das aulas com as salas, evitando assim duplicidade nos dados das tabelas, como era possível observar na tabela 6.

### **2.4.5.** **Quinta forma**

Para Marchi (2013), a quinta forma normal pode ser definida no seguinte conceito:

A 5FN ou Forma normal de Projeção-Junção baseia-se no conceito de dependência de junção. Dependência de junção expressa a capacidade de um tabela, que tenha sido decomposta em duas ou mais tabelas menores, de se reconstruir novamente a partir das novas tabelas, obtendo-se a tabela original. Logo, uma tabela está na 5FN quando não é mais possível decompô-la. Há um consenso que são raras as exceções que necessitam da passagem para a 5FN, na maioria dos casos, quando uma tabela está na 4FN também estará na 5FN.

Conforme é explicado por Marchi (2013), a quinta forma normal é raramente utilizada, basicamente é utilizada quando existem informações dependentes decompostas em tabelas separadas que precisam estar juntas em uma tabela.

## **2.5.** **SQL Server**

O SQL Server é um sistema gerenciador de banco de dados mais populares da atualmente, presente em diversas empresas ao redor do mundo, devido a sua grande comunidade técnica e facilidade de uso, é preferência em diversas empresas desenvolvedoras de software, porém sua primeira versão surgiu incialmente em 1988, sendo disponibilizado junto a versão do Windows NT (uma das versões iniciais do Windows) e posteriormente foi evoluindo como um produto separado, evoluindo recursos e funcionalidades novas em cada versão (PACIEVITCH, 2020).

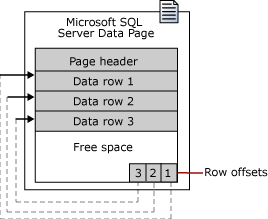
Os bancos de dados relacionais em geral, seguem um padrão de codificações que deve ser iguais para todos os bancos de dados, este é o padrão ANSI de programação, o SQL possui a linguagem de programação chamada de T-SQL (Transact SQL), que possui os comandos do padrão ANSI e além disso, funcionalidades que somente o SQL Server possui, como comandos de agregações e funções que não fazem parte do padrão do SQL Server.

Recentemente a Microsoft lançou a versão do SQL Server 2019, hoje o SQL Server além de ser um sistema gerenciador de banco de dados, o mesmo possui funcionalidades de atividades como processamento de massas de dados (conhecido pelo termo Big Data), suporte a Data Warehouse (conhecido por DW, uma forma de armazenamento e processamento de dados através de visões) e processos de BI (Business Inteligence), além de possuir sua versão online no Azure, uma plataforma digital de serviços, como máquinas virtuais e servidores de banco de dados SQL ou NoSQL (MICROSOFT, 2020).

## **2.6.** **Indexação de tabelas e estrutura de dados**

O SQL Server possui um mecanismo de armazenamento de informações em páginas de dados, segundo a documentação do Microsoft SQL Server (MSSQL) a página de dados é a unidade de armazenamento fundamental no armazenamento dos dados, qualquer informação gravada no banco de dados é gravado em páginas de dados, cada página de dados possui um tamanho de 8 quilobytes (ou 8KB). A estrutura da página de dados (representado visualmente pela Figura 3) pode ser dividida entre o cabeçalho, que contém algumas informações de metadados e linhas de dados.

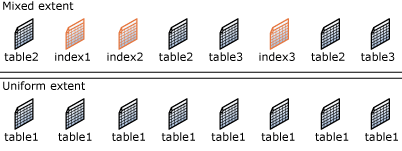
Figura 3 - Página de dados SQL Server



Fonte: Microsoft SQL Server Docs. Disponível em: <<https://docs.microsoft.com/pt-br/sql/relational-databases/pages-and-extents-architecture-guide?view=sql-server-ver15>>.

Entendendo como funcionam as páginas, é possível compreender as extensões, que nada mais é do que um conjunto de oito páginas de dados, segundo as documentações da Microsoft, as extensões (representada pela figura 4) podem ser classificadas em duas, uniformes e mixadas, onde as extensões uniformes possuem dados de somente uma tabela e extensões mixadas possuem páginas de diversas tabelas.

Figura 4 - Extensão de páginas



Fonte: Microsoft SQL Server Docs. Disponível em: <<https://docs.microsoft.com/pt-br/sql/relational-databases/pages-and-extents-architecture-guide?view=sql-server-ver15>>.

