UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Desenvolvendo Soluções de Business Intelligence para Empresas Sergipanas

Trabalho de Conclusão de Curso

Felipe Morais Aragão

Departamento de Computação/UFS

São Cristóvão – Sergipe

2019

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Felipe Morais Aragão

Desenvolvendo Soluções de Business Intelligence para Empresas Sergipanas

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Gilton José Ferreira da Silva
Coorientador(a): Methanias Colaço Júnior

São Cristóvão – Sergipe

2019

Dedico este trabalho à minha família, especialmente aos meus pais Isabel e Aércio, aos meus irmãos Izabela e Aércio Junior, aos meus tios Adeilson e Neuma, e aos meus colegas de trabalho.

Agradecimentos

Agradeço primeiramente aos meus pais Isabel e Aércio por sempre acreditarem e investirem em mim, me apoiando nos momentos difíceis. Sem eles eu não teria chegado até esta etapa da minha vida.

Aos demais familiares, especialmente aos meus tios Adeilson e Neuma, pelo apoio e incentivo.

Aos meus amigos, especialmente a Sara e Larissa, que sempre me ajudaram e incentivaram em vários momentos.

Aos professores que contribuíram para minha formação, em especial a Gilton José e Methanias Colaço, respectivamente orientador e coorientador, que tiveram papel fundamental na elaboração deste trabalho. Agradeço também pela paciência e conhecimentos passados.

E a todos que direta ou indiretamente fizeram parte da minha formação.

Às vezes, ao inovar, você comete erros.
É melhor reconhecê-los logo e
partir para melhorar suas outras inovações.
(Steve Jobs)

Resumo

A necessidade de informações em tempo real para ajudar à tomada de decisão na atualidade é cada vez mais recorrente, uma característica presente nas empresas que elaboram essas soluções é a não distinção entre o ambiente OLTP *(On-Line Transaction Process)* e OLAP (*On-Line Analytical Process*), desta forma a satisfação do usuário pode decair com decorrer dos anos de uso. Este trabalho tem como objetivo construir uma arquitetura para dar suporte as consultas analíticas em tempo real, integrado por meio do processo de Extração, Transformação e Carga (ETL) em conjunto com o Change Data Capture (CDC) para cargas contínuas. Informações foram pesquisadas no mercado sergipano para obter detalhes específicos sobre o uso dos seus *workloads*, onde foi constatado que trinta e seis porcento das empresas pesquisadas utilizam o sistema de gerenciamento de banco de dados Microsoft SQL Server, além disso uma empresa foi selecionada e nela foram efetuadas análises de código *Structured Query Language* (SQL), entrevistas e levantamento de requisitos. Os resultados alcançados foram satisfatórios e a grande maioria dos aspectos comparados demonstraram um melhor uso de processador, memória primária e disco. Logo, o problema de desempenho situado na empresa foi solucionado e também com a capacidade prover dados em tempo real. desta maneira o CDC é uma forma viável de atualizar informações em um DW com pouca latência porque fornece um mecanismo de simples implementação para o rastreamento dos dados, trabalha de forma assíncrona e causa um baixo impacto no servidor transacional.

Palavras-chave: *Extract, Transform e Load* (ETL); Banco de Dados; *Data Warehouse* (DW); *Business Intelligence* (BI); *Change Data Capture* (CDC); Tempo Real e Latência Baixa.

Abstract

Nowadays, the acquisition of real-time information is necessary to support decision making. A common characteristic among companies that have developed this type of solution is the non-distinction between the On-Line Transaction Process (OLTP) and On-Line Analytical Process (OLAP). Therefore, user's satisfaction may decline over the years. This paper aims to build an OLAP environment to support real-time analytic queries with the integration through an Extract, Transform and Load (ETL) in conjunction with Change Data Capture (CDC) for continuous loads. Data was collected from Sergipe's market focusing on specific information on the use of its workloads.The results demonstrated that the vast majority of companies uses Microsoft SQL Server database management system. In addition, we selected one company as a tester to perform code analysis in Structured Query Language (SQL), interviews and requirements collection. The achieved outcomes were satisfactory as the large majority of the comparative aspects expressed superiority, thus solving the company's performance issues and incapacity to provide real-time information. CDC is a great choice for analytical environments once it works asynchronously and has a low impact on the OLTP environment. Therefore, CDC is a good choice for updating information on a low latency DW.

Keywords: Extract, Transform e Load (ETL); Database; Data Warehouse (DW); Business Intelligence (BI); Change Data Capture (CDC); Real-Time and Low Latency.

Lista de ilustrações

Figura 1 – Modelo Estrela 25
Figura 2 – Modelo Floco de Neve 25
Figura 3 – Fluxo de Captura de Dados do CDC com um *Log* de Transação 29

Figura 4 – Arquitetura Atual da Empresa Alfa 36
Figura 5 – SGBD's Utilizados no Mercado Sergipano 37
Figura 6 – Arquitetura em Tempo Real 38
Figura 7 – Habilitar CDC no Banco de Dados 39
Figura 8 – Habilitar CDC na Tabela 39
Figura 9 – Metadados e Dados Gerados na Tabela Orgão 41
Figura 10 – Exemplo de Pacote Base Visão Geral 42
Figura 11 – Exemplo de Pacote Base Visão Interna 42
Figura 12 – Exemplo de Pacote Incremental Visão Externo 43
Figura 13 – Exemplo de Pacote Incremental Visão Interna 44
Figura 14 – Pacote de Execução em Paralelo 45
Figura 15 – Tabela ADM_LOG 48
Figura 16 – Diagrama Relacional do DW 50
Figura 17 – Tempo de Execução na CPU 52
Figura 18 – Consumo de Memória 53
Figura 19 – Leituras Lógicas 53
Figura 20 – Escritas Lógicas 54
Figura 21 – Leituras Físicas 55
Figura 22 – Uso de CPU vs Tempo 55

Lista de tabelas

Tabela 1 – OLTP vs OLAP 20

Tabela 2 – Infraestrutura da Alfa 34
Tabela 3 – Cotidiano da Alfa 34
Tabela 4 – Tabela Vs Fluxo 45
Tabela 5 – Pacotes Restantes 46
Tabela 6 – Principais Tabelas da *Staging Area* 47
Tabela 7 – Tabela de Prefixos 47
Tabela 8 – Configuração do Servidor de Teste 51
Tabela 9 – Tabela de Estratégias 51

Lista de abreviaturas e siglas

CDC	Change Data Capture
OLTP	On-Line Transaction Process
OLAP	On-Line Analytical Process
ETL	Extract Transform Load
DW	Data Warehouse
DM	Data Mart
MOLAP	Multidimensional On-Line Analytical Process
ROLAP	Relational On-Line Analytical Process
HOLAP	Hybrid On-Line Analytical Process
SSAS	SQL Server Analysis Services
SSIS	SQL Server Integration Services
BI	Business Intelligence
CEO	Chief Executive Officer
DBMS	Database Management System
SGBD	Sistema de Gerenciamento de Banco de Dados
DDL	Data Definition Language
DML	Data Manipulation Language
SQL	Structured Query Language
1FN	Primeira Forma Normal
2FN	Segunda Forma Normal
3FN	Terceira Forma Normal
ODS	Operational Data Store
IDE	Integrated Development Environment
SSDT	SQL Server DataTools

SC2017	Visual Studio Community 2017
SAD	Sistema de Apoio à Decisão
ZLDWH	Zero-Latency Data Warehouse
RTDC	Real-Time Data Cache
CT	Change Table
MB	MegaByte
KB	KiloByte

Sumário

1 Introdução 13
1.1 Objetivos 14
1.2 Estrutura do Documento 15

2 Fundamentação Teórica 16
2.1 Importância dos dados para as organizações 16
2.2 Sistemas de Banco de Dados 16
2.2.1 Índices 18
2.2.2 Normalização 18
2.2.3 Procedimento Armazenado 19
2.2.4 *Trigger* 19
2.2.5 Transação 19
2.2.6 *Log* de Transações 19
2.2.7 Modelo Transacional vs Modelo Analítico 20
2.3 *Business Intelligence* 20
2.3.1 ETL 21
2.3.2 Evolução do ETL 22
2.3.3 *Data Warehouse* 22
2.3.4 *Staging Area* 24
2.3.5 Dimensão e Fato 24
2.3.6 Representação dos Dados no *Data Warehouse* 24
2.3.7 *Data marts* 26
2.3.8 Dimensão de Alteração Lenta 26
2.3.9 *Data Warehouse* com Latência Zero e Técnicas ETL 26
2.4 *Change Data Capture* 27
2.4.1 Estratégias de Implementação 28
2.4.2 Arquitetura do CDC no SQL *Server* 28

3 Metodologia 31
3.1 Materiais 31
3.2 Métodos 32

4 Desenvolvimento 33
4.1 Modelo Atual 33
4.1.1 Processo Analítico da Alfa 35
4.2 Modelo Esperado 36

4.2.1 Camada OLTP 37
4.2.2 Camada OLAP 41
4.2.2.1 SSIS 41
4.2.2.2 *Staging Area* 46
4.2.2.3 *Data Warehouse* 48
4.2.3 Camada Aplicação 49
4.3 Resultados e Discussões 49
4.3.1 Resultados e Discussões 51

5 Considerações Finais 57

Referências 59

1 Introdução

A habilidade de inter-relacionar fatos presenciados como um meio para guiar ações para um propósito específico é essencial para um negócio. Fornecer tais meios para que um gerente consiga tomar decisões rápidas e confiáveis para sua a organização, é tarefa de uma plataforma de Business Intelligence (BI) (GROSSMANN; RINDERLE-MA, 2015).

Uma empresa ao iniciar seu ciclo de vida acaba criando somente um ambiente transacional para atender todas as suas necessidades, esse ambiente com o tempo acaba não se adequando as suas transações diárias, relatórios, *dashboards*, etc. De acordo com Braghittoni (2017) esse tipo de ambiente corriqueiro em muitas empresas é definido como um ambiente transacional ou sistema OLTP (*On-Line Transaction Process*), o ambiente OLTP é projetado para suportar milhares de requisições simultâneas sem nenhum problema em um ambiente ideal.

A medida que as informações são cadastradas no sistema é bastante comum que o usuário solicite-as agrupadas ou sumarizadas de alguma forma. O adequado é que esses dados sejam transferidos para um ambiente analítico, também denominado como OLAP (*On-Line Analytical Process*). Esse ambiente é arquitetado para se adequar as requisições com um grande volume de dados, e também estruturado de forma dimensional e sem normalização, privilegiando assim consultas complexas. Um sistema bem projetado deve conter uma instância analática e transacional, ambas atuam em conjunto para fornecer ao usuário agilidade e confiança. Então, tornasse desejável definir uma ótima arquitetura e separar as suas responsabilidades entre o OLTP e OLAP (SHARDA; DELEN; TURBAN, 2016).

Os dados gerados pelos ambientes OLTP's precisam ser importados para o ambiente OLAP. Quando se trata de definir um ambiente OLAP, a sugestão mais aceita é a criação de um *Data Warehouse* (DW). Um DW é um repositório de dados com alta disponibilidade para aplicações de BI e Sistema de Suporte à Decisão (SAD), o termo em si indica um conjunto imenso de atividades relacionadas como projetar, implementar e a consumir. Dessa forma, a definição e construção de um DW é necessária (VERCELLIS, 2011).

Tank (2012) define um período em que um DW deve receber novos dados, esse período é chamado de latência, sendo que pode variar de dias, até poucas horas. Reduzir a latência é essencial para empresas competitivas, visto que acaba impactando diretamente em melhores decisões. Manter os dados atualizados minimizando o período de latência é uma tarefa árdua, pois exige uma interação contínua com à fonte proveniente dos dados. Logo, escolher uma tecnologia que possa proporcionar uma baixa latência e com o menor impacto possível é fundamental.

Um DW com baixa latência pode ser adquirido com a utilização de um ETL (Extract, Transform e Load) de cargas incrementais, existem várias maneiras de implementar o ETL e cada uma variando na qualidade dos dados e no desempenho. A otimização do processo ETL para tomada de decisão em tempo real acabou se tornando cada vez mais crucial. Em síntese, um ETL efetivo com baixa latência garante que boas decisões sejam tomadas (ALI; MOHAMED, 2016).

Com o objetivo de alcançar um ETL em tempo real é preciso de uma latência baixa, a tecnologia de captura de dados desenvolvida pela Microsoft chamada de *Change Data Capture* (CDC) encaixasse perfeitamente. Essa tecnologia pode ser descrita como um fluxo de dados com todas as alterações geradas no sistema OLTP, possibilitando assim uma sincronização com o OLAP. Outra característica importante é o seu ótimo desempenho, o CDC foi projetado para ter o menor impacto possível no banco de dados transacional. Utilizando o CDC, as alterações realizadas pelos usuários serão refletidas o mais rápido do possível no DW (MICROSOFT, 2017).

1.1 Objetivos

Este trabalho tem como objetivo construir uma arquitetura para dar suporte as consultas analíticas em tempo real, integrado por meio do processo de Extração, Transformação e Carga (ETL) em conjunto com o Change Data Capture (CDC) para cargas contínuas. Permitindo requisições em tempo real para ambientes que precisam das informações de forma rápida.

- Projetar o modelo dimensional do banco de dados também chamado de *data warehouse*;
- Arquitetar o processo de carga ETL (*Extract, Transform e Load*). Esse processo envolve o a extração dos dados de diversas fontes externas, validar os dados e carregar os dados no DW;
- Definir a área de armazenamento temporário, também conhecida como *Staging Area*;
- Especificar o CDC para integrações contínuas em tempo real.

1.2 Estrutura do Documento

Para facilitar a navegação e melhor entendimento este documento está estruturado em capítulos e seções, que são:

- Capítulo 2 - Fundamentação Teórica: Aborda os conceitos importantes para o entendimento do trabalho, como o sistema de gerenciamento de banco de dados e *Business Intelligence*;
- Capítulo 3 - Materiais e Métodos: Demonstra todas as ferramentas aplicadas e também a metodologia adotada no trabalho;
- Capítulo 4 - Desenvolvimento: Descreve a arquitetura adotada pela Alfa com a elaboração da nova solução;
- Capítulo 5 - Considerações Finais: Faz uma visão geral do projeto desenvolvido, mostrando como os objetivos foram alcançados, possíveis melhorias e trabalhos futuros.

2

Fundamentação Teórica

Neste capítulo são abordados os fundamentos necessários para o entendimento do trabalho, começando pelo conceito de Sistemas de Banco de Dados, *Business Intelligence* e terminando com *Change Data Capture*.

2.1 Importância dos dados para as organizações

Um *Chief Executive Officer* (CEO) de uma empresa pode querer visualizar se o faturamento da empresa este mês está abaixo do esperado. A necessidade de informações precisas são essenciais para qualquer negócio, visto que informações com poucos detalhes podem levar a um julgamento falho da situação. Diante desse cenário, caso o CEO tivesse em suas mãos uma ferramente que disponibilizasse as informações de forma contextual e clara, ele poderia tomar uma decisão mais rápida e eficaz a respeito do problema em questão (BRAGHITTONI, 2017).

Entender a diferença entre dado e informação se faz realmente necessária. Dado é definido como algo bruto e ele por si só não tem significado. Os dados são materiais brutos que não revelam significado nenhum até serem processados. Enquanto, a informação é o resultado do processamento dos dados. Quando os dados são organizados de acordo com um contexto, o resultado desse agrupamento contextualizado é o que gera a informação. A informação pode ser organizada para relevar padrões simples, assim como também padrões complexos para o usuário.(CORONEL; MORRIS; ROB, 2009).

2.2 Sistemas de Banco de Dados

Nos últimos anos o avanço da tecnologia levou a uma mudança de paradigma na humanidade. Há pouco tempo as pessoas costumavam escrever em papéis o decorrer dos acontecimentos na empresa, o adendo da tecnologia proporcionou que as informações fossem registradas em

um programa de computador. Esse programa é um banco de dados com um conjunto de dados relacionados e que por sua vez os dados são fatos conhecidos que podem ser registrados e com algum significado implícito. Por exemplo, temos os nomes, os números de telefone, os endereços e entre outros (ELMASRI, 2008).

Segundo Elmasri (2008), um banco de dados possui as seguintes características implícitas:

- **Minimundo:** É a representação de algum aspecto existente do mundo real, muitas vezes chamado de universo de discurso, eventuais mudanças nele são refletidas no banco de dados;
- **Relacionamento:** É uma coleção lógica e coesa com algum significado inerente. Uma massa aleatória de dados não é descrita como um banco de dados;
- **Específico:** Um banco de dados é projetado, construído e populado com dados para um determinada finalidade. Ele foi definido para uma gama de usuários e para um grupo específico de aplicações previamente concebidas.

Um *Database Management System* (DBMS) ou Sistema de Gerenciamento de Banco de Dados (SGBD) é uma coleção de dados inter-relacionados e um conjunto de programas para acessar esses dados. A coleção de dados normalmente é chamada de banco de dados. Esses sistemas são projetados para gerenciar uma quantidade demasiada de informações. O gerenciamento envolve a definição de estruturas para armazenamento de informações e fornecer meios para a manipulação dessas informações. Exemplos de SGBDs são: Oracle, SQL Server, DB2, PostgreSQL ,MySQL, etc (SILBERSCHATZ; KORTH; SUNDARSHAN, 2012).

