# APS Técnicas de Implementação de Banco de Dados

**Nome:** Carlos Eduardo Serpa Brito
**Matrícula:** 2020118290

# Introdução

Os Bancos de Dados nada mais são do que sistemas que agrupam um grande volume de informações relacionadas entre si, armazenando e gerenciando esses dados, essenciais para o funcionamento de diversas empresas mundialmente. Atualmente, possuímos diversos tipos diferentes presentes no mercado, nos quais, são divididos em bancos relacionais e bancos não relacionais.

Seguindo essa linha de raciocínio, irei apresentar nesse artigo, um estudo comparativo utilizando dois exemplos de bancos, um relacional (SQL Server) e outro não relacional (CASSANDRA), apresentando como esses diferentes bancos se comportam, quais são suas diferenças, como são realizados os processos, vantagens e desvantagens, dentre outros tópicos.

- **SQL Server:** O SQL Server é um banco de dados relacional pertencente a Microsoft, ele surgiu no ano de 1988, em parceria com a Sybase, atualmente um dos bancos de dados mais utilizados no mundo.

- **CASSANDRA:** O CASSANDRA é um banco de dados não relacional, inicialmente desenvolvido pelo Facebook em 2008, mas atualmente é mantido pele fundação Apache e outros colaboradores.

# Metodologia

Para o desenvolvimento desse artigo, foi realizado uma pesquisa por meio de artigos científicos e matérias presentes em sites referenciados no final do documento, assim como, a utilização prática dos softwares de gerenciamento de banco de dados, como o SQL Server 2019 e o Apache Cassandra 3.11.

# Desenvolvimento

Antes de começarmos nosso estudo comparativo entre os dois BD's, é importante compreendermos o que são os bancos relacionais e os não relacionais;

- Modelo Relacional: O BD relacional representa o modelo de dados proposto por E. F. Codd em 1970, onde a estrutura de armazenamento de dados lógicos utiliza tabelas, colunas, linhas de dados relacionados entre si. No modelo relacional, cada linha possui uma ID exclusiva chamada de chave, e as colunas possuem atributos dos dados, o que facilita o estabelecimento das relações entre os dados.

- Modelo Não Relacional: Também conhecido como bancos de dados NoSQL, ou seja, não utilizam consultas SQL (em teoria, porém ainda sim, existem bancos com suportes a consultas SQL), o modelo não relacional, representa um banco de dados que, diferentemente do relacional, não utiliza o método de tabelas, linhas e colunas, presentes em grande parte dos sistemas tradicionais de BD, no lugar disso, o BD não relacional utiliza um modelo de armazenamento otimizado e específico para o tipo de dados que está sendo armazenado.

- Por mais que, não iremos aprofundar esse assunto neste artigo, ainda é valido ressaltar, que o modelo não relacional também pode se dividir em 4 diferentes tipos; Key-value stores, Graph stores, Column stores e Document stores. O Banco de dados CASSANDRA é do tipo Key-value stores, ou Chave-Valor, onde os valores são armazenados e indexados por meio de uma chave.

Entendido os modelos de bancos de dados, agora podemos dar início ao nosso estudo comparativo entre o banco SQL Server e o banco Cassandra.

**- Gerenciamento de Buffer:** Tendo em vista que, o BD SQL Server tem como sua principal finalidade o armazenamento e a recuperação de dados, dessa forma, a E/S intensa de disco requerem muitos recursos e tempo para conclusão, logo, o SQL Server busca em tornar essas E/S o mais

eficiente possível, e para alcançar essa eficiência o gerenciamento de buffer é essencial.

O Gerenciamento do buffer dentro do **SQL Server** ocorre da seguinte forma, o componente de gerenciamento de buffer é dividido em dois mecanismos: o gerenciador de buffer para acessar e atualizar páginas de banco de dados e o cache do buffer (também chamado de pool de buffers), para reduzir a E/S do arquivo de banco de dados. Um buffer é uma página de 8 KB da memória, mesmo tamanho de uma página de dados ou de índice. Portanto, o cache do buffer é dividido em páginas de 8 KB.

Além disso, o gerenciador de buffer fornece contadores para realizar a monitoração de como os **SQL Server** utiliza os seguintes recursos:

- Memória para armazenar páginas de dados.
- Contadores para monitorar a E/S física, como leituras e gravações das páginas do banco de dados do SQL Server .
- Extensão do pool de buffers para estender o cache do buffer usando o armazenamento rápido não volátil, como SSD (unidade de estado sólido).

Esse monitoramento auxilia a determinar se está existindo gargalo, se o desempenho pode ser melhorado, a frequência com que o SQL Server precisa ler dados a partir do disco, dentre outros parâmetros.