Quando vamos inserir alguma informação caso o SQL Server não tenha página de dados já criadas em branco é necessário no momento da inserção dos dados gerar estas páginas, isso envolve custos de processamento no servidor que o SQL Server está instalado, consumindo recursos de processador, disco e memória, por isso sempre é recomendado deixar páginas de dados já criadas no banco de dados, evitando assim lentidões em inserções massivas no sistema.

Os índices são objetos essenciais para o funcionamento de um banco de dados de sistemas online, este objetivo tem o objetivo de otimizar consultas realizadas no banco de dados, por isso a ausência de índices eficazes em um banco de dados pode causar gargalos no servidor e acabar afetando o tempo de resposta da consulta. Segundo a documentação da MSSQL (Microsoft SQL Server 2020), os índices devem ser criados de forma que ofereçam suporte a consultas complexa, contendo filtros e ordenação de dados na mesma, no SQL Server existem diversos tipos de índices:

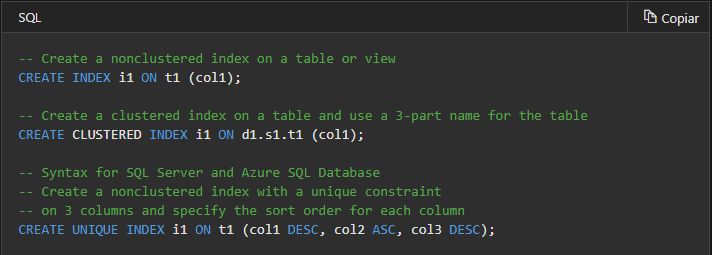
* Clusterizado
* Não clusterizado
* Exclusivo
* Filtrado
* Columnstore
* Hash
* Não clusterizado com otimização em memória

Os índices nada mais são do que cópias dos dados de uma tabela em um objeto interno do SQL Server, com o intuito de otimizar a leitura dos dados de uma determinada tabela, pense no índice como se fosse o sumário e a tabela um livro, para a Microsoft (2019) um índice pode ser definido como:

Um índice é uma estrutura em disco associada a uma tabela ou exibição, que agiliza a recuperação das linhas de uma tabela ou exibição. Um índice contém chaves criadas de uma ou mais colunas da tabela ou exibição. Essas chaves são armazenadas em uma estrutura (árvore B) que habilita o SQL Server a localizar a linha ou as linhas associadas aos valores de chave de forma rápida e eficaz.

Uma tabela pode ter vários índices, todavia deve-se tomar cuidado para não criar muitos, pois acaba aumentando o tamanho da tabela e aumenta o tempo das operações de inserção e atualização de dados, pois é necessário replicar as alterações nos índices criados, ou seja, um índice desnecessário pode prejudicar a performance de uma consulta (RODRIGUES, 2020). Para criar um índice utilizando o SQL Server, observe a sintaxe do SQL Server demonstrada na figura 5:

Figura 5 - Sintaxe de criação de índices



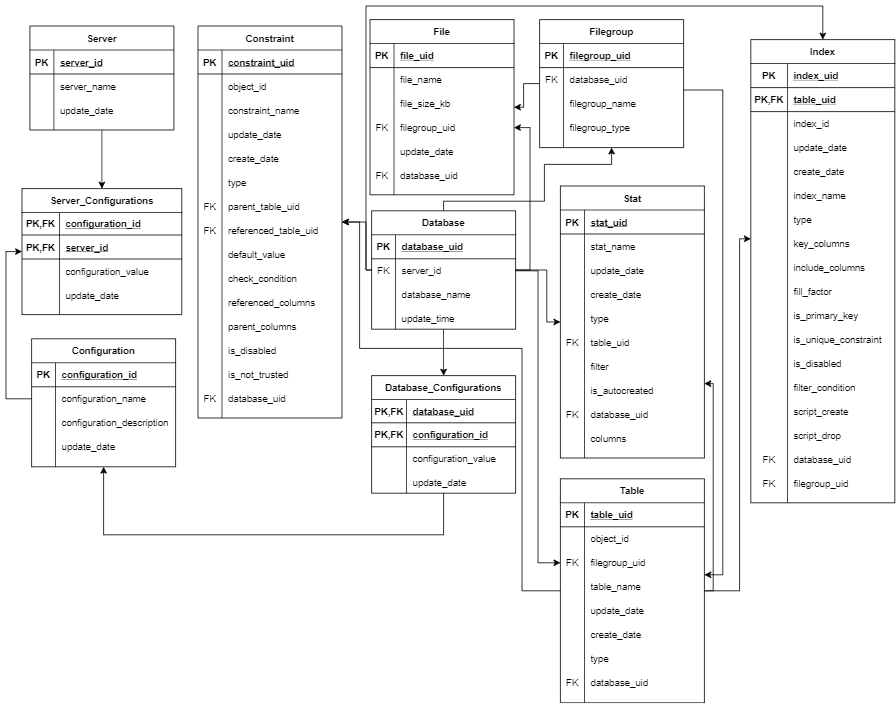
Fonte: Microsoft SQL Server Docs. Disponível em: < <https://docs.microsoft.com/pt-br/sql/t-sql/statements/create-index-transact-sql?view=sql-server-ver15> >.