Silberschatz, Korth e Sundarshan (2012) definem um SBGD com as seguintes estruturas:

- **Visão dos dados:** Uma finalidade de um SGBD é fornecer ao usuário múltipla visões abstratas dos dados, ou seja, o sistema deve abstrair como os dados são armazenados e persistidos no sistema;
- **Abstração de dados:** Projetistas utilizam estruturas de dados complexas para fazer a representação dos dados no banco de dados. Muitos usuários não são treinados para lidar com tal complexidade, logo, os desenvolvedores ocultam a complexidade dos usuários sob vários níveis de abstrações. Alguns níveis de abstrações são: físico, lógico e visão;
- **Instâncias e esquemas:** A medida que o tempo passa é normal o banco de dados tenha algumas mudanças, novas informações são inseridas, deletadas ou atualizadas o tempo todo. A coleção das informações armazenadas do banco de dados em um determinado momento é uma instância do banco de dados. O projeto geral do banco de dados é o esquema dele, e detalhe que os esquemas raramente ou nunca são modificados;

- **Modelo de dados:** Como um apoio a estrutura de um banco de dados está o modelo de dados. Ele é uma coleção de ferramentas conceituais para descrever dados, relações de dados, semântica de dados e restrições de consistências. Os modelos de dados são classificados em quatro categorias: modelo relacional, modelo de entidade/relacionamento, modelo de dados baseado em objeto e modelo de dados semi-estruturado;
- **Linguagens de banco de dados:** É a linguagem adotada pelo banco de dados. Geralmente fornecem uma *Data Definition Language* (DDL) e uma *Data Manipulation Language* (DML). Na prática, essas duas linguagens representam uma única linguagem que é a *Structured Query Language* (SQL).

2.2.1 Índices

Um dos melhores modos de reduzir a utilização de disco é fazer o uso de um índice, ele permite encontrar dados em uma tabela sem precisar fazer o escaneamento inteiro da mesma. Analogamente, um índice de um banco de dados é a mesma coisa que um índice de um livro, visto que quando desejamos procurar alguma frase olhamos primeiramente no seu índice antes de sair fatiando o livro. Caso contrário, sem o índice para auxiliá-lo você iria ter que olhar o livro página por página até encontrar o que você estava procurando. Então, com um índice você sabe exatamente onde procurar pela informação (FRITCHEY, 2018).

2.2.2 Normalização

O processo de normalização foi proposto por Codd (1972), esse processo impõe uma série de critérios sobre as tabelas do banco de dados. A aplicação das formas normais visam a eliminação de falhas no projeto, redundância e várias entidades em uma mesma tabela. O processo, que prossegue em um padrão de cima para baixo, que avalia cada relação em comparação com os critérios para as formas normais, e decompondo as relações conforme a necessidade(ELMASRI, 2008).

Elmasri (2008) define as três principais formas normais da seguinte forma:

- **Primeira Forma Normal (1FN):** Historicamente ela foi definida para reprovar atributos multivalorados, atributos compostos e suas combinações. Ela afirma que o domínio de um atributo deve incluir apenas valores atômicos e que o valor de qualquer atributo em uma tupla deve ser um único valor do domínio desse atributo;
- **Segunda Forma Normal (2FN):** Define que para estar na 2FN qualquer atributo de uma relação deve ser funcionalmente dependente da chave primária. Uma observação importante é que ela deve estar também na 1FN para ser 2FN;
- **Terceira Forma Normal (3FN):** Uma tabela encontra-se na terceira forma normal, quando, além de estar na 2FN, não contém dependências transitivas. Uma dependên-

cia transitiva ocorre quando uma coluna, além de depender da chave primária da tabela, depende de outra coluna ou conjunto de colunas da tabela.

2.2.3 Procedimento Armazenado

Um procedimento armazenado ou *stored procedure* é o nome escolhido para uma coleção de declarações escritas na linguagem SQL. Procedimentos armazenados podem ser usados para encapsular transações de negócio, por exemplo: você pode criar um procedimento armazenado para vendas de produto, uma atualização de crédito ou uma adição de crédito a um cliente, assim encapsulando o código no procedimento armazenado torna possível. A sua utilização é bastante empregada para reduzir o tráfego de internet e aumentar o performance, tal que o procedimento armazenado fica no servidor (CORONEL; MORRIS; ROB, 2009).

2.2.4 *Trigger*

Silberschatz, Korth e Sundarshan (2012) definem um *trigger* como uma instrução que o sistema executa automaticamente como um efeito colateral de uma modificação no banco de dados. O banco de dados armazena um *trigger* como se fosse um dado normal, de modo que é persistido e acessível a toda operação no banco de dados. Quando entramos com um *trigger* em um banco de dados, o SBGD assume a responsabilidade de executá-lo sempre que um determinado evento ocorrer e a condição correspondente for satisfeita.

2.2.5 Transação

Uma transação é uma unidade atômica de trabalho que deve ser concluída totalmente ou não ser invalidada de forma alguma. Para fins de recuperação o sistema precisa registrar quando cada transação começa, termina e confirma ou aborta. As transações devem possuir várias propriedades, essas são: atomicidade, consistência, isolamento e durabilidade (ACID). Com isso, o SGBD se encarrega de manter a veracidade de seu conteúdo (ELMASRI, 2008).

2.2.6 *Log* de Transações

Um SGBD usa um *log* de transação para manter um rastreamento de todas as transações que ocorrem no banco de dados e a informação armazenada neste *log* é utilizada para diversos fins; por exemplo: dar *rollback* de algum bloco de código, o término abrupto do programa, recuperação de falha e uma possível falha no disco. O SBGD executa várias transações no banco de dados ao mesmo tempo que automaticamente atualiza o *log* de transações, embora a manutenção do log incremente a taxa de processamento do SBGD, esse custo é válido para manter a segurança e consistência dos dados (CORONEL; MORRIS; ROB, 2009).

2.2.7 Modelo Transacional vs Modelo Analítico

Os sistemas de informação suportam várias transações cotidianas como caixas eletrônicos, depósitos no banco, saque da conta corrente e assim por diante, esses sistemas trabalham em cima de um modelo transacional ou *On-Line Transaction Process* (OLTP). O OLTP é projetado para suportar uma grande carga de transações tendo como operações básicas o atualizar, inserir e deletar. Por sua vez, é comum obter informações detalhadas sobre determinados eventos no sistema, no entanto, o modelo OLTP não realiza com grande desempenho essas requisições. Então o On-Line Analytical Process (OLAP) entra em cena, que ao contrário de sistemas OLTP são desnormalizados, pois, a normalização não é efetiva para sistemas OLAP na entrega de análises gerenciais. A Tabela 1 ilustra uma breve comparação entre cada ambiente (TURBAN et al., 2009).

Tabela 1 – OLTP vs OLAP

Características	**OLTP**	**OLAP**
Volatilidade	Dados dinâmicos	Dados estáticos
Estado da informação	Somente dados atuais	Dados atuais e históricos
Dimensão do Tempo	Implícito e atual	Explícito e variante
Granularidade	Dados detalhados	Dados agregados e consolidados
Atualizações	Contínuas e irregulares	Periódica e regulares
Atividades	Repetitivas	Imprevisíveis
Flexibilidade	Baixa	Alta
Performance	Alta, alguns segundos por query	Pode ser baixa para queries complexas
Usuários	Usuários comuns	Usuários avançados
Funções	Operacionais	Analíticas
Propósito de uso	Transacional	Queries complexas e suporte de decisão
Prioridade	Alta performance	Alta flexibilidade
Métricas	Alta taxa de transações	Resposta rápida
Tamanho	Megabytes para gigabytes	Gigabytes para terabytes

Fonte: (VERCELLIS, 2011)

2.3 *Business Intelligence*

Gbosbal e Kim (1986) definem *Business Intelligence* (BI) como um conjunto de conceitos e métodos para melhorar o processo de tomada de decisão. Um BI baseia-se em agrupar informações de diversas fontes e apresentá-las de forma centralizada sob uma perspectiva em comum, ele pode ser definido como um conjunto de modelos matemáticos e metodologias de análise que exploram a disponibilidade de gerar informação e conhecimento útil (BRAGHITTONI, 2017).

Turban et al. (2009) definem BI como um termo "guarda-chuva"que inclui arquiteturas, ferramentas, banco de dados, aplicações e metodologias. Os principais objetivos do BI são: permitir o acesso interativo aos dados que opcionalmente pode ser em tempo real, proporcionar a manipulação dos dados e fornecer aos analistas de negócios a capacidade de fornecer uma análise adequada. Analisando os dados, situações, históricos e atuais, os tomadores de decisão conseguem valiosas informações que podem servir como base para futuras decisões.

Grossmann e Rinderle-Ma (2015) sumarizam diferentes definições de BI e especificam os seguintes recursos:

- **Tarefa do BI:** A principal tarefa do BI é prover suporte à decisão para metas específicas. Cada meta é definida em um contexto de atividades de negócio, que por sua vez abrangem diversas áreas de domínio;
- **Fundação do BI:** Suporte à decisão depende de informações empíricas baseadas em dados, além disso, essa base empírica utiliza diferentes tipos de conhecimento e teorias para geração de informação;
- **Realização do BI:** O suporte à decisão tem que utilizar as atuais capacidades de tecnologias da comunicação e informação;
- **Entrega do BI:** Um sistema de BI deve entregar informações coerentes, em qualquer hora possível, para a pessoa certa e da melhor forma apropriada existente.

2.3.1 ETL

Para que os dados fiquem disponíveis na plataforma BI, torna-se necessário um processo de cargo, esse processo é dominado de ETL (*Extract, Transform and Load*). Atualmente existem várias ferramentas no mercado que permitem a construção do ETL de forma visual (arrastar e soltar). Temos como algumas ferramentas o PowerCenter, Pentaho *Data Integration* e SQL *Integration Services* (SSIS) (BRAGHITTONI, 2017).

Braghittoni (2017) define as três etapas do ETL da seguinte forma:

- ***Extract:*** É um processo de extração periódica dos dados de diferentes origens, essa periodicidade implica na latência de como as informações são analisadas, que por sua vez podem ser mensais, semanais ou diárias. Também faz necessário informar que o processo de carga é repetido diversas vezes, adicionalmente meios de tratamentos de erros e rotinas de *logs* são incluídos para auxiliar no processo;
- ***Transform:*** É um processo pelo qual os dados são trabalhos, colocados sob um formato específico, validados mediante regras específicas de negócio, calculados e etc;
- ***Load:*** È o processo em que os dados são definitivamente carregados na plataforma de BI. A inclusão deve ser feita de forma incremental, pois, uma vez que a informação é inserida, jamais poderá ser alterada. Entretanto existem técnicas para lidar com tais situações dentro do BI.

2.3.2 Evolução do ETL

Tradicionalmente, o processo ETL é executado periodicamente (semanalmente, diariamente). Com a crescente popularidade dos *data warehouses* e *data marts*, a habilidade de atualizar os dados se tornou mais importante. Sistemas inteiros irão reconstruir completamente o seu projeto ETL para alimentar um DW que é utilizado para relatórios, esteja sempre atualizado. Entretanto, o processo ETL precisava evoluir para seguir as tendências do mercado, um novo mecanismo precisaria surgir para isso (TANK, 2012).

Kakish e Kraft (2012) Citam a evolução do ETL em 3 gerações, que serão descritas logo abaixo:

- **Primeira Geração:** A primeira geração foi escrita em código nativo da plataforma do sistema operacional, podendo ser somente executado na plataforma nativa do sistema. A maioria dos códigos gerados eram escritos em COBOL, porque a primeira geração de dados era armazenada em *mainframes*. Essas ferramentas faziam o processo de integração fácil desde que o código fosse nativo;
- **Segunda Geração:** A segunda geração embutia um motor ETL para executar processos de transformações. O desenvolvimento foi simplificado para os projetistas para somente uma linguagem de programação, entretanto dados vindos de várias fontes heterogêneas deveriam passar pelo motor de ETL linha por linha. Essa geração ficou marcada por problemas de sobrecarga no sistema;
- **Terceira Geração:** A terceira geração construiu uma arquitetura distribuída com a habilidade de gerar código nativo SQL, essa geração foi marcada por reduzir o tráfego de internet para melhorar o desempenho de carga entre vários banco de dados. Outra característica marcante é o uso de um SBGD relacional para transformação dos dados.

2.3.3 *Data Warehouse*

Em simples termos, um *data warehouse* (DW) é uma piscina de dados produzidos para apoiar a tomada de decisão, também é um repositório dos dados atuais e históricos de potencial interesse para os gerentes de uma organização. Os dados são estruturados para estarem sempre disponíveis e serem utilizados pelas diversas formas de atividades de processamento analítico, como por exemplo: mineração de dados, consultas, relatórios e outras aplicações de suporte a decisão. Um DW pode ser caracterizado com as seguintes características: orientado por assunto, integrado, varia de acordo com o tempo e não é volátil (SHARDA; DELEN; TURBAN, 2016).

Vercellis (2011) cita diversos motivos para que um DW seja separado do banco de dados OLTP. Entre eles, ele cita os mais relevantes, que são:

- **Integrado:** Em muitas situações sistemas de suporte à decisão devem obter informações de diferentes origens, entretanto disponibilizar as informações de forma centralizada para acesso entre as diversas partes da organização. Um DW integra múltiplas origens e disponibiliza o acesso as informações de maneira fácil;

- **Qualidade:** Os dados transferidos do sistema operacional para o *data wareshouse* são examinados e corrigidos a fim de obter informações confiáveis e livres de erros o máximo do possível;

- **Eficiência:** Uma consulta destinada a extrair informações para uma análise de inteligência de negócios se torna onerosa em termos de recursos de computação e tempo de processamento, como consequência se a consulta for direcionada para o banco transacional, este poderia correr sério risco de ficar ineficiente;

- **Extensibilidade:** Os dados armazenados em sistemas transacionais estendem-se por um limite intervalo de tempo no passado. De fato, devido a limitações na capacidade de memória, os dados em relação a períodos passados são regularmente removidos dos sistemas OLTP e permanentemente arquivados em dispositivos de armazenamento em massa *off-line*, como DVDs ou fitas magnéticas. Entretanto, o DW fornece a capacidade para armazenar informações históricas.

Sharda, Delen e Turban (2016) detalham cada uma das principais características de um DW abaixo:

- **Orientado por assunto:** Os dados são organizados por assuntos detalhados, como vendas, produtos ou clientes, contendo somente informações relevantes para o suporte à decisão. Orientação a assunto habilita o usuário a determinar não somente como o seu negócio está sendo executado, mas também o porquê;

- **Integrado:** A integração está fortemente ligada à orientação a assunto, um DW deve lidar com dados provenientes de diversas fontes,p ara fazer isso ele deve lidar com conflitos de nomes e anomalias entre as unidades de medidas. Adicionalmente, um DW é presumido para ser totalmente integrado;

- **Variável com o tempo:** Um DW armazena dados do passado e os dados não fornecem necessariamente o atual status (exceto em sistemas de tempo real). Eles detectam tendências, desvios, e um relacionado de longo prazo para previsões e comparações. Todo DW tem uma dimensão temporal e ela é uma das dimensões mais importantes que todos os DW devem apoiar;

- **Não volátil:** Após os dados serem carregados no DW os usuários não podem mudar ou atualizar os dados. Eventuais alterações são registradas como novos registros e caso um determinado dado se torne obsoleto é descartado.

2.3.4 *Staging Area*

Um dos postulados do DW é que ele não é volátil, ou seja, se um registro entrou nele, então não pode ser alterado, entretanto no processo de carga teremos que avaliar se o dado está correto. Eventualmente alterar a formatação dele, verificar se ele deve ser mesmo carregado ou se ele é uma duplicidade e etc. Uma área intermediária se torna necessária para trabalhar com os dados, sendo posteriormente, populada no processo de carga e depois devidamente limpa para um novo ciclo (BRAGHITTONI, 2017).

A *Staging Area* pode fazer uso de uma área intermediária opcional cujo nome é ODS (*Operational Data Store*), essa área deve ser uma base de dados com utilização previsível, parcialmente estruturada e analítica com um histórico de apenas 30 ou 60 dias e as informações organizadas por área de negócio. Na realidade, os dados são armazenados como no ambiente operacional dentro do ODS, ou seja, não estão modelados para consultas gerenciais, entretanto, podem ser utilizados no auxílio de restauração de cargas problemáticas (JÚNIOR, 2004).

2.3.5 Dimensão e Fato

Dimensões são associadas com entidades que englobam os processos em volta da organização, tabelas do tipo dimensão correspondem as entidades primárias presentes em um DW, sendo que na maioria das vezes seus dados são de origem de sistemas OLTP, como clientes, produtos, funcionários, localização e assim por diante. Certas dimensões são internamente estruturadas de acordo com relações hierárquicas, como a dimensão tempo, ela pode ser hierarquicamente estruturada da seguinte forma: dia, semana, mês e ano, também de uma forma similar a dimensão produto tem sua hierarquia formada por: rua, cep, cidade, região, país. De certa forma, dimensões predeterminam o caminho principal pelo qual uma análise OLAP será previamente elaborada (VERCELLIS, 2011).

Uma tabela fato contém transações de vendas e faz referências a várias tabelas dimensões, como: produto, fornecedor, cliente e tempo. As correspondentes medidas de interesse são atributos como quantidade de itens vendidos, preço unitário e desconto. Neste exemplo uma tabela fato permite um analista avaliar as tendências das vendas ao longo do tempo, também é possível avaliar o total de produtos comprados por um único cliente, ou os fornecedores que mais venderam produtos no ano (VERCELLIS, 2011).