Já dentro do **CASSANDRA**, o sistema aloca escritas em buffer fazendo com que as escritas sejam sempre uma operação sequencial, isso com várias operações de entrada e saída em disco acontecendo ao mesmo tempo. Esse fluxo faz com que a performance da tecnologia Cassandra seja tão alta.

**- Processamento de Consultas:** As consultas dentro do assunto de bancos de dados, nada mais são do que instruções, também podemos considerar como comandos, utilizado para realizar a recuperação de algum determinado dado que está presente dentro do banco. Essas instruções

são processadas dentro do sistema, após isso é realizado uma análise léxica/sintática e uma validação, e se tudo ocorrer corretamente, o bloco de comandos vai para um plano de execução, o qual pode ser executado diretamente (modo interpretado), ou armazenado e executado em outro possível momento (modo compilado), e por último é retornado a única coisa que importa para o usuário, a resposta.

Dentro do **SQL Server**, a realização dessas consultas é feita por meio da linguagem de consultas estruturada ou Structured Query Language (SQL), utilizando em grande parte das vezes o comando “SELECT”. O SQL possui diversos subconjuntos são eles:

- DML: Linguagem de Manipulação de Dados.
- DML: Linguagem de Manipulação de Dados.
- DDL: Linguagem de Definição de Dados.
- DCL: Linguagem de Controle de Dados.
- DTL: Linguagem de Transação de Dados.
- DQL: Linguagem de Consulta de Dados.

O **SQL Server** possui dois tipos diferentes para realizar a execução desses comandos:

- Execução em modo de linha: o processamento de consultas é realizado sequencialmente uma linha após a outra.
- Execução em modo de lote: método para processas várias linhas simultaneamente.

Já dentro do **CASSANDRA**, essa realização de instruções é por meio da linguagem CQL ou Cassandra Query Language, semelhante ao SQL, essa linguagem também é utilizada para interagir com o banco, tendo seus comandos divididos nas seguintes categorias:

- Definição das Estruturas de Dados.
- Manipulação dos Dados.
- Consultas.

**- Transações:** A transação dentro do âmbito de BD, nada mais é do que uma unidade de trabalho que foi ou será executada dentro do sistema do banco de dados. Se uma determinada transação tiver êxito (Commit Transaction), ou seja, for concluída, todas as alterações de dados feitas durante a transação, estarão confirmadas e aplicadas permanentemente dentro do banco de dados. Se uma transação encontrar erros (Rollback Transaction) e precisar ser cancelada, ou seja, não for concluída, todas as suas alterações de dados serão apagadas.

Dentro do SQL Server, existem as seguintes transações:

- Transações de confirmação automática: Cada instrução individual é uma transação.

- Transações explícitas: Cada transação é iniciada explicitamente com a instrução BEGIN TRANSACTION e finalizada explicitamente com uma instrução COMMIT ou ROLLBACK.

- Transações implícitas: Uma transação nova é iniciada implicitamente quando a transação anterior é concluída, mas cada transação é explicitamente concluída com uma instrução COMMIT ou ROLLBACK.

- Transações de escopo de lote: Aplicável apenas a MARS (Conjuntos de Resultados Ativos Múltiplos), a transação Transact-SQL explícita ou implícita iniciada em uma sessão MARS se torna uma transação de escopo de lote. A transação de escopo de lote que não é confirmada ou revertida quando um lote é concluído, será revertida automaticamente pelo SQL Server.

No **CASSANDRA**, diferentemente do SQL Server, não existe transações, é utilizado a persistência atômica, executar um bloco de N comandos de uma só vez: com isso, os dados só serão persistidos se todos os comandos forem executados com sucesso.

**- Controle de Concorrência:** O Controle de concorrência é quando, dentro de um BD, diferentes pessoas (usuários) tenta acessar o mesmo dado que está presente no banco, dessa forma, é realizado um controle entre essas transações, para garantir que elas sejam executadas de forma segura e sigam as regras ACID (atômico, consistente, isolado e durável).

No **SQL Server**, a realização desse controle, é por meio do método de bloqueio, o protocolo utilizado pode ser alterado e configurado pelo usuário de acordo com o que ele deseja, podendo por exemplo, indicar o grau de isolamento desejado entre transações, definindo como será feito o bloqueio dos dados.