Note que após o comando “CREATE” é colocado o tipo do índice, quando o tipo do índice não é informado o SQL Server assume que o mesmo é não clusterizado (NONCLUSTERED), após o tipo do índice é necessário escrever “INDEX”, declarar o nome do índice, seguido por “ON” e o nome da tabela, observe que é necessário informar as colunas e a ordenação das mesmas (ordem ascendente ou descendente), normalmente as colunas informadas no índice são as que sofrem filtragem ou ordenação de dados.

# **3.** **Desenvolvendo o projeto proposto**

## **3.1.** **Diagramas UML**

### **3.1.1.** **MER - Modelo de Entidade e Relacionamento**



### **3.1.2.** **Dicionário de dados**

### **3.1.3.** **Diagrama de Classes**

### **3.1.4.** **Descrição dos Casos de Uso**

#### **3.1.4.1. …**

#### **3.1.4.2. ....**

## **3.2.** **Requisitos não funcionais**

* O sistema operacional deve ser Windows 8.1 e superior;
* O banco de dados da aplicação deve ser SQL Server;
* Os bancos de dados que serão analisados e o servidor de banco deve ser SQL Server;
* O usuário do SQL Server que será responsável por gerar os relatórios deve ter permissão *dbowner*;
* O sistema deverá entregar as informações via relatório ou tela;
* O processamento será cliente-servidor;
* As informações coletadas podem ser armazenadas no banco de dados da aplicação;
* O sistema será desenvolvido em C#;
* O acesso ao sistema será através de um software instalado localmente no terminal;
* O armazenamento das informações será realizado de forma híbrida, onde parte das informações serão salvas no disco rígido e outra parte no banco de dados;

### **3.2.1.** **Manutenção**

A manutenção dos computadores será realizada de forma periódica conforme o padrão organizacional do usuário, todavia é recomendado que seja feita uma manutenção nos equipamentos onde o software está instalado, de forma que seja otimizado o espaço em disco através da limpeza de arquivos temporários, assim como a limpeza física dos equipamentos, evitando a depreciação dos materiais. A manutenção no sistema será feita de forma de solicitações dos clientes e entregas, onde será feito o suporte tendo em busca melhorias e correções, a cada mês será feita uma atualização de release com todas as melhorias e correções realizadas, buscando sempre incentivar o usuário a manter-se atualizado.

### **3.2.2.** **Suporte**

O suporte será fornecido através de interações por e-mails do cliente e pelo repositório da plataforma, onde as mensagens serão analisadas e respondidas, caso seja uma dúvida ainda não presente nas documentações, a mesma será adicionada e será indicado ao cliente onde consultar, caso seja um erro, o mesmo será corrigido e disponibilizado na próxima release (sendo possível gerar uma versão específica para um determinado erro, dependendo da gravidade do mesmo), caso seja enviado uma sugestão de melhoria, a implementação da mesma será analisada e disponibilizada na próxima release caso seja pertinente ao software.

### **3.2.3.** **Infraestrutura**

Os dados do aplicativo serão salvos de forma híbrida, de forma que as informações sejam armazenadas no terminal onde o software foi instalado e parte das demais informações no banco de dados. Será necessário criar um banco de dados específico para a aplicação, no mesmo servidor dos bancos de dados que serão analisados pela ferramenta. O servidor que será analisado precisará ter o SQL Server instalado, sendo necessário ter a versão 2008 ou superior. Os requisitos mínimos para a instalação do software é 4 GB de memória RAM, 1 GB de espaço livre em disco, o sistema operacional deve ser Windows 8.1 ou superior, sendo necessário a instalação dotnet framework 4.7.2 ou superior.

### **3.2.4.** **Segurança**

Deverá ser programado uma rotina automática de backup no banco de dados, para salvar os dados da aplicação, a periodicidade deverá ser definida pelo padrão da organização, sendo o recomendado de no mínimo de uma semana e no máximo de um dia. O aplicativo deve ser instalado no diretório local da aplicação, evitando salvar o aplicativo na rede, que pode gerar acesso indevido ao aplicativo. O servidor de banco de dados pode ser tanto local quanto hospedado em algum servidor, pois o aplicativo está analisando as bases do servidor que a aplicação for configurada.