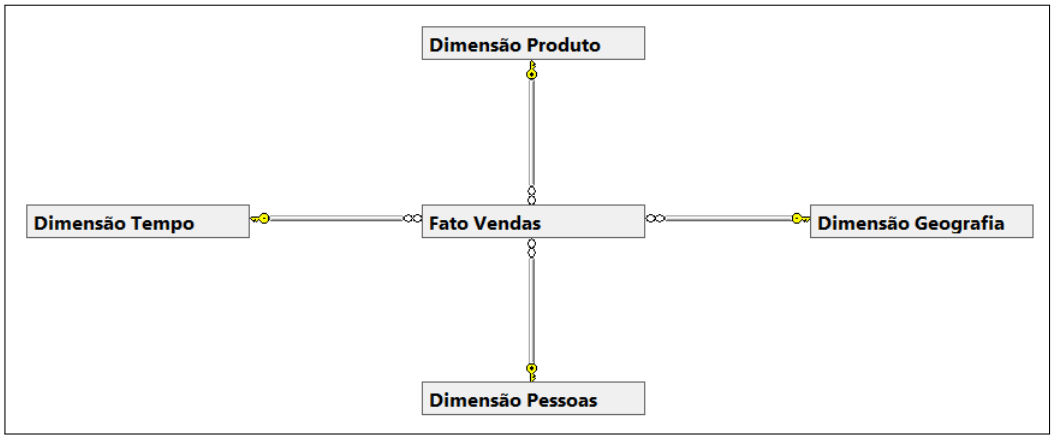
2.3.6 Representação dos Dados no *Data Warehouse*

O *design* da representação dos dados em um *data warehouse* é baseado em um conceito de modelagem dimensiona, ele é um modelo projetado para recuperação de grande volume de dados. O armazenamento dos dados em um DW deve ser projetado de uma maneira que acomode os dados, mas também acelere o processamento de consultas multidimensionais complexas.

Frequentemente, o modelo estrela e o modelo floco de neve são os mais utilizados para alcançar esse objetivo (SHARDA; DELEN; TURBAN, 2016).

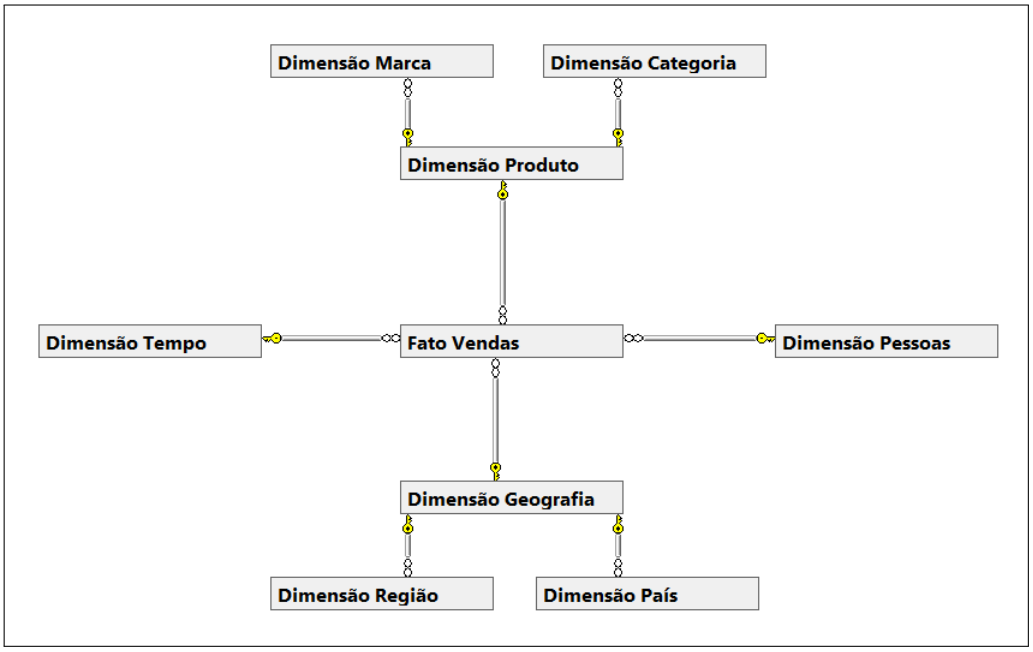
O modelo estrela é o mais utilizado e é o mais simples dentre os modelos existentes, nele uma tabela fato central é criada e tem várias tabelas ao seu redor ligadas por chaves primárias. A outra opção de modelagem chama-se floco de neve, essa modelagem organizada as tabelas de um modo semelhante a um modelo normalizado. As Figuras 1 e 2 ilustram bem o modelo estrela e floco de neve respectivamente (SHARDA; DELEN; TURBAN, 2016).

Figura 1 – Modelo Estrela



Fonte: Adaptado de (SHARDA; DELEN; TURBAN, 2016).

Figura 2 – Modelo Floco de Neve



Fonte: Adaptado de (SHARDA; DELEN; TURBAN, 2016).

2.3.7 *Data marts*

Data marts (DM) são sistemas que mineram todos os dados necessários para um departamento específico de uma companhia, como *marketing* ou logística, adicionalmente um DM pode ser considerado como uma parte funcional de um *data warehouse* em uma escala menor, e também um tipo mais específico de DW de uma companhia. Em geral, *data marts* são integrados com outros dados que a organização possui, por exemplo: um DM de *marketing* irá conter dados extraídos de um DW central, tais informações obtidas podem ser a respeito dos clientes ou funcionários. Por fim, muitas empresas preferem projetar e desenvolver de modo incremental um *data mart* em vez de um *data warehouse* central, a fim de reduzir o tempo de implementação e custo (VERCELLIS, 2011).

2.3.8 Dimensão de Alteração Lenta

Ao cadastrar um produto no sistema esse produto daqui a algum tempo pode ser alterado, essas dimensões que sofrem alterações com o decorrer do tempo são chamadas de *Slowly Changing Dimension* (SCD). A depender da taxa de mudanças na dimensão é esperado que seja utilizada alguma técnica para adequar-se a ela. (BRAGHITTONI, 2017)

Costa et al. (2015) citam as principais SCD's da seguinte forma:

- **1SCD:** Nessa abordagem o valor antigo sobrescreve a linha da dimensão atual, assim o atributo sempre irá refletir o valor mais recente proveniente do ambiente operacional;
- **2SCD:** Há um tratamento e armazenamento de histórico, através da criação de novos registros, neste tipo todos os valores de um atributo ao longo do tempo são armazenados no DW;
- **3SCD:** É utilizada quando há uma mudança e existe a necessidade de manter o histórico sem a criação de um novo registro, essa característica é alcançada através do armazenamento do histórico em vários atributos de um mesmo registro de dados.

2.3.9 *Data Warehouse* com Latência Zero e Técnicas ETL

Data Warehouse com Latência Zero ou Zero-Latency Data Warehouse (ZLDWH) é um versão estendida de um projeto de um DW que permitirá uma processo completo de *business intelligence* para observar, entender, prever, reagir, reogarnizar, monitorar, automatizar e controlar loops de feedback com uma latência mínima (NGUYEN; TJOA, 2006).

Ali e Mohamed (2016) Definem algumas técnicas utilizadas para a elaboração de um ZLDWH:

- **ETL quase em tempo real:** O custo efetivo solução para aplicações que não tem alta demanda por dados em tempo real é apenas aumentar a frequência de carregamento, por exemplo: carregar duas vezes os dados ao dia;

- **Alimentação Direta com Gotejamento:** Esta abordagem dados verdadeiramente em tempo real podem ser alcançados movendo ininterruptamente as alterações nos dados da origem, e posteriormente inserindo ou atualizando a tabela fato. Existe um problema de escalabilidade com essa abordagem porque procedimentos complexos não executam bem atualizações contínuas. um desvantagem importante desta abordagem é que constantes atualizações nas tabelas utilizadas pelas ferramentas OLAP levam a degradação das consultas do DW;

- **Gotejar e Virar:** Nesta abordagem o dado é inserido e atualizado em tabelas no DS que estão no mesmo formato das tabelas o DW. O DW pode acessar os dados obtendo uma cópia das tabelas de preparo no DS para as tabelas de fatos, além disso a janela de atualização pode variar de horas para minutos;

- **Cache de dados externos em tempo real:** Armazenar os dados fora do DW em um cache de dados em tempo real ou Real-Time Data Cache (RTDC). A função do RTDC é carregar os dados em tempo real num banco de dados que são provenientes dos sistemas originários, esta abordagem resolve o problema de escalabilidade do DW por fazer consultas diretas ao RTDC para acesso em tempo real.

2.4 *Change Data Capture*

Atualizar ou inserir novas informações em um DW não é uma tarefa trivial e no mercado existem duas formas comumentemente aceitas. A primeira abordagem consiste em inserir todo o universo de dados sempre que for atualizar o DW, enquanto a segunda abordagem insere todos os dados no DW e posteriormente com contínuas cargas incrementais das mudanças efetuas no OLTP. Com o propósito de construir uma aplicação em tempo real utilizando atualizações incrementais e com menor tempo de latência possível, desta forma a segunda opção se ajusta ao cenário perfeitamente (KAKISH; KRAFT, 2012).

CDC é uma aproximação baseada na segunda abordagem para integração de dados, baseado na identificação, captura, e na entrega das mudanças feitas nos sistemas transacionais ou operacionais. O CDC faz a integração dos dados de uma maneira simples por processar somente as mudanças, assim realizando de forma muito eficiente a parte "*Extract*" do processo de ETL. Caso seja implementada de maneira correta a redução da latência é notável entre a data que ocorreu a mudança no sistema origem e a hora que a atualização está disponível no DW (TANK, 2012).

2.4.1 Estratégias de Implementação

Ali e Mohamed (2016) Definem um lista com as principais metodologias utilizadas para construir um CDC:

- ***Triggers* no banco de dados:** Um *trigger* é um pedaço de código opcional que pode ser configurada para executar ou capturar qualquer mudança nas linhas das tabelas. Sua grande vantagem é que não requer mudança na camada de aplicação, entretanto, acaba denegrindo o desempenho do OLTP e também gerando problemas de *deadlocks* nas tabelas onde os *triggers* residem;
- **Armazenar Metadados:** Esta técnica consiste em executar consultas ciclicamente para capturar e identificar as mudanças realizadas. Adicionalmente é necessário a utilização de algumas propriedades nas tabelas, por exemplo, ter uma coluna chamada "última atualização" ou um inteiro crescente como chave primária;
- **Particionamento:** Alguns sistemas podem utilizar um limite de particionamento. As tabelas são particionadas com uma data chave, que permite identificar facilmente novos dados. Por exemplo, uma tabela de pedidos é particionada por semana, então será fácil identificar os novos registros da semana.
- **Processamento do *log* de transações:** A maioria dos bancos relacionais escrevem as suas mudanças dos seus dados em um mecanismo interno chamado log de transações, ele pode ser utilizado *log* para extrair as mudanças armazenadas dentro dele. Implementar do zero esse mecanismo é extremamente inviável por o *log* ter pouca documentação e estar em um formato binário, porém a maioria dos SGBD's já fornecem essa implementação. Diante disso, esta não requer nenhum código adicional no sistema OLTP e executa suas operações de maneira assíncrona e não utiliza os dados originais do OLTP.

Para uma aplicação em tempo real capturar mudanças dos registros nas tabelas de um banco de dados é o essencial e utilizar o próprio *log* de transações é a mais aceitável. Então todas as instruções DML serão capturadas e processadas durante cada ciclo, visto que devem ser auxiliadas com informações das colunas e os metadados exigidos para uma aplicação de consumo (ZHOU; YANG; XU, 2011).

2.4.2 Arquitetura do CDC no SQL *Server*

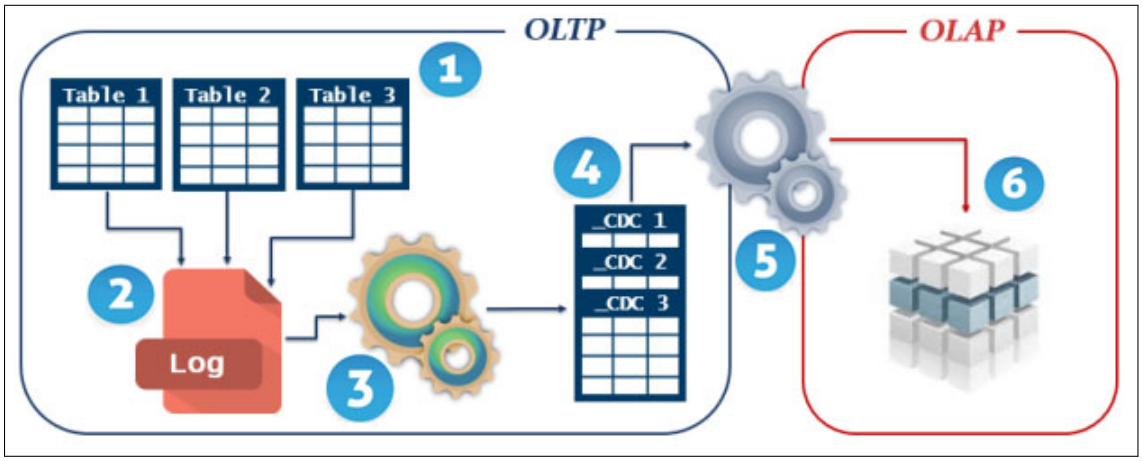
Através do CDC em conjunto com o SQL *Server* torna-se possível a criação de ambientes que apresentam processamento diário de informações. Uma tecnologia que utilize uma réplica dos dados de uma tabela origem não é apropriada, por exemplo, consultar a origem pra ler todos os seus dados para carregar o DW a cada ciclo, em vez disso, uma tecnologia de fluxo seguro de dados com somente as alterações é mais apropriada, a fim de evitar processar todo o montante de

dados. O CDC entra justamente nesse ponto fornecendo uma tecnologia de *streaming* de dados para cargas incrementais (MICROSOFT, 2017).

O CDC atua na captura dos dados para não prejudicar o sistema transacional, esta captura é realizada através dos dados salvos dentro do arquivo de *log* que são guardados quando a transação original for concluída. Monitorar as mudanças do OLTP e processá-las não é uma tarefa trivial, muitas vezes demanda um esforço imenso de configuração e diversas alterações no OLTP. Esse processo de captura é efetuado de modo assíncrono, dando mais estabilidade ao ambiente OLTP, evitando assim possíveis *deadlocks* nas tabelas de trabalho do próprio OLTP, concluindo o CDC demonstra ser um excelente mecanismo na captura e processamento dos dados (DONSELAAR, 2015).

Luiz (2016) define todo o processo de captura dos dados, iniciando com interações no OLTP até a sua disponibilidade no OLAP. Esse processo é contextualizado na Figura 3 que será descrito a seguir:

Figura 3 – Fluxo de Captura de Dados do CDC com um *Log* de Transação



Fonte: Adaptada de (LUIZ, 2016)

- **Etapa 1:** Os principais componentes desta etapa são as tabelas do OLTP. Um mapeamento detalhado de quais tabelas e quais colunas devem ser monitoradas é realizado aqui, e também é importante afirmar que não necessariamente todas as tabelas serão monitoradas;
- **Etapa 2:** O *log* de transação do banco de dados registra todas as instruções DML's executadas sobre as tabelas. Então, nessa etapa serão registradas todas as suas alterações para uma posterior análise;
- **Etapa 3:** A partir da definição de quais tabelas e colunas que serão monitoradas, entra em cena o job do banco de dados. O papel deste job é escanear o log de transação para identificar ocorrências sobre um determinado registro na sua tabela;

- **Etapa 4:** Para cada tabela mapeada existe uma outra tabela espelhada, essa tabela contém os registros da tabela de origem com cada operação DML registrada, e também um conjunto de colunas com metadados sobre o que aconteceu;
- **Etapa 5:** O CDC é acoplado a uma ferramenta de ETL para extrair todas as mudanças decorrentes a partir de um marco inicial. O marco inicial é definido pela primeira carga no ambiente OLAP, além disso, cargas subsequentes irão utilizar esse ponto de partida para identificar quais registros sofreram mudanças. Por fim, durante o próprio ETL as instruções de *update*, *insert* e *delete* serão separadas para processamento individual com a finalidade sincronizar o OLTP e o OLAP;
- **Etapa 6:** As novas informações já estão devidamente carregadas em suas dimensões e fatos. A disponibilidade dos dados para interpretações em uma ferramenta de BI é real.

3
Metodologia

O capítulo tem o objetivo de especificar as ferramentas e metodologia aplicadas ao trabalho. Expondo primeiro os materiais utilizados e subsequentemente os métodos usados, e também uma breve descrição deve ser informada sobre cada item exposto.

3.1 Materiais

Problemas são resolvidos de uma maneira mais inteligente com a ferramenta certa. A ferramenta desempenha um papel essencial no desenrolar da problemática, seja ela facilitando a solução ou agilizando-a. O desenvolvimento de *software* está diretamente ligado com boas ferramentas que são empregadas em uma boa documentação, auxílio na documentação, elaboração de gráficos, etc. Então a viabilidade do projeto pode ser incrementada com a escolha correta dos materiais dispostos no mercado.