Graus de Isolamento:

- SERIALIZABLE: Só permite que escalas serializáveis sejam executadas. Bloqueia os dados até o final da transação, e impede a inserção de novas tuplas nas tabelas em uso. É o modo mais restritivo, ou seja, o que resulta em menor concorrência.
- REPEATABLE READ: Bloqueia os dados até o final da transação, garantido que os valores dos dados acessados não serão modificados por outras transações, mas permite a inserção de novas tuplas na tabela, podendo originar "tuplas fantasma"
- READ COMMITED: Bloqueios são usados durante o acesso ao dado. Com isso, os valores dos dados acessados podem ser modificados por outras transações, e podem surgir "tuplas fantasma". É a opção default.
- READ UNCOMMITED: Não usa bloqueio, podendo resultar em leitura de dados sendo escritos (dirty read), valores dos dados acessados podem mudar durante a transação, e tuplas podem ser inseridas ou removidas durante a transação. É o modo menos restritivo, que resulta em maior concorrência, mas que deve ser usado somente quando não se tem nenhum compromisso com a consistência do resultado.

Deadlocks: Os deadlocks são quando o Banco de Dados percebe que existem dois processos um esperando pelo outro para iniciarem suas atividades, gerando uma situação de bloqueio, onde nenhum dos dois processos podem prosseguir.

O método de bloqueio do **SQL Server** não impede que esses deadlocks ocorram, entretanto ele detecta um deadlock automaticamente e escolhe a transação que está a mais tempo esperando para iniciar.

Dentro do **CASSANDRA**, como comentado antes, a presença de transações é praticamente inexistente, consequentemente o processo de controle de concorrência também é inexistente.

**- Benefícios e Cenários:**

Vantagens **SQL Server**:

- Software de gestão de alto nível.

- Excelente suporte para recuperação de dados.

Desvantagens **SQL Server**:

**-** Opções de licenciamento são muito caras, ou seja, custo elevado.

- Compatibilidade Limitada, o SQL Server é projetado para ser executado em apenas servidores Windows.

- Usabilidade, o SQL Server utiliza uma linguagem principal, a qual se difere das de aplicações de outros bancos relacionais como o MySQL e o Oracle.

Vantagens **CASSANDRA**:

- Alta disponibilidade.

- Performance.

- Extremamente tolerante a falhas.

- Altamente distribuído

-Suporta N datacenters nativamente.

Desvantagens **CASSANDRA**:

- Não irão resolver problemas de escalabilidade de um Website.

- Baixo suporte técnico.

- Difícil instalação e esforço para manutenção.

- Pouca mão de obra com conhecimento em NoSQL.

Quando não utilizar **CASSANDRA**?

- Se precisar de bastante consistência, a aplicação terá que garantir.

- Se o volume de dados ou o throughput da aplicação for muito pequeno.

- É necessário analisar se o modelo da aplicação suporta o paradigma colunar.

Limitações do **CASSANDRA**:

- Em bancos de dados não relacionais como o CASSANDRA, não há junções como em bancos de dados relacionais. Ou seja, é necessário executar várias consultas e unir os dados manualmente em seu código, o que pode gerar sérios problemas.

- Como o CASSANDRA não trata de forma automática as operações como transações da mesma maneira que um banco como o SQL Server, é necessário escolher manualmente criando uma transação e em seguida, verificar manualmente, confirmar ou retroceder manualmente.

# Conclusão

Dessa forma, concluímos que, não necessariamente um desses dois bancos, sejam eles bancos relacionais ou não relacionais, são melhores um que o outro, ambos possuem suas características, vantagens e desvantagens, e para a escolha de qual banco utilizar, e necessário levarmos em conta todo um contexto de utilização, viabilidade, circunstância e requisitos, para dessa forma, compararmos, e tomarmos a decisão de qual banco será melhor para a situação em que se encontra.

# Referências


- https://docs.microsoft.com/pt-br/sql/t-sql

https://bdm.unb.br/bitstream/10483/9040/1/2014_LizaneAlvaresLeite.pdf

- https://pt.wikipedia.org/wiki/Microsoft_SQL_Server

- https://rockcontent.com/br/blog/banco-de-dados/

- https://docs.microsoft.com/pt-br/azure/architecture/data-guide/big-data/non-relational-data

- https://www.oracle.com/br/database/what-is-a-relational-database/

- https://cic.unb.br/~alchieri/disciplinas/posgraduacao/sd/artigoG2.pdf

- https://www.geeksforgeeks.org/difference-between-ms-sql-server-and-cassandra/

- https://db-engines.com/en/system/Cassandra%3BMicrosoft+SQL+Server

- https://medium.com/nstech/apache-cassandra-8250e9f30942

http://www.repositorio.ufc.br/bitstream/riufc/25218/1/2014_tcc_sclima.pdf

- https://ericagallindo.com.br/lib/exe/fetch.php?media=disciplinas:bdii-20182:apresentacao_bdii.pdf