## **3.3.** **Métodos para controle de segurança do sistema**

### **3.3.1.** **Controle de Segurança Lógica**

### **3.3.2.** **Plano de Contingência**

## **3.4.** **Layout dos Relatórios**

### **3.4.1.** **Nome do relatório**

#### **3.4.1.1.** **Tela do relatório**

#### **3.4.1.2.** **Exemplo do relatório**

#### **3.4.1.3.** **Instruções SQL**

#### **3.4.1.4.** **Ferramenta utilizada para desenvolver e apresentar o relatório**

# **4.** **Implementações Futuras**

# **5.** **Conclusão**

# **6.** **Referências Bibliográficas**

ALVES, G. O QUE É UM BANCO DE DADOS? Acessado em 13 de maio de 2020. Disponível em: <<https://dicasdeprogramacao.com.br/o-que-e-um-banco-de-dados/>>.

MACHADO, D. NORMALIZAÇÃO EM BANCO DE DADOS. Acessado em 8 de abril de 2020. Disponível em: <https://medium.com/@diegobmachado/normaliza%C3%A7%C3%A3o-em-banco-de-dados-5647cdf84a12>.

MARCHI, K. NORMALIZAÇÃO. Acessado em 13 de maio de 2020. Disponível em: <<http://kessiamarchi.blogspot.com/2013/10/normalizacao.html>>.

MELLO, I. MODELAGEM DE DADOS: FORMAS NORMAIS. Acessado em 26 de maio de 2020. Disponível em: <<http://www.consultoriadba.com/post/2016/05/10/modelagem-de-dados-formas-normais>>.

MICROSOFT. CREATE INDEX (TRANSACT-SQL). Acessado em 22 de junho de 2020. Disponível em: < <https://docs.microsoft.com/pt-br/sql/t-sql/statements/create-index-transact-sql?view=sql-server-ver15>>.

MICROSOFT. GUIA DE ARQUITETURA E DESIGN DE ÍNDICES DO SQL SERVER. Acessado em 6 de fevereiro de 2020. Disponível em: <<https://docs.microsoft.com/pt-br/sql/relational-databases/sql-server-index-design-guide?view=sql-server-ver15>>.

MICROSOFT. Guia de arquitetura de página e extensões. Acessado em 15 de junho de 2020. Disponível em: <<https://docs.microsoft.com/pt-br/sql/relational-databases/pages-and-extents-architecture-guide?view=sql-server-ver15>>.

MICROSOFT. Índices clusterizados e não clusterizados descritos. Acessado em 7 de julho de 2020. Disponível em: < <https://docs.microsoft.com/pt-br/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described?view=sql-server-2017>>.

MICROSOFT. SQL SERVER 2019. Acessado em 13 de junho de 2020. Disponível em <<https://www.microsoft.com/pt-br/sql-server/sql-server-2019>>.

PACIEVITCH, Y. SQL SERVER. Acessado em 13 de junho de 2020. Disponível em: <[https://www.infoescola.com/informatica/sql-server/#:~:text=O%20SQL%20Server%20%C3%A9%20um,melhorar%20o%20programa%20ap%C3%B3s%20isto.](https://www.infoescola.com/informatica/sql-server/" \l ":~:text=O SQL Server é um,melhorar o programa após isto.)>.

REZENDE, R. A HISTÓRIA DOS BANCO DE DADOS. Acessado em 3 de março de 2020. Disponível em: <<https://www.devmedia.com.br/a-historia-dos-banco-de-dados/1678>>.

RODRIGUES, J. Índices MySQL: Otimização de consultas. Acessado em 7 de julho de 2020. Disponível em: < <http://www.linhadecodigo.com.br/artigo/3620/indices-mysql-otimizacao-de-consultas.aspx>>.

SANCHES, A. R. DISCIPLINA: FUNDAMENTOS DE ARMAZENAMENTO E MANIPULAÇÃO DE DADOS. Acessado em 8 de abril de 2020. Disponível em: <<https://www.ime.usp.br/~andrers/aulas/bd2005-1/aula3.html>>.

SILVA, D. TECNOLOGIA - BANCO DE DADOS. Acessado em 20 de maio de 2020. Disponível em: <<https://www.estudopratico.com.br/banco-de-dados/>>.

# **7.** **Relatório de Participação dos alunos no estágio**