Uma listagem será exibida com todos os materiais empregados neste trabalho a seguir:

- ***Visual Studio Community 2017*:** É um Ambiente de Desenvolvimento Integrado, também conhecido do inglês *Integrated Development Environment* (IDE). O Visual Studio Community 2017 (VSC2017) [1] possui vários recursos de versionamento de código e uma vasta documentação na *internet*;
- ***Microsoft SQL Server Integration Services*:** O *SQL Server Integration Services* (SSIS) será utilizado para fazer o ETL em conjunto com o VSC2017, para configurá-lo é necessário instalar o *SQL Server Data Tools* (SSDT) [2] que é o pacote de utilitários para o VSC2017. A criação de um pacote é feita de uma forma quase que visual e posteriormente cada pacote é enviado para o servidor de banco de dados para ser executado;

[1] Encontrado em: http://bit.ly/2kQzvS0
[2] Encontrado em: http://bit.ly/2kRXVuy

- ***Microsoft SQL Server Enterprise 2016*:** Na sua arquitetura de 64-bit é o SGBD adotado para armazenar o banco de dados do ambiente analítico e a sua versão é a 13.0.4224.16 [3]. Nesse servidor irá conter o banco de dados do DW e a *Staging Area*.

3.2 Métodos

- **Pesquisas Publicadas:** Pesquisas foram realizadas no mercado sergipano com o propósito de analisar a atual situação do dele em relação a utilização de SGBD's. Com o intuito de construir um raciocínio a respeito do mercado foi lançada a primeira pesquisa, esta foi de propósito geral cuja qualquer empresa poderia respondê-la. Um segundo levantamento foi elaborado, mas desta vez para uma empresa alvo que apresenta o problema descrito neste trabalho;
- **Entrevistas:** A realização de entrevistas para o entendimento da situação da empresa Alfa foi importante, a necessidade de interações com os desenvolvedores para chegar a uma solução viável acabou sendo efetivada. Ao total foram efetuadas quatro entrevistas com algum membro específico da empresa, elas procuravam obter o máximo de detalhes sobre a forma que a empresa fazia uso para obter os dados;
- **Quadro Kanban:** O kanban foi adotado para organizar todos os entregáveis do projeto. Quando o escopo do projeto foi definido o quadro foi idealizado para ajudar no cumprimento do cronograma.

[3] Encontrado em: http://bit.ly/2mnCBxs

4

Desenvolvimento

Este capítulo deve informar o problema e a solução proposta com o intuito de contextualizar o tema proposto. Este capítulo está divido em três seções: Modelo Atual, Modelo Esperado, e Resultados e Discussões, onde cada seção terá uma descrição geral de seu conteúdo e uma contextualização com o trabalho proposto.

4.1 Modelo Atual

Por motivos de sigilo e estratégia não foram reveladas informações privadas sobre a empresa, e por isso será adotado o nome fictício para fins didáticos neste trabalho como empresa "Alfa".

A empresa de tecnologia Alfa está passando por uma situação em que as solicitações de relatórios para o seu banco de dados acabam degradando o desempenho do servidor. Esta ocorrência pode ser caracterizada por deixar o seu processador em cem por cento de processamento, o uso de basicamente todos os seus 64 Gigabytes de memória RAM e também pressionando a matriz de discos a sua capacidade máxima. A Alfa não possui uma arquitetura bem definida, com isso, quero dizer que a separação do OLTP e OLAP não está sendo feita.

Foi entregue um questionário para a Alfa com várias perguntas sobre o seu cotidiano, infraestrutura, tecnologias, capacidade do servidor e outras indagações. Abaixo temos a Tabela 2 listando as principais características de infraestrutura da empresa.

A Alfa em seu servidor de banco de dados dispõe de uma quantidade de armazenamento físico disponível para o trabalho, memória RAM e a utilização de RAID. O fato de ter dois servidores pode ser um bom indício para implantar uma solução onde cada servidor vai abrigar uma camada, sendo elas OLTP e OLAP. Porém, um contraponto importante é a ausência de normalização no seu banco de dados. Elmasri (2008) cita que é importante ter um banco de dados normalizado para evitar falhas no projeto, assim como também redundâncias desnecessárias.

Tabela 2 – Infraestrutura da Alfa

Pergunta	**Resposta**
Qual o porte da empresa?	Pequena empresa até 42 funcionários.
Quais os participantes entrevistados?	2 Sócios e um Gerente de Projetos
Quais banco de dados são utilizados?	Somente o *SQL Server 2016*.
Quantos servidores a empresa dispõe?	Dois servidores.
Qual sua hospedagem?	Somente física.
Quantidade de memória *Random Access Memory* (RAM)?	Entre 32 *Gigabytes* e 64 *Gigabytes*.
Quanto de armazenamento permanente?	Entre 1 *Terabyte* e 2 *Terabytes*.
Quanto de armazenamento é utilizado?	Até 100 *Gigabytes*.
Qual tecnologia de armazenamento foi adotada?	*Redundant Array of Independent Disks* (RAID).
Utiliza alguma forma normal no banco de dados?	Somente a 1FN.
A taxa de impressão de relatórios no mesmo servidor de aplicação?	16 relatórios por minuto em horários de pico.

A Tabela 2 representa uma característica comum entre várias empresas que é a concorrência das transações e relatórios em um mesmo servidor. Braghittoni (2017) afirma que esse tipo de solução implica diretamente em dois problemas. Primeiro, o servidor não é escalonável e com o passar do tempo terá problemas de desempenho, atualmente e a depender do relatório causa uma instabilidade tão alta nele que é preciso reiniciá-lo. Segundo, a satisfação do usuário tende a diminuir, pois o servidor irá responder mais lentamente para atender as suas solicitações.

Informações que descrevam o comportamento do cotidiano em seu servidor é interessante para complementar a problemática da empresa. Mostrar como os usuários utilizam o sistema é de suma importância, assim como também a taxa de utilização do sistema. A correta descrição desses requisitos irá definir qual tecnologia pode ser a melhor, sendo uma solução envolvendo um DW ou *Big Data* por exemplo. A Tabela 3 ilustra esse aspecto da empresa.

Tabela 3 – Cotidiano da Alfa

Pergunta	**Resposta**
Quantidade de acessos simultâneos no sistema?	Até 5000 usuários.
Quantidade de registros por tabela em média?	1,2 milhões de registros por tabela em média.
Existem tabelas com grandes quantidades de registros?	Sim, existe uma tabela com 45 milhões de linhas.
Existem tabelas com muitas colunas?	A maior tabela contém 122 colunas.
Qual a utilização de *Central Processing Unit* (CPU) do servidor?	Sim.
Picos de CPU quando solicitado a impressão de relatório?	Sim.
Sua aplicação costuma perder desempenho ao abrir relatórios?	Sim.
Fornece dados sumarizados ao cliente?	Sim.
As informações precisam ser atualizadas com qual frequência?	Menor atraso possível.
Como você definiria a taxa de operações DML por hora?	143.622 operações em média nos horários de pico.
Como você definiria a taxa de consultas por hora?	1.780.577 consultas em média nos horários de pico .

Todos esses aspectos descritos confirmam a existência do OLAP trabalhando em conjunto com o OLTP no mesmo servidor. Um servidor com um poder computacional aceitável não deve ter problemas de desempenho em virtude de uma taxa razoável de usuários simultâneos, e para-

lelamente realizando algumas operações cotidianas normais. Da mesma forma, a disponibilidade dos dados em tempo real em relatórios não implica em uma alta utilização de CPU. Portanto, a identificação e separação da arquitetura é uma solução para o problema.

4.1.1 Processo Analítico da Alfa

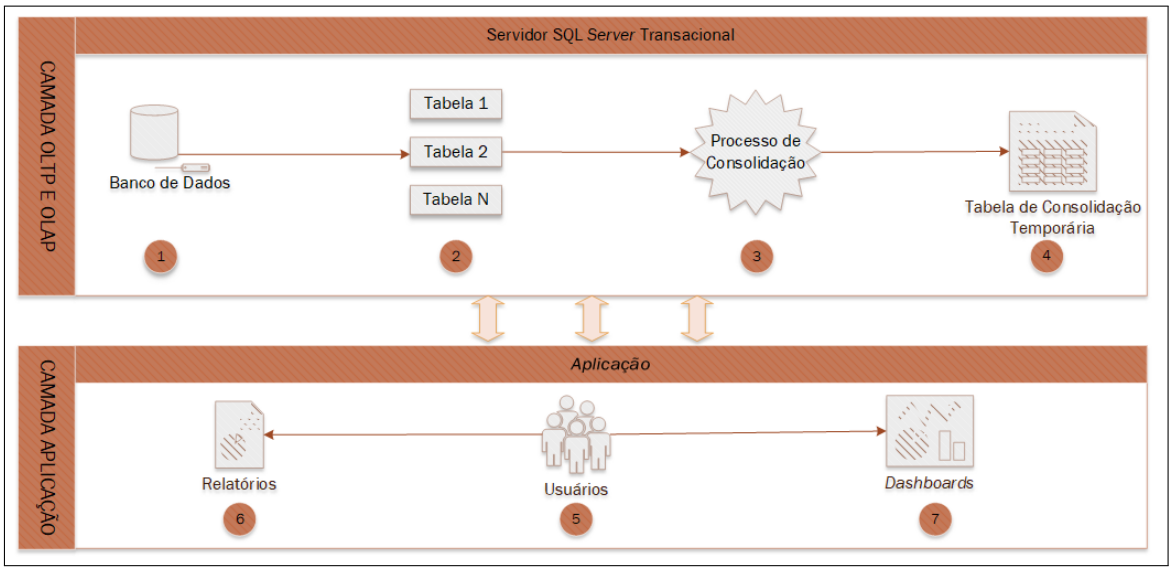
Suponha que uma empresa financeira deva somar todas as suas operações para obter o saldo diariamente, e a cada novo dia é preciso refazer os cálculos para mostrar o saldo corretamente, se caracterizando como um ciclo. Eventualmente, mais dados devem ser processados com o tempo e mais esforço deve ser feito para a efetuação do cálculo. Esse processo é análogo ao que ocorre na Alfa que é repetido sempre que o usuário solicita um determinado relatório, que resulta no agrupamento de diversas informações para impressão do relatório em tempo real a cada requisição. Donselaar (2015) Define tempo real como eventos que ocorreram no mundo real com respostas em questão de segundos, ou seja, a disponibilidade de um dado deve ser registrada com o mínimo de atraso possível.

Uma estratégia de recálculo é inadequada para uma boa utilização de recursos do servidor e a empresa Alfa faz uso desse princípio na exibição dos relatórios em tempo real. O impacto dessa abordagem está ligada diretamente ao servidor, porque vai deixar de atender as requisições em tempo hábil devido ao processamento do montante de dados.

Os componentes do processo analítico da empresa serão descritos a seguir conforme a Figura 4:

- **Item 1:** O Microsoft SQL Server Enterprise (64-bit) na versão 13.0.4224.16 hospedando o banco de dados da empresa;
- **Item 2:** Tabelas inerentes ao negócio da empresa, entretanto somente algumas tabelas foram mapeadas para o processo. Algumas características importantes a serem destacados são: a presença de diversas colunas em uma única tabela, tipos de dados inadequados, uso incorreto de índices e a praticamente ausência de normalização;
- **Item 3:** O processo de consolidação envolve a unificação de vários itens de diversas tabelas, esse processo é executado toda vez que o usuário solicita o relatório. No final, o processo irá gravar os dados em uma área temporária em disco. Esse processo é atualmente executado por um procedimento armazenado e tem diversos parâmetros de entrada;
- **Item 4:** Tabela onde os dados são temporariamente armazenados e com uma volatilidade altíssima. Uma chamada qualquer ao relatório que faz o processo de consolidação implica no preenchimento dessa tabela, esse processo é repetido várias e várias vezes dentro do banco de dados;
- **Item 5:** O usuário solicita as informações para a camada superior. A informação pode ser disponibilizada através de um relatório ou *dashboard*, respectivamente são os itens 6 e 7.

Figura 4 – Arquitetura Atual da Empresa Alfa



Como exibido na Figura 4 as operações analíticas são realizadas dentro do próprio servidor OLTP. Um ponto crítico nessa arquitetura é a geração dos fatos a todo momento que são consultados, essa forma de realizar consultas eleva a um nível muito alto o estresse no servidor. Então, essa arquitetura não oferece um grau de escalabilidade aceitável e vai apresentar gargalos.

4.2 Modelo Esperado

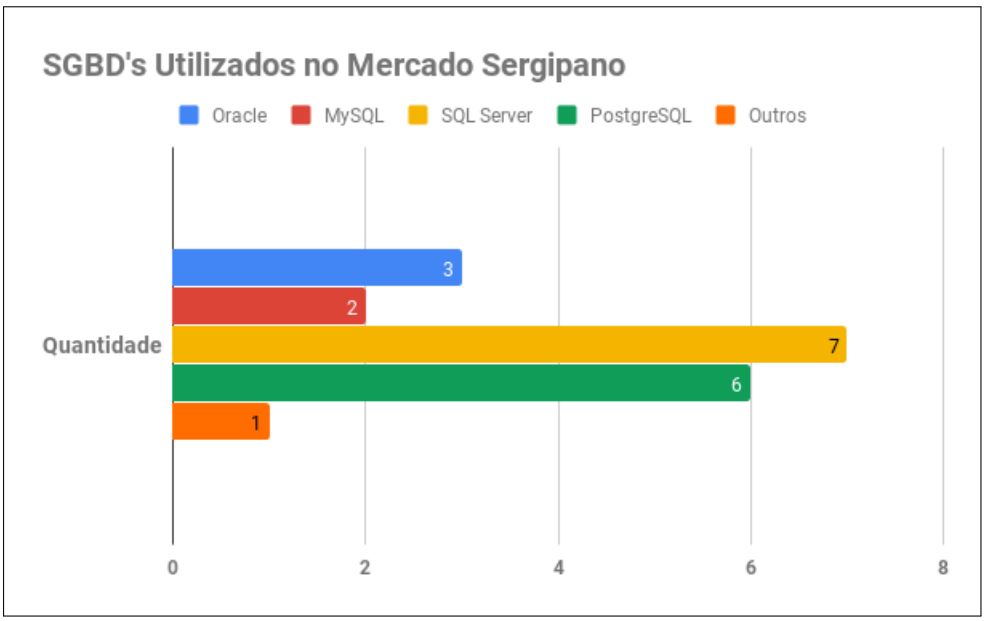
A arquitetura proposta para a Alfa envolve a elaboração do ambiente OLTP e OLAP. A ideia é criar um ambiente em tempo real para impressão de relatórios e que ao mesmo tempo os usuários realizem suas operações cotidianas. Impreterivelmente é de extrema importância uma solução que esteja dentro das definições do que é ser em tempo real. Assim, todos os meios necessários para uma arquitetura em tempo real devem ser adotados.

Cada ambiente deve ter o seu próprio servidor para um melhor desempenho. A Alfa tem dois servidores disponíveis para a implantação da solução. O ideal é que cada servidor seja responsável por cada ambiente, ou seja, um servidor fica responsável pelo OLTP e o outro pelo OLAP. Essa separação arquitetural retira a responsabilidade do banco de dados da aplicação de gerar relatórios, passando somente a processar as suas transações. Portanto, uma boa distinção entre cada camada se torna mais que obrigatória para um ambiente em tempo real.

A arquitetura proposta para a empresa Alfa foi elaborada sobre quatro pilares. O primeiro, o objetivo que a empresa Alfa almeja, que é a impressão de relatórios com um excelente desempenho. O segundo, a utilização do Microsoft SQL Server pela própria empresa. O terceiro, um procedimento armazenado cedido que teve o seu conteúdo averiguado, porém, não será exposto por motivos de sigilo. O quarto, a necessidade de disponibilidade das informações em tempo real.

A solução baseada é no escaneamento assíncrono do arquivo de *log* do banco dados SQL Server, mas a mesma tecnologia já existe para outros bancos. O SQL Server foi escolhido por ser o mais usado no mercado sergipano. A Figura 5 ilustra a afirmação.

Figura 5 – SGBD's Utilizados no Mercado Sergipano



Uma arquitetura em tempo real para impressão de relatórios para a empresa alfa deve ser entregue com três camadas principais. As camadas são: a OLTP, a OLAP e a aplicação. Uma visão geral da ideia é descrita conforme a Figura 6.

4.2.1 Camada OLTP

A fonte oriunda de todas as informações é o banco de dados SQL Server, ele é núcleo principal dessa camada. A camada foi elaborada usando somente um banco de dados, mas um outro banco de dados pode ser anexado ao processo facilmente. Uma coisa importante é que a camada foi projetada para funcionar somente com um banco de dados SQL Server, entretanto existem implementações de CDC em outros SGBD's.

A principal função dessa camada é fornecer os dados, logo os dados devem ser capturados da maneira mais rápida possível e com o menor impacto possível no servidor. Além disso o usuário é responsável por gerar uma ou várias transações no sistema, isso implica na necessidade atualizar o DW, porém, não é de sua responsabilidade realizar essa operação.

Para captura de dados em uma tabela é preciso habilitar primeiramente o CDC no banco de dados, isso é feito executando um procedimento armazenado sobre o banco em questão. A Figura 7 usa o banco de dados "Alfa_OLTP" e nele é executado o procedimento "sys.sp_cdc_enable_db". O banco de dados a partir desse ponto deve permitir a captura de dados para as tabelas especificadas.

Figura 6 – Arquitetura em Tempo Real

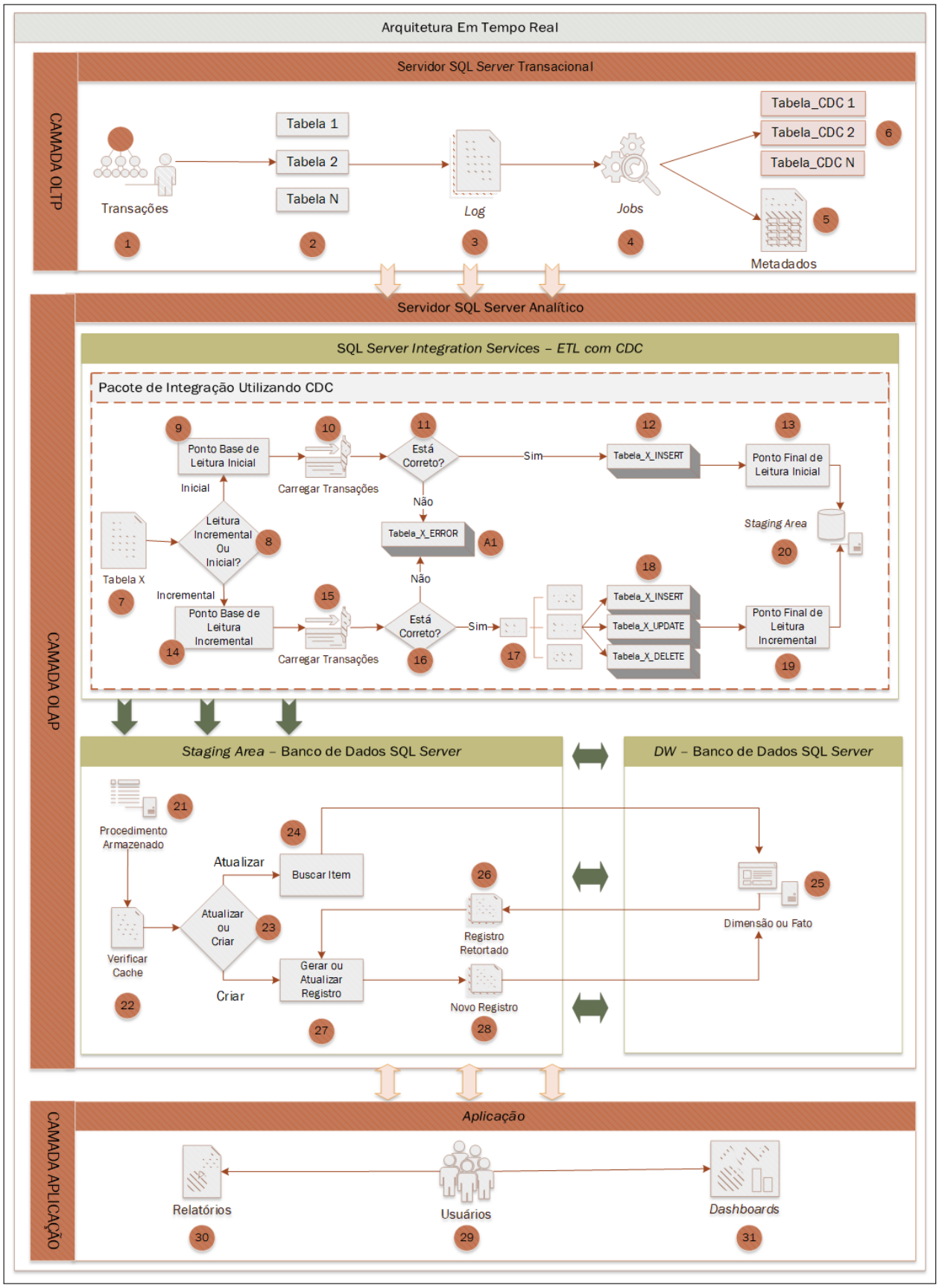
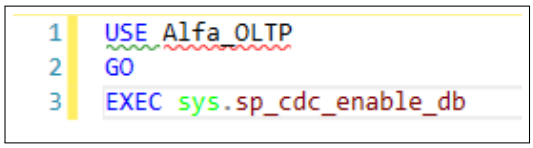


Figura 7 – Habilitar CDC no Banco de Dados



```
1  USE Alfa_OLTP
2  GO
3  EXEC sys.sp_cdc_enable_db
```

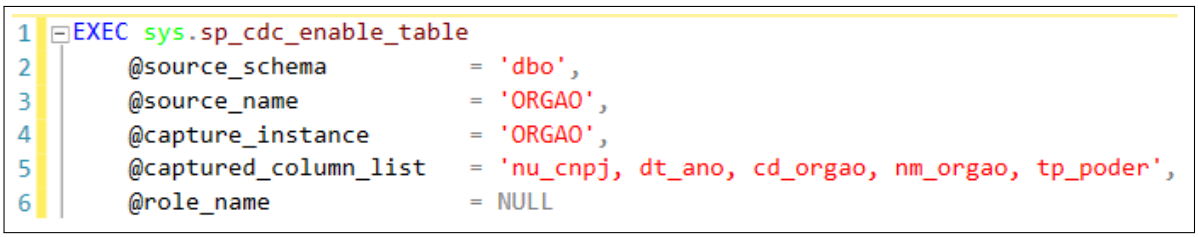
O banco de dados contém todas as tabelas inerentes ao fluxo e neste mesmo banco existem diversas tabelas, mas somente quatorze tabelas fazem parte do fluxo. As tabelas foram identificadas com um olhar minucioso sobre um procedimento armazenado oferecido e habilitadas para captura de dados com CDC.

Cada uma das quatorze tabelas identificadas foram habilitadas para captura de dados com CDC. Novamente um outro procedimento armazenado deve ser executado para a monitoração das tabelas, e o seu nome é “sys.sp_cdc_enable_table”. Existem diversos parâmetros obrigatórios e facultativos para ele, mas, só serão utilizados quatro deles para a configuração. Esses parâmetros são os seguintes:

- **source_schema:** Nome do *schema* da tabela cujo CDC será habilitado e geralmente esse valor é "dbo", que é justamente o *schema* padrão do banco de dados;
- **source_name:** Nome da tabela que o CDC será habilitado;
- **captured_column_list:** lista de colunas que serão rastreadas na tabela e somente as colunas utilizadas no processo foram especificadas para uma melhor otimização. Caso esse parâmetro não for informado, todas as colunas da tabela serão mapeadas por padrão;
- **role_name:** O propósito da *role* é controlar o acesso aos dados de alteração, este parâmetro é obrigatório, porém, pode receber o valor *NULL*.

Uma tabela foi escolhida para exemplificar essa etapa de configuração e a tabela em questão foi a “ORGAO”. Com ela foram especificadas cinco colunas para o andamento do processo, lembrando que a tabela contém várias colunas, porém, só foram escolhidas as necessárias. Por fim, será criada uma *Change Table* (CT) com a seguinte nomenclatura: ORGAO_CT. A Figura 8 demonstra o processo de configuração por completo.

Figura 8 – Habilitar CDC na Tabela



```
1  EXEC sys.sp_cdc_enable_table
2      @source_schema          = 'dbo',
3      @source_name            = 'ORGAO',
4      @capture_instance       = 'ORGAO',
5      @captured_column_list   = 'nu_cnpj, dt_ano, cd_orgao, nm_orgao, tp_poder',
6      @role_name              = NULL
```

O *log* de transação do banco de dados SQL *Server* vai armazenar todas as informações de cada transação. Ao habilitar uma tabela com o CDC um *job* ficará responsável por escaneá-la assincronamente, e após o escaneamento de cada transação referente a tabela, um registro de cada transação é gravado na tabela espelho e também uma série de metadados.

Quando o Alfa_OLTP foi ativado para CDC automaticamente foram criados dois *jobs*, cada um possui uma tarefa distinta, porém, crucial. O primeiro *job* "cdc.Alfa_OLTP_capture" é responsável por buscar as informações contidas no *log* de transação. O segundo *job* é o "cdc.Alfa_OLTP_cleanup", ele está associado ao processo de expurgo dos dados salvos na CT. Os dois *jobs* serão descritos com mais detalhes logo abaixo:

- **cdc.Alfa_OLTP_capture:** *Job* delegado para analisar os dados dentro do *log* de transação. Para efeitos de maximização da arquitetura em tempo real, o intervalo entre cada escaneamento no logo foi definido para um segundo, por padrão o escaneamento do *log* de transação ocorre a cada 5 segundos;
- **cdc.Alfa_OLTP_cleanup:** Esse *job* é encarregado por expurgar a massa de dados capturadas pelo CDC. O período entre cada expurgo pode ser configurada, e o período entre cada limpeza tem o intervalo de três dias por padrão.

Como já citado, ao ativar uma tabela para o CDC um conjunto de metadados são atrelados a mesma também. Os principais metadados serão listados a seguir:

- **__$start_lsn:** Valor do *log* antes do início da transação ser efetuada;
- **__$end_lsn:** Valor do *log* após a conclusão da transação;
- **__$seqval:** Identificador responsável por ordenar as transações. Caso uma transação gere mais de uma linha, esse identificador será responsável por enumerar todas as linhas unicamente;
- **__$operation:** Indica qual comando DML foi efetuado na transação. Os comandos disponíveis são: 1 - Delete, 2 - Inserir, 3 - Antes de Atualizar e 4 - Depois de Atualizar;
- **__$update_mask:** Representa quais colunas foram alteradas pela instrução de atualização e o formato da coluna é armazenado em um formato binário. Caso ocorra um comando de inserir ou deletar todas as colunas serão mostradas.

Cinco operações DML's serão submetidas para ilustrar o processo de geração de metadados e captura dos dados. A tabela "ORGAO" terá em sequência sendo executados os seguintes comandos: inserir três órgãos com os seguintes nomes: Gama, Delta, Épsilon; remover o órgão Delta e atualizar o nome do órgão Épsilon para Beta. O resultado dessa sequência de comandos é observado na Figura 9.

Figura 9 – Metadados e Dados Gerados na Tabela Orgão

__$start_lsn	__$end_lsn	__$seqval	__$operation	__$update_mask	nm_orgao
0x00030d2f000531b40004		0x00030d2f000531b40002	2	0x1f	Gama
0x00030d2f000531b60004		0x00030d2f000531b60002	2	0x1f	Delta
0x00030d2f000531b80004		0x00030d2f000531b80002	2	0x1f	Épsilon
0x00030d2f000540b30001		0x00030d2f000540900017	1	0x1f	Delta
0x00030d2f000540ea0001		0x00030d2f000540e90002	3	0x08	Épsilon
0x00030d2f000540ea0001		0x00030d2f000540e90002	4	0x08	BETA

4.2.2 Camada OLAP

Nesta camada, o seu objetivo principal é processar as informações provenientes da camada OLTP. A ferramenta ETL adotada foi o SSIS da Microsoft e possui uma fácil integração com o CDC e foi escolhido por esse motivo. No SSIS foi definido dois pacotes para cada uma das tabelas identificadas no processo. Um pacote é responsável por uma carga inicial e o outro faz a carga incremental. Ao fim de cada pacote é encaminhado para a *Staging Area* para executar um procedimento armazenado. Alguns outros tipos de pacotes também foram definidos para otimização do fluxo. Por fim, as informações são enviadas para o DW.

4.2.2.1 SSIS

O primeiro grande componente dessa camada é o ETL e por estar utilizando do início ao fim ferramentas da Microsoft o processo de integração ficou mais fácil. A própria ferramenta disponibiliza uma diagramação de fácil utilização em cima do CDC. Ao todo foram criados trinta e dois pacotes. Esses pacotes podem ser classificados da seguinte forma: Incrementais, Base, Configuração, Geração e Paralelismo.

Um pacote base segue basicamente a mesma lógica entre todos, cada pacote apaga os dados da *Staging Area*, registra o ponto de leitura inicial, importa e verifica os dados do OLTP, registra o ponto final de leitura e encaminha para o processamento. A Figura 10 ilustra a visão geral de um pacote base.

Uma perspectiva mais detalhada de um pacote base é mostrada na Figura 11 e a importação dos dados relacionados a tabela mapeada com CDC é efetuada. Validações e conversões são realizadas nas informações presentes, nesta mesma etapa registros inválidos são enviados para uma área de erro e também novos dados são gerados a partir de algumas colunas. E no final do processo a *Staging Area* é acionada.

O principal objetivo do pacote incremental é ser executado a cada ciclo o mais rápido possível. A perspectiva externa de um pacote incremental difere do pacote base em alguns pontos. Um pacote incremental executa paralelamente a limpeza das tabelas de *INSERT*, *UPDATE* e *DELETE*, adicionalmente um ponto de controle CDC inicial é registrado, antes desse ponto de controle ser executado o pacote base correspondente deve ser executado, caso contrário,

Figura 10 – Exemplo de Pacote Base Visão Geral

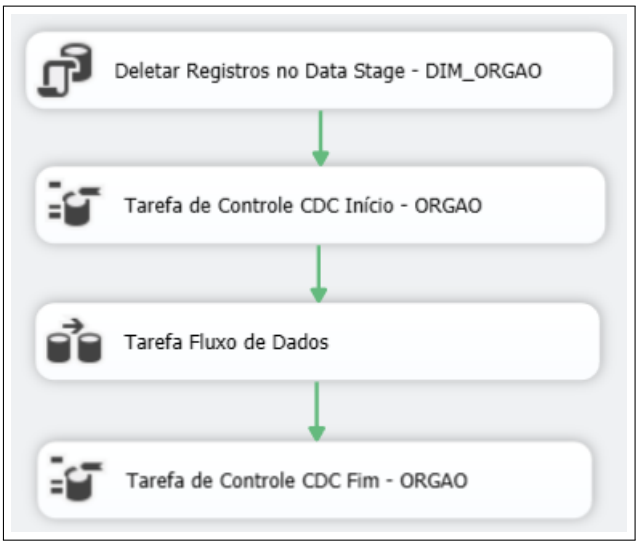
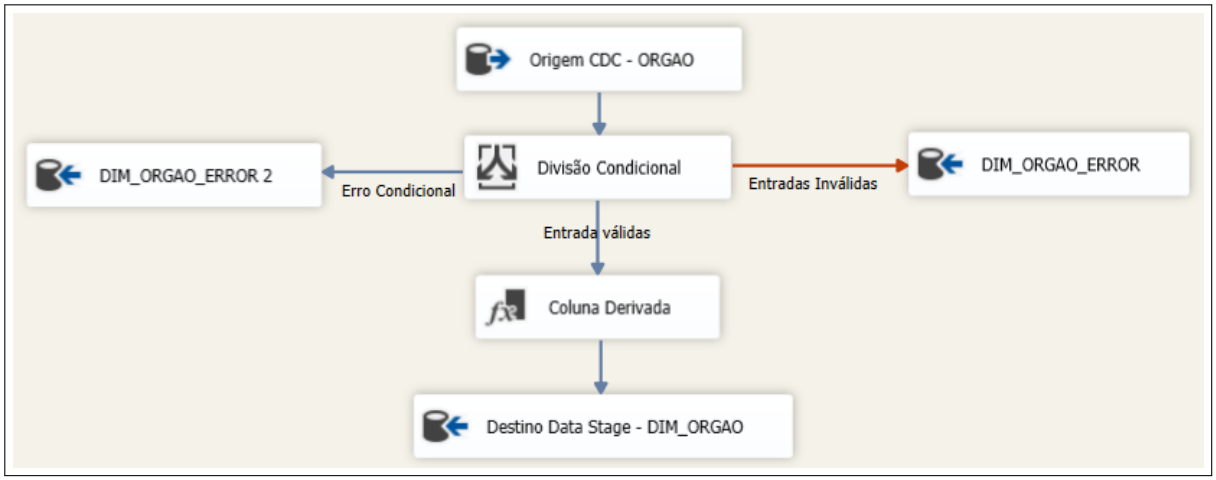


Figura 11 – Exemplo de Pacote Base Visão Interna

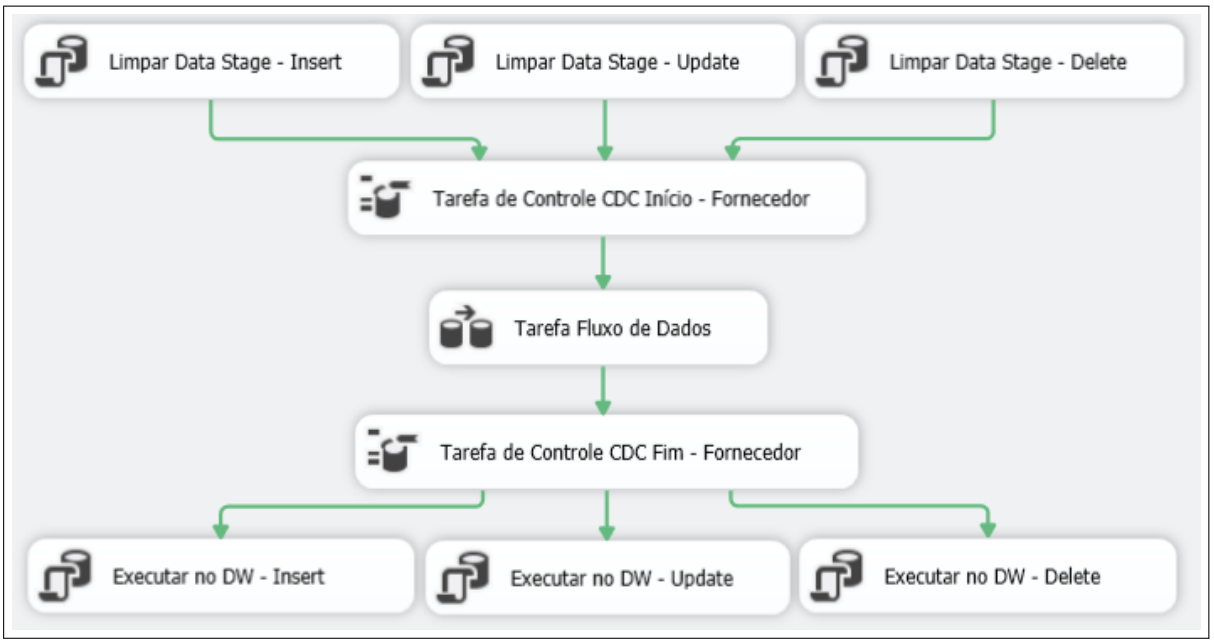


ocorrerá um erro. O fluxo de dados do pacote é executado e seguido do controle de CDC final, Posteriormente a *Staging Area* irá processar em paralelo os comandos de inserir, atualizar e deletar. Por fim, os dados serão enviados para o DW. Esse fluxo do pacote incremental é representado pela Figura 12

A perspectiva interna de um pacote incremental tem dois pontos importantes. Primeiro, a fonte de dados a ser consumida é a CT ao invés da tabela original e segundo as operações DML's serão decompostas. Outro fato importante sobre esse pacote é como ele processa diversas mudanças em um único registro, isto implica em uma única alteração resultante após múltiplas modificações em um único registro para encaminhamento.

Múltiplas alterações em um registro vão resultar em uma única linha. Caso o CDC capture diversas alterações em um único registro durante um determinado ciclo, somente a última instância do registro será levada em conta. Esse comportamento pode ser configurado, mas para a empresa Alfa só é interessante a última alteração do registro.

Figura 12 – Exemplo de Pacote Incremental Visão Externo



Um cenário de uso sobre múltiplas alterações em um único registro deve ser descrito para melhor entendimento, para isso será demonstrado várias alterações em um fornecedor fictício. O fornecedor com o nome "João Moura Nascimento" teve o seu nome renomeado para "João Moura de Nascimento", porém, outra atualização foi necessária porque o seu nome é "João Moura do Nascimento". No exemplo citado duas atualizações foram realizadas sobre o fornecedor, mas somente a última será encaminhada para processamento.

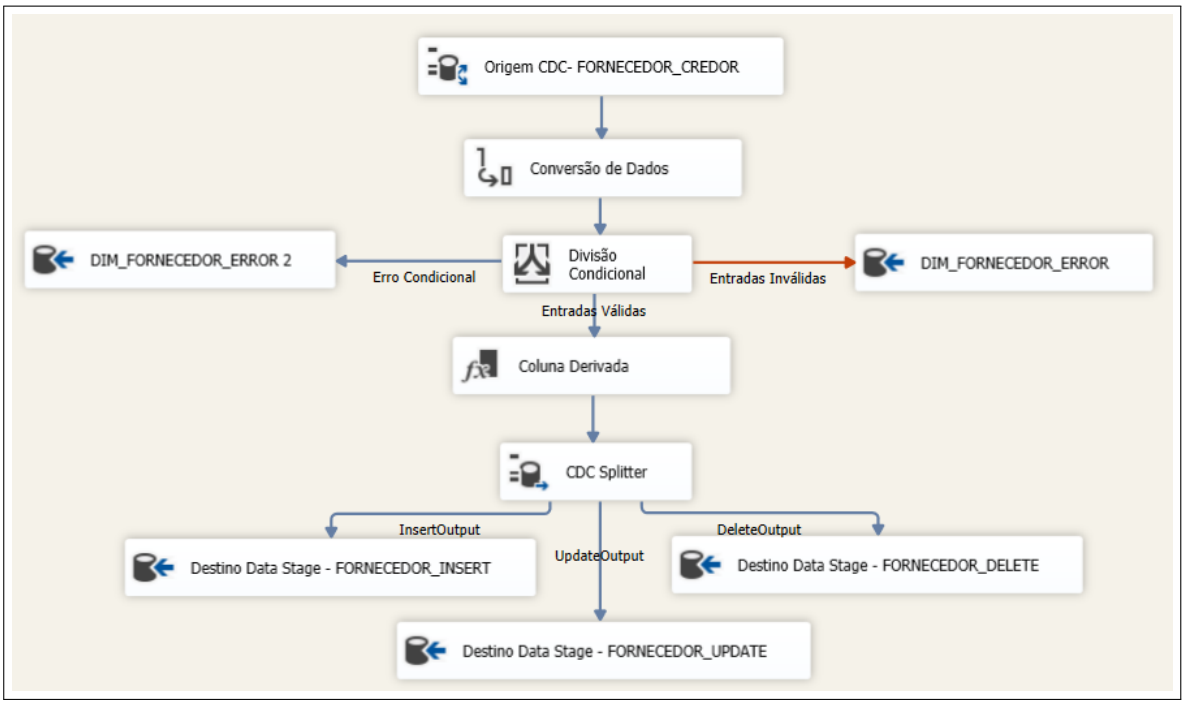
Todas as operações DML's serão separadas para processamento, este é o segundo ponto importante do pacote incremental, visto que cada transação realizada na camada OLTP pode ser classificada como: inserir, atualizar e deletar. Um componente do SSIS é destinado a realizar a divisão e após isso cada transação é devidamente encaminhada para a *Staging Area*.

Um pacote de configuração é utilizado principalmente na carga inicial, este tipo de pacote está atrelado a duas tarefas principais. A primeira, ao fazer a carga inicial todas as restrições do banco de dados do DW são removidas, isso é feito para acelerar o processo de carga do DW. A segunda, todos os índices criados são removidos, isso aumenta performance da carga inicial e também evita que os índices fiquem fragmentados. Ao final do processo de carga inicial todas as configurações desabilitadas ou retiradas devem ser restauradas.

A tabela fato deve ser somente carregada após cada dimensão for preenchida corretamente e também foi designado um pacote específico para realizar este procedimento que é o pacote de geração. Existe somente um pacote desse tipo que será somente executado na carga inicial. Outro detalhe importante é que na carga incremental a atualização da tabela fato está embutida em cada pacote.

Para atingir o objetivo de uma arquitetura em tempo real os pacotes não podem ser

Figura 13 – Exemplo de Pacote Incremental Visão Interna



executados de forma sequencial. Com o intuito de ter um menor ciclo entre cada execução um pacote de paralelismo foi adicionado ao SSIS, este é o último tipo de pacote e sua função é agrupar e executar todos os pacotes incrementais em paralelo. Uma visão geral desse pacote é descrita na Figura 14.

Outra consideração importante sobre o pacote de paralelismo é o que deve ser executado em paralelo. um objeto centralizador foi definido para ajudar na tarefa, onde pacotes acima do centralizador processam mudanças que afetam as tabelas de dimensões, e pacotes abaixo do centralizador processam informações que refletem na tabela de fato. Esta distinção foi feita para garantir que as informações estejam coerentes ao passarem para o DW.

O processamento do ETL está dividido entre dois fluxos e cada fluxo tem um conjunto de pacotes associados. O primeiro fluxo é o de carga inicial, ele é responsável por ter tomar todas as medidas necessárias para garantir a primeira carga inicial no DW. O segundo fluxo é chamado de fluxo de carga incremental, este garante que as decorrentes alterações do OLTP sejam processadas, por fim, basicamente para cada tabela identificada no processo existe dois pacotes associados a ela . A Tabela 4 lista a maioria dos pacotes criados no ETL através do SSIS.

Para cada pacote presente no fluxo inicial ou incremental uma nomenclatura foi adotada. No fluxo inicial os pacotes começam com o prefixo “INIT”, enquanto nos pacotes incrementais iniciam com o prefixo “INC”. Uma numeração para fins de ordenação foi adotada para cada pacote, sendo que em alguns pacotes um prefixo “F” pode aparecer antes da numeração, este “F” representa que essa tabela gera informações para uma tabela fato. Ao final da numeração temos o nome da tabela associada, dessa forma estabelecendo um padrão para os pacotes.

Figura 14 – Pacote de Execução em Paralelo

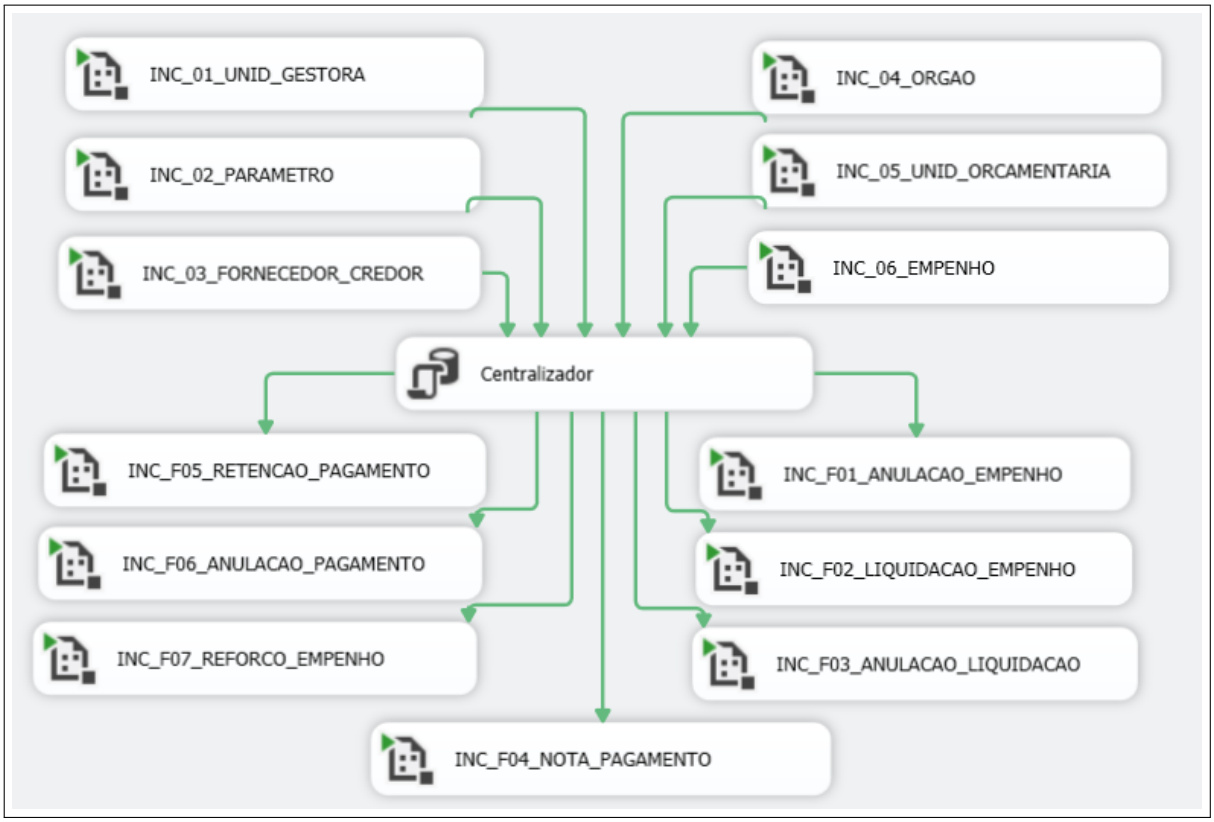


Tabela 4 – Tabela Vs Fluxo

Tabela	**Fluxo Inicial**	**Fluxo Incremental**
Unid_Gestora	INIT_01_Unid_Gestora	INC_01_Unid_Gestora
Parametro	INIT_01_Unid_Gestora	INC_02_Parametro
Fornecedor	INIT_03_Fornecedor	INC_03_Fornecedor
Orgao	INIT_06_Acao	INC_04_Orgao
Unid_Orcamentaria	INIT_06_Acao	INC_05_Unid_Orcamentaria
Acao	INIT_06_Acao	INC_06_Acao
Empenho	INIT_07_Empenho	INC_07_Empenho
Anulacao_Empenho	INIT_F01_Anulacao_Empenho	INC_F01_Anulacao_Empenho
Liquidacao_Empenho	INIT_F02_Liquidacao_Empenho	INC_F02_Liquidacao_Empenho
Anulacao_Liquidacao	INIT_F03_Anulacao_Liquidacao	INC_F03_Anulacao_Liquidacao
Nota_Pagamento	INIT_F04_Nota_Pagamento	INC_F04_Nota_Pagamento
Retencao_Pagamento	INIT_F05_Retencao_Pagamento	INC_F05_Retencao_Pagamento
Anulacao_Pagamento	INIT_F06_Anulacao_Pagamento	INC_F06_Anulacao_Pagamento
Reforco_Empenho	INIT_F07_Reforco_Empenho	INC_F07_Reforco_Empenho

No fluxo inicial algumas tabelas apresentam o mesmo pacote de execução. O motivo para essa abordagem é que no DW estas tabelas foram unificadas em uma única dimensão, ou seja, uma tabela desnormalizada foi criada com base nessas tabelas. O mesmo pacote também foi associado a essas tabelas que interagem com várias tabelas durante a sua execução.

Todos os pacotes descritos até agora estão diretamente ligados a uma ou várias tabelas, porém existem outros pacotes que não correspondem a esta afirmação. Ao total são cinco pacotes adicionais para interação durante o processo, estes pacotes foram criados para facilitar e

centralizar uma determinada responsabilidade. A Tabela 5 lista todos os pacotes com uma breve explicação sobre cada um.

Tabela 5 – Pacotes Restantes

Nome	**Fluxo**	**Descrição**
INIT_00_BEGINNING	Inicial	Remover os índices e as restrições do banco.
INIT_Z98_DATE	Inicial	Preencher a dimensão de data no DW.
INIT_Z99_GENERATE_FACTS	Inicial	Preencher a tabela de fato no DW.
INIT_Z999_ENDING	Inicial	Restaurar os índices e as restrições do banco.
INC_PARALLEL	Incremental	Executar os pacotes em paralelo.

4.2.2.2 *Staging Area*

A *Staging Area* foi projetada de acordo com o modelo citado por Braghittoni (2017), ele define que qualquer tabela pertencente a *Staging Area* deve ter o tipo de dados *varchar* (N). Essa abordagem é útil porque a *Staging Area* deve armazenar todas as informações oriundas da camada OLTP, visto que não deve impor nenhum mecanismo de validação aos dados.

Uma coluna com o tipo de dados *varchar* (N) precisa ter o seu tamanho máximo definido. Para cada coluna da tabela com esse tipo de dados foi analisada a coluna correspondente na tabela origem, a ideia é analisar o quanto deve ser o tamanho adicional com base na coluna da camada OLTP. Basicamente para um campo como CPF que possui onze dígitos um coluna de quinze dígitos foi criada para ela, esse tipo de campo não tem uma variação de tamanho entre os seus dados, justificando assim o seu acréscimo. Entretanto para um campo dinâmico onde o tamanho do dado pode alternar consideravelmente, um adicional de vinte dígitos foi definido.

Cada tabela presente na camada OLTP contém quatro tabelas associadas dentro da *Staging Area*. Um determinado sufixo é acoplado ao nome de cada tabela para identificar a sua finalidade, os sufixos disponíveis são: INS, UPD, DEL e ERR. Além desses, existe um outro sufixo chamado de CAC que será explicado posteriormente.

Uma descrição do significado implícito de cada sufixo deve ser descrita, temos os seguintes sufixos:

- **INS:** Indica a tabela vai armazenar todas as operações de *INSERT* da camada OLTP;
- **UPD:** Sufixo que informa que a tabela em questão tem em suas linhas somente operações de *UPDATE*, as linhas da tabela também são oriundas da camada OLTP;
- **DEL:** Uma tabela que contenha esse sufixo deve guardar somente transações de *DELETE*, assim como o item anterior suas informações são provenientes da camada OLTP;
- **ERR:** Armazena todas as linhas que falharam durante o processo de ETL. A linha presente nessa tabela pode representar duas informações distintas, uma que indica verificação condicional e outra que indica inconsistência. A verificação condicional é resultante de

uma instrução programada, enquanto a falha é proveniente de exceção ao processar aquela linha, este tipo de tabela é alimentada pelo ETL, divergindo dos outros tipos.

Com os principais tipos descritos agora é preciso exibir as principais tabelas presentes na *Staging Area*. A Tabela 6 apresenta as principais tabelas presentes nessa etapa.

Tabela 6 – Principais Tabelas da *Staging Area*

Inserir	**Atualizar**	**Deletar**	**Erro**
Unid_Gestora_INS	Unid_Gestora_UPD	Unid_Gestora_DEL	Unid_Gestora_ERR
Parametro_INS	Parametro_UPD	Parametro_DEL	Parametro_ERR
Fornecedor_INS	Fornecedor_UPD	Fornecedor_DEL	Fornecedor_ERR
Orgao_INS	Orgao_UPD	Orgao_DEL	Orgao_ERR
Unid_Orcamentaria_INS	Unid_Orcamentaria_UPD	Unid_Orcamentaria_DEL	Unid_Orcamentaria_ERR
Acao_INS	Acao_UPD	Acao_DEL	Acao_ERR
Empenho_INS	Empenho_UPD	Empenho_DEL	Empenho_ERR
Anulacao_Empenho_INS	Anulacao_Empenho_UPD	Anulacao_Empenho_DEL	Anulacao_Empenho_ERR
Liquidacao_Empenho_INS	Liquidacao_Empenho_UPD	Liquidacao_Empenho_DEL	Liquidacao_Empenho_ERR
Anulacao_Liquidacao_INS	Anulacao_Liquidacao_UPD	Anulacao_Liquidacao_DEL	Anulacao_Liquidacao_ERR
Nota_Pagamento_INS	Nota_Pagamento_UPD	Nota_Pagamento_DEL	Nota_Pagamento_ERR
Retencao_Pagamento_INS	Retencao_Pagamento_UPD	Retencao_Pagamento_DEL	Retencao_Pagamento_ERR
Anulacao_Pagamento_INS	Anulacao_Pagamento_UPD	Anulacao_Pagamento_DEL	Anulacao_Pagamento_ERR
Reforco_Empenho_INS	Reforco_Empenho_UPD	Reforco_Empenho_DEL	Reforco_Empenho_ERR

Ações devem ser tomadas para atualizar as informações presentes no DW, porém, nem todas as informações residem nele para dar suporte. Uma dimensão ás vezes não contém todas as informações associadas a uma tabela do OLTP, nessas situações uma área especial foi criada para guardar e dar suporte a essas condições. O sufixo “CAC” presente a uma tabela indica que ela é utilizada como um mecanismo de cache de dados, deste modo as tabelas que levam esse sufixo são: Parametro, Liquidacao_Empenho, Nota_Pagamento, Orgao e Unid_Orcamentaria.

A nomenclatura das colunas presentes nas tabelas seguem um padrão, a nomenclatura consiste em utilizar um prefixo antes de cada coluna da tabela. Essa abordagem foi utilizada porque a empresa Alfa segue esse padrão para modelagem do seu banco de dado. A Tabela 7 abaixo lista todos os prefixos usados.

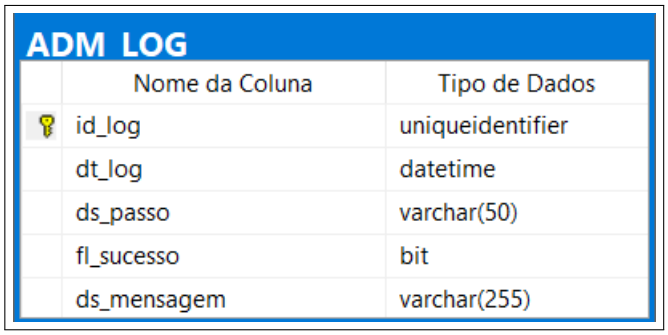
Tabela 7 – Tabela de Prefixos

Nome	**Significado**
id	Identificador único para a tabela associada e é um valor inteiro.
dt	Representa que a coluna é do tipo *DateTime*.
ds	A coluna é do *varchar*(N) e é uma coluna para descrição.
nu	A coluna em questão é do tipo inteiro curto ou longo.
sq	Um número sequencial qualquer podendo ser um inteiro curto ou longo.
fl	Identifica que essa coluna é do tipo *bit*.
nm	A coluna representa um nome de algum objeto.
tp	Representa o tipo que a entidade deve ser.
vl	Representa que um valor numérico está associado a coluna.
cd	Um código é atribuído ao valor do campo.

Uma política de *log* foi implantada dentro da *Staging Area*. Uma tabela chamada ADM_LOG vai registrar o que ocorreu em cada procedimento armazenado executado, a informa-

ção deve ser retida mesmo se o procedimento armazenado desempenhar seu papel corretamente ou erroneamente. A tabela contém as seguintes colunas: id_log, identificador único para a tabela; dt_log, data em que o registro foi criado; ds_passo, breve descrição da etapa que está sendo executada; fl_sucesso, indica se a operação foi executada com sucesso; ds_mensagem, descrição completa da etapa. A Figura 15 mostra a definição geral dessa tabela.

Figura 15 – Tabela ADM_LOG



Os últimos componentes presentes nessa camada interna são os procedimentos armazenados. O procedimento armazenado é executado para garantir que as informações sejam processadas e enviadas para o DW e algumas decisões sejam tomadas, eles são responsáveis por atualizar as informações no DW, remover informações errôneas do DW, inserir novas informações e verificações de dados em cache.

Um conjunto de procedimentos armazenados foram definidos para cada fluxo. Uma quantia de dezoito procedimentos criados no fluxo inicial, e trinta e nove procedimentos para o fluxo incremental, totalizando cinquenta e seis procedimentos presentes na *Staging Area*.

4.2.2.3 *Data Warehouse*

O *Data Warehouse* para a empresa alfa foi projetado para ter o melhor desempenho possível. Em geral temos um DW hospedado em um banco de dados SQL Server em conjunto com o modelo estrela para o relacionamento entre a tabela fato e suas dimensões. Com isso, temos um DW definido para dar suporte a arquitetura em tempo real.

Cada dimensão presente no DW foi implementada com SCD2, com exceção somente da dimensão de data porque não é necessário. Ao implementar o SCD2 para uma dimensão o histórico é preservado, entretanto, caso a informação seja cadastrada erroneamente o registro será apagado e inserido novamente. Existem também outras características presentes nas dimensões que são: a presença de uma chave primária baseada em um inteiro incremental, a desnormalização de algumas dimensões para um melhor desempenho, algumas dimensões apresentam autorrelacionamento com o propósito de formar uma hierarquia, a existência de colunas redundantes para acelerar o fluxo incremental e a configuração de índices para melhorar a performance de buscas através do SSIS.

O DW apresenta somente uma tabela fato e novamente as boas práticas foram aplicadas a esta tabela, ou seja, a tabela armazena somente valores que podem ser mensurados e também juntamente com os identificadores relacionados a cada dimensão. Uma consideração importante sobre essa tabela é que seus dados nunca são atualizados, quando um usuário realizar uma operação de atualização no OLTP essa atualização se transforma em um deletar decorrente de um inserir. Em suma, a tabela fato tem todas as movimentações ou valores que serão utilizados para realizar consultas com base no saldo.

Ao todo foram definidas seis dimensões e uma fato para o DW. A Figura 16 exibe uma visão geral do DW, assim como também o relacionamento entre as tabelas.

4.2.3 Camada Aplicação

A camada de aplicação é simples e nela são criadas as consultas para posteriormente enviá-las para a camada OLAP. A estruturação da camada em si pode ser feita por qualquer *framework* de consulta e que pode ser definido de acordo com a necessidade da empresa. A maneira de comunicação entre essa camada e a OLAP deve ser transparente ao usuário, ou seja, configurações adicionais feitas pelo usuário não podem estar no escopo. Então, prover um mecanismo para forjar consultas e ao mesmo tempo receber os dados da consulta é desejável.

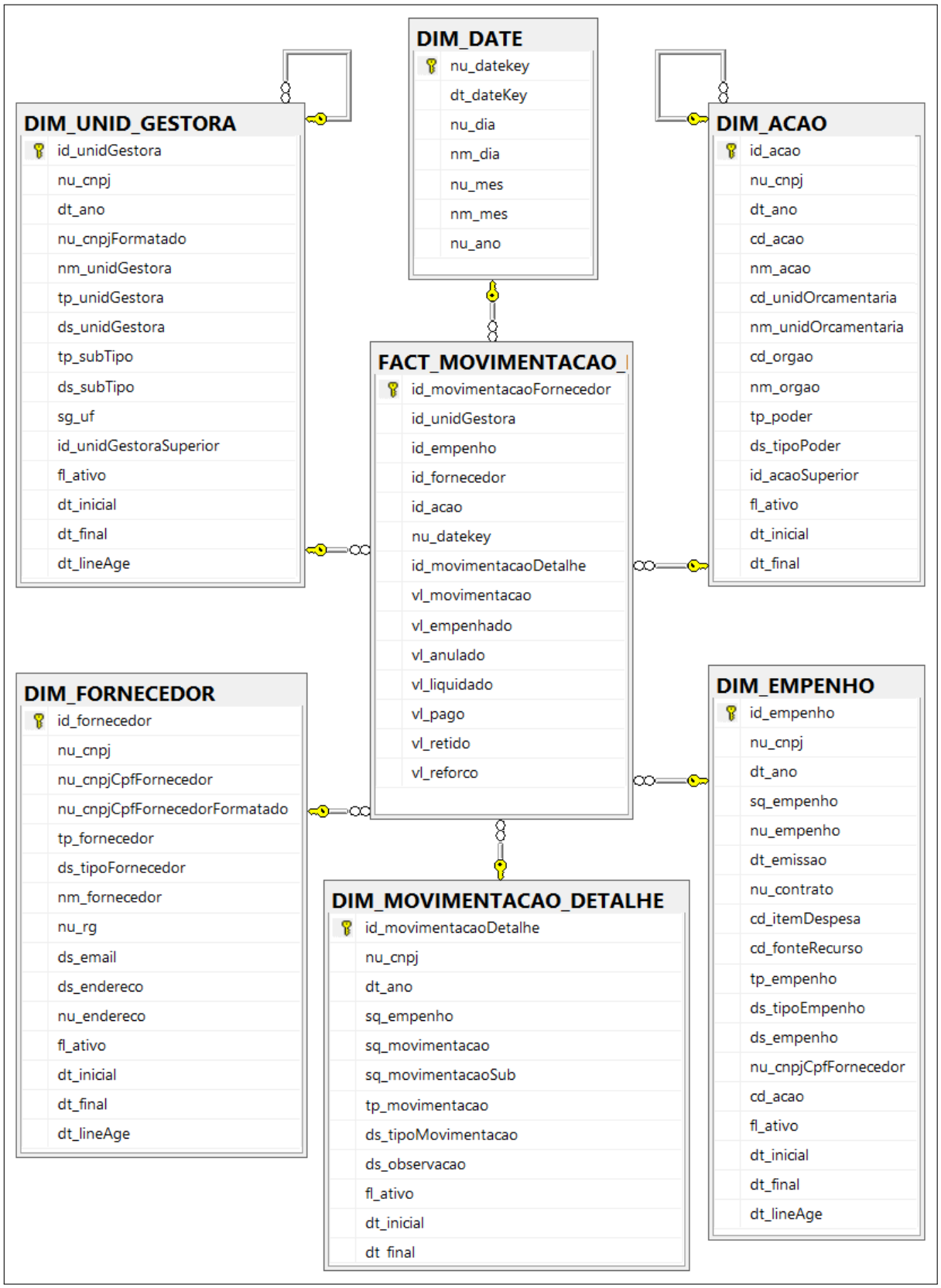
O consumo de informações é realizado através da execução de procedimentos armazenados definidos na aplicação. Um procedimento deve ter um conjunto de parâmetros para que o usuário possa solicitar o relatório de diversas formas, isso possibilita a filtragem das informações em numerosas formas para o usuário. Sempre que o cliente demandar uma informação em uma perspectiva diferente um novo procedimento deve ser concebido, resultando na manutenção de diversos procedimentos.

A camada de aplicação deve se comunicar unicamente com a camada de OLAP. A comunicação é bilateral entre elas, isto é, a camada de aplicação envia solicitações para OLAP e a OLAP envia os dados para aplicação. Essa separação de interesses é importante para um melhor entendimento dos seus elementos e também para uma eventual manutenção. Em síntese, para uma arquitetura em tempo real uma organização adequada entre cada componente é indispensável.

4.3 Resultados e Discussões

Um ambiente para executar a solução elaborada para à empresa Alfa foi idealizado e deve ser analisado, este ambiente possui ótimas configurações para efetivar a proposta e coletar ótimas métricas, entretanto esse servidor não é o mesmo utilizado pela Alfa. Para eliminar efeitos de latência com a *Internet* a solução deve rodar localmente, porque o objetivo primário é analisar o impacto da captura de dados em tempo real sobre o SQL Server. Então, deve ser construído meios para analisar as métricas coletadas sobre a execução destinada a Alfa.

Figura 16 – Diagrama Relacional do DW



A máquina escolhida deve conter um servidor SQL Server instalado para execução e também um banco de dados para a *Staging Area* e outro para o DW, ambos são banco de dados

SQL Server. Como já citado anteriormente o motivo de utilizar o banco de dados SQL Server é a sua ampla utilização no mercado sergipano e também o SSIS deve ser configurado no o servidor e é dentro dele que os pacotes devem ser executados. Com isso, temos um ambiente disponível para a execução e adicionalmente a Tabela 8 disponibiliza as especificações do servidor de testes.

Tabela 8 – Configuração do Servidor de Teste

Item	**Especificação**
CPU	Intel(R) Xeon(R) CPU E3-1220 v6 @ 3.00GHz.
RAM	16 GigaBytes.
Disco	1 Terabyte de SSD.

4.3.1 Resultados e Discussões

A construção de um método para a avaliação do trabalho se torna necessário, dessa forma foram elaboradas cinco estratégias para avaliação do desempenho da consulta em comparação com a antiga solução apresentada pela Alfa. Cada estratégia representa um procedimento armazenado sendo executado sobre um contexto específico e após a sua execução estatísticas devem ser coletadas. As estatísticas vão ser capturadas utilizando o *Query Store* do próprio SQL Server, ele permite capturar tudo que está sendo efetuado dentro do servidor e disponibiliza em uma ferramenta para uma posterior análise.

Um conjunto de estratégias foram definidas para averiguação da solução, cada uma consiste na execução de um bloco de código SQL como um predicado diferente. O predicado tem como objetivo criar uma consulta sobre diferentes perspectivas simulando a real utilização da empresa Alfa. A Tabela 9 enumera todas as estratégias criadas para efeitos de comparação entre as soluções.

Tabela 9 – Tabela de Estratégias

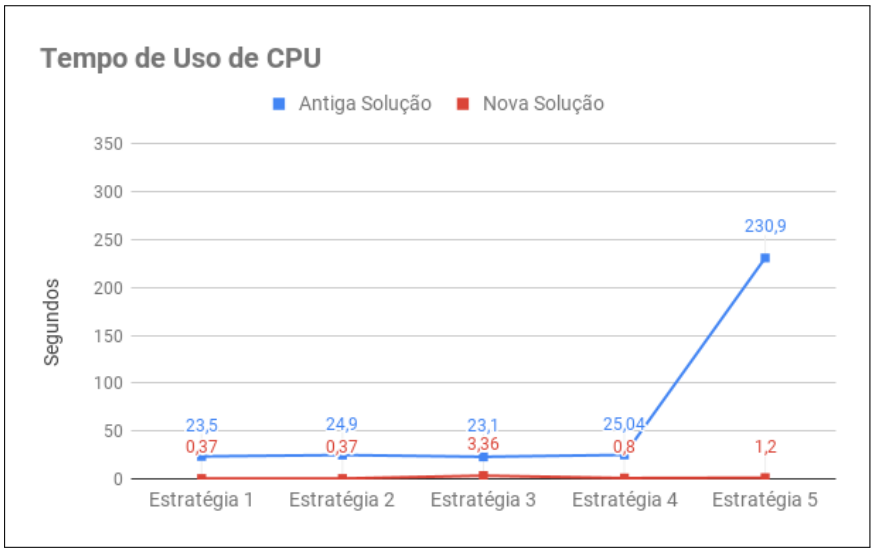
Item	**Predicado**
Estratégia 1	Extrato dos fornecedores não consolidado entre 2015/01/01 e 2015/12/01
Estratégia 2	Somatório dos fornecedores não consolidado entre 2015/01/01 e 2015/12/01
Estratégia 3	Fornecedores com saldo não consolidado ordenados por nome entre 2015/01/01 e 2015/12/01
Estratégia 4	Fornecedores sem saldo não consolidado ordenados por nome entre 2015/01/01 e 2015/12/01
Estratégia 5	Fornecedores com saldo consolidado ordenados por nome entre 2015/01/01 e 2015/12/01

A partir da definição de cada estratégia agora é possível coletar dados sobre o trabalho, cada estratégia foi executada cinco vezes no ambiente de testes e também cinco vezes na antiga solução, após isso a média aritmética foi retirada. Um ponto a ser destacado é a estratégia 5, ela apresenta um comportamento divergente em relação as outras estratégias, esse comportamento se deve ao fato da quantidade de dados processados nela. As seguintes métricas foram coletadas em cada estratégia: Tempo de execução na CPU, consumo de memória, leituras lógicas efetuadas, escritas lógicas anotadas e leituras físicas registradas.

A primeira métrica a ser explanada é o tempo de execução na CPU, o tempo de CPU consiste na quantidade de processamento que o servidor teve que efetuar para realizar uma

determinada operação. A Figura 17 disponibiliza o tempo de processamento de cada consulta em segundos e com essa figura temos um excelente indicativo da efetividade da solução. As quatro primeiras estratégias apresentaram um desempenho relativamente igual, entretanto a quinta estratégia apresenta uma curva muito mais acentuado sendo cerca de 194,41 vezes melhor em comparação com a antiga solução. Essa curva acentuada da última estratégia se deve ao fato da quantidade de dados processados para o resultado final, provando que a solução atual tem um desempenho constante , enquanto a antiga apresenta um comportamento exponencial. Em síntese, todas as estratégias apresentaram um ótimo desempenho com um destaque a última estratégia.

Figura 17 – Tempo de Execução na CPU

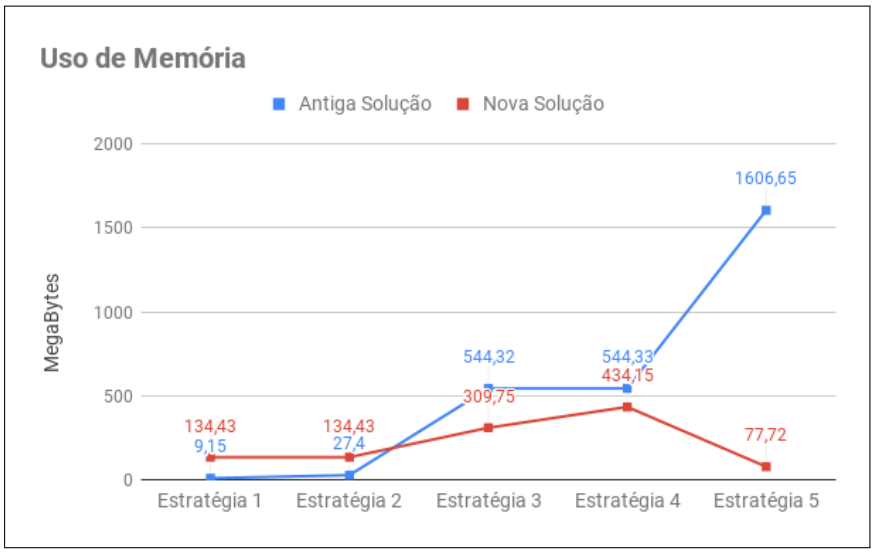


A segunda métrica a ser analisada é o consumo de memória, essa medida indica o total de memória alocada no servidor para a execução da consulta. O total consumido é exibido em *MegaBytes* (MB) para uma melhor visualização, além disso o total de uso de memória é variável de acordo com os operadores adotados para executar o bloco de código em SQL.

A antiga solução apresentou um melhor desempenho nas estratégias 1 e 2, entretanto a medida que os predicados se tornam mais exigentes as estratégias decorrentes demostram um desempenho superior na nova solução, e adicionalmente a estratégia 5 deve ser destacada que mesmo utilizando uma massa de dados exorbitante apresentou um excelente desempenho. Uma consideração importante sobre a estratégia 5 é a maneira de como é efetuada a consolidação na antiga solução, que por sua vez utiliza dobro de linhas de código SQL em comparação com as outras estratégias. Dessa forma, a nova solução é melhor para consultas mais exigentes, mas com a ressalva que isso depende de como o código SQL foi escrito. A Figura 18 ilustra o uso de memória entre as soluções.

A terceira métrica observada é a quantidade de leituras lógicas realizadas durante cada execução, ela informa a quantidade de páginas lidas da área de cache. Nessa medida quanto

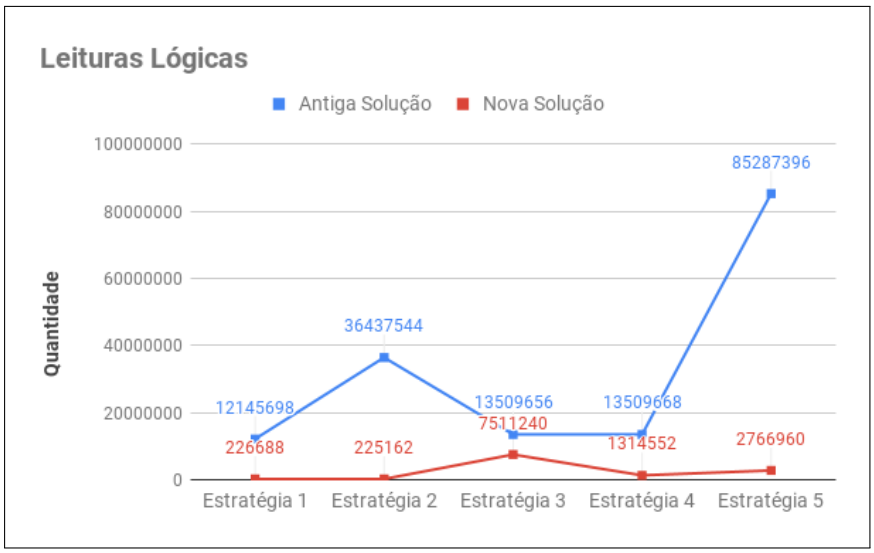
Figura 18 – Consumo de Memória



maior for o valor registrado maior o estresse causado no servidor, também é necessário destacar que uma leitura lógica equivale a ler oito *KiloBytes* (KB) de dados.

Confrontando os dados capturadas de todas as estratégias sobre o ponto de vista da terceira métrica , ela mostra que a nova solução tem um melhor desempenho em todas as abordagens. As quatro primeiras estratégias apresentaram um desempenho semelhante, porém novamente é preciso destacar o desempenho da estratégia 5 na nova solução que demonstrou uma performance 63,35 vezes melhor em relação a antiga solução. Desse modo resultados satisfatórios também foram encontrados analisando essa medida e a Figura 19 apresenta uma comparação entre as soluções.

Figura 19 – Leituras Lógicas

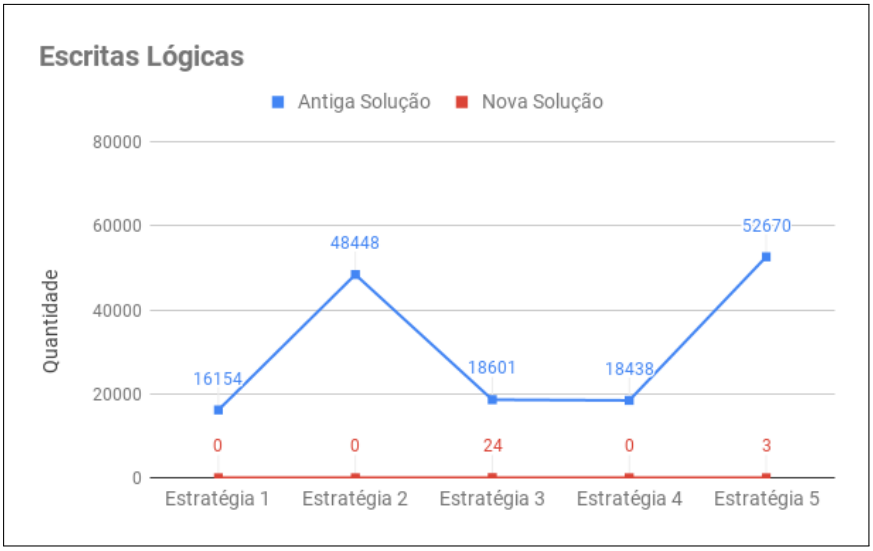


A quarta métrica indica a quantidade de escritas lógicas efetuadas em uma área temporária, de modo análogo as leituras lógicas quanto maior for essa métrica também será o consumo

de recursos do servidor, e também uma escrita lógica processa oito KB de dados na memória.

A nova solução em todas as abordagens demonstraram praticamente zero na utilização de escritas lógicas comprovando assim sua eficiência. O principal motivo por a nova solução não fazer uso dessa métrica é a utilização de operadores simples durante a sua execução. Então, o grau de utilização baixo na nova solução dessa métrica indica outra vez o seu ótimo desempenho em relação à antiga solução. A Figura 20 exibe um gráfico comparando as soluções.

Figura 20 – Escritas Lógicas



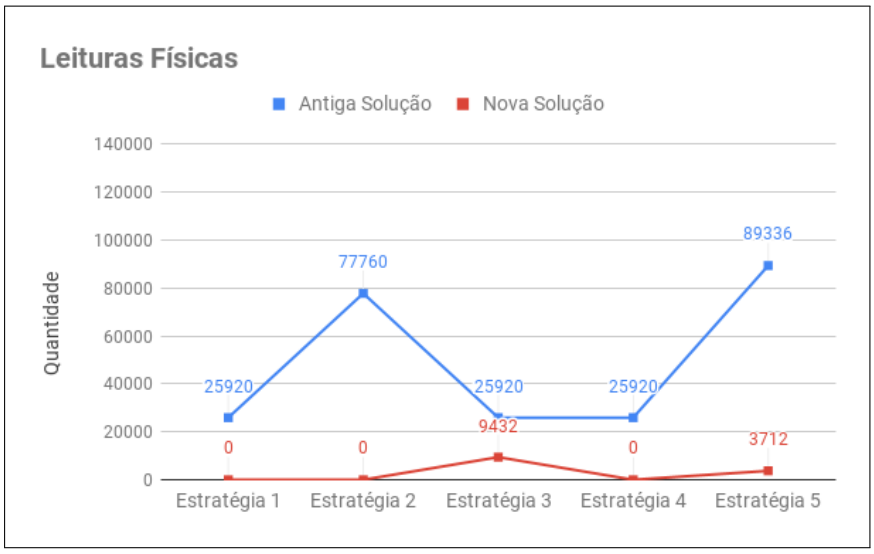
A quinta e última métrica é a quantidade de escritas físicas realizadas em disco, esta por sua vez pode indicar se um enorme consumo de disco é necessário para disponibilizar o resultado, uma escrita física é expressa em valores unitários onde cada unidade equivale a oito KB de dados escritos em disco.

A nova solução apresenta uma performance muito superior em relação à antiga solução, assim é notável na nova abordagem o baixo uso de escrita em disco em cada estratégia. Destaque novamente vai para a estratégia 5 que apresenta uma amplitude elevada em comparação com as outras estratégias, como os dados são processados em uma tabela temporária na antiga solução, isso implica em mais escrita no disco para preencher os dados na temporária. Em suma, a melhoria observando essa métrica foi satisfatória com resultados indicando o baixo consumo de disco. A Figura 21 exibe um gráfico confrontando as soluções.

O resultado de todas as estratégias elaboradas com os seus indicadores foram satisfatórios, cada estratégia na nova abordagem teve praticamente um desempenho superior em todas as métricas analisadas, com exceção somente de algumas. Esses dados expressam o ótimo desempenho em relação a nova solução, mas não informa o impacto de atualizar as atualizações em tempo real. Para resolver essa questão um novo teste deve ser efetuado para capturar o impacto causado pelas atualizações constantes realizadas pelo SSIS.

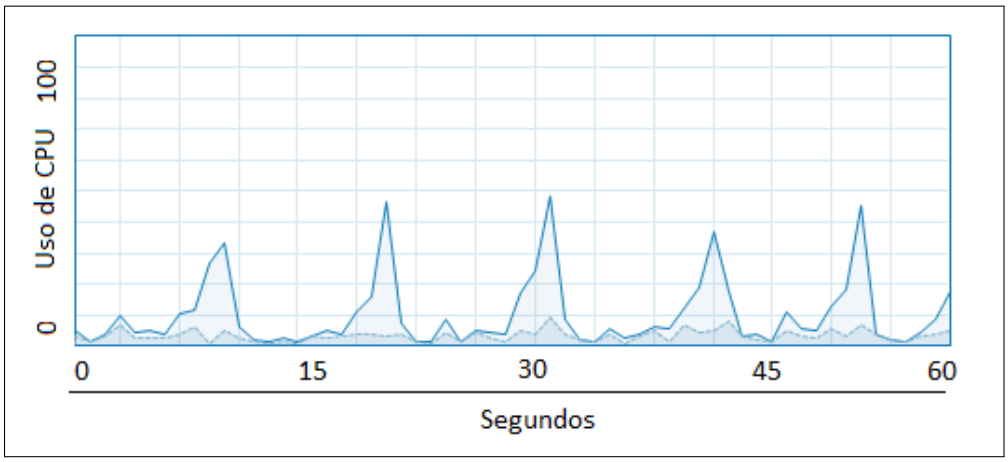
A Figura 22 exibe o impacto causado na utilização de CPU do servidor para atualização

Figura 21 – Leituras Físicas



das informações no DW em tempo real. Um comportamento é observado com um leve pico na taxa de consumo da CPU, que deve ser oriundo das execuções constantes dos pacotes incrementais constantemente, também fica evidente que cada pico de uso ocorre a cada quase doze segundos, mas essa taxa pode variar a depender da quantidade de informações processadas.

Figura 22 – Uso de CPU vs Tempo



A latência para inclusão de novas informações em média está levando cerca de sete segundos, sendo que a medida que as transações forem realizadas no banco de dados OLTP, e em paralelo o processamento delas no DW com uma latência aceitável respectivamente. Outra consideração importante é que quanto menor for à latência maior será o uso de CPU, apresentando assim um comportamento inversamente proporcional. Dessa forma, um DW em tempo real foi arquitetado com sucesso pelo seu baixo nível de latência.

Uma política de *log* foi executada para monitorar a execução de todos os procedimentos armazenados na *Staging Area*. Cada procedimental é responsável por informar se foi executado com sucesso ou não, sendo que todas as informações são registradas na tabela ADM_LOG.

Também é desejável a mesma implantação de *log* para o controle de execução dos pacotes dentro do SSIS, caso ocorra algum erro no pacote é registrado somente no arquivo de *log* no SSIS que não é conveniente.

Uma minimização no uso de CPU para cargas incrementais pode ser efetuada somente com a troca dos tipos das colunas *Varchar (N)* para *NVarchar (N)*. A modelagem do DW para tipos textuais está fixada como *Varchar (N)*, porém o CDC trabalha com uma performance melhor quando a coluna é *NVarchar (N)*, o motivo para o CDC ser melhor com esse tipo é que os dados de colunas de textos são capturados com esse tipo quando são gerados no OLTP, e caso a coluna do DW esteja com a mesma tipagem não seria necessário nenhuma conversão de tipos. Com isso, a eliminação de uma etapa nos pacotes incrementais pode ser realizada resultando em um menor consumo de CPU.

Uma melhoria pode ser efetuado com a utilização um índice colunar na tabela fato do DW. O índice colunar é frequentemente empregado em *workloads* analíticos, provendo uma melhor performance para consultas analíticas. Na configuração atual do DW somente índices que trabalham com linhas foram empregados, dessa forma provendo uma aceleração no processo de busca nos pacotes incrementais do SSIS.

As consultas elaboradas para execução e demonstração da solução não são dinâmicas. Uma consulta dinâmica criada por uma camada intermediária deve ser mais vantajosa que o modelo atual de consultas estáticas, visto que consultas estáticas injetam muito mais coisas no predicado da consulta para atendar o que foi solicitado. Então, a definição dessa camada para preparação de consultas dinâmicas deve prover uma melhor flexibilidade na utilização do DW em tempo real.

5

Considerações Finais

A situação enfrentada pela empresa Alfa é recorrente em diversas empresas sergipanas, visto que pela falta de recursos nessas empresas, elas arquitetam suas soluções em um único ambiente. Como já citado a separação do ambiente analítico e o transacional é de suma importância porque a permanência do OLAP no próprio OLTP baixa o desempenho do servidor, isto gera um incômodo para os usuários do sistema devido a lentidão ao processar as transações cotidianas e em paralelo as consultas analíticas. O propósito deste trabalho está em desenvolver uma solução que forneça consultas analíticas em tempo real por meio de cargas contínuas.

Um DW foi construído e integrado com uma tecnologia de fluxo de dados chamada CDC, esta por sua vez permitiu o processamento das operações provenientes do OLTP e com a capacidade de tempo oferecer as informações em tempo real no OLAP. A utilização do CDC para alimentar um ambiente analítico com baixa latência é uma ótima opção porque causa um baixo impacto em ambos os ambientes, além disso temos um outro fator importante que é o uso de um servidor distinto voltado somente as consultas complexas no DW.

O trabalho realizado gerou ótimos resultados e teve o seu principal objetivo alcançado. A atual solução empregada pela Alfa eleva a taxa de processamento do servidor e também a medida que o volume de dados aumenta mais onerosa se torna a operação, além disso a atual abordagem demonstrou um comportamento exponencial quando mais informações são requeridas. Em contrapartida, a nova abordagem empregada apresentou em praticamente todos os pontos analisados um excelente resultado e muito acima do esperado, isto é reflexo do baixo consumo de recursos do servidor para fazer a análise, pois as informações já foram preparadas e armazenadas para o consumo em um formato estrela.

Para trabalhos futuros incluem a utilização de um índice colunar na tabela fato, através deste tipo de índice as consultas analíticas são aceleradas. Também poderão ser realizadas melhorias na etapa de processamento dos pacotes, visto que há uma amplitude considerável de configurações neles e a busca de uma configuração que miniminize o impacto é interessante.

Outro ponto importante é a melhoria no mecanismo de *log* no SSIS, a forma que está feita pode ser melhorada para prover mais informações detalhadas sobre a execução de cada pacote, permitindo assim uma melhor rastreabilidade em caso de falha. A utilização de cubos para extração de informações pode ser adotada, um cubo contém diversos algoritmos complexos que ajudam no cálculo de consultas analíticas.

Referências

ALI, A. A.; MOHAMED, W. M. Monitoring business transactions for a real-time data warehouses. *International Journal of Computer Applications*, Foundation of Computer Science, v. 146, n. 8, 2016. Citado 3 vezes nas páginas 14, 26 e 28.

BRAGHITTONI, R. *Business intelligence: implementar do jeito certo e a custo zero*. [S.l.]: Casa do Código, 2017. ISBN 9788555192524. Citado 8 vezes nas páginas 13, 16, 20, 21, 24, 26, 34 e 46.

CORONEL, C.; MORRIS, S.; ROB, P. Database systems: design, implementation, and management. *Cengage Learning*, v. 9, 2009. Citado 2 vezes nas páginas 16 e 19.

COSTA, J. K. G. et al. Experimentação na indústria para aumento da efetividade da construção de procedimentos etl em um ambiente de business intelligence. In: *SBSI-Simposio Brasileiro de Sistemas de Informacao*. [S.l.: s.n.], 2015. Citado na página 26.

DONSELAAR, V. *Low latency asynchronous database synchronization and data transformation using the replication log*. Dissertação (Mestrado) — University of Twente, 2015. Citado 2 vezes nas páginas 29 e 35.

ELMASRI, R. *Fundamentals of database systems*. [S.l.]: Pearson Education India, 2008. Citado 4 vezes nas páginas 17, 18, 19 e 33.

FRITCHEY, G. *SQL Server 2017 Query Performance Tuning: Troubleshoot and Optimize Query Performance*. [S.l.]: Apress, 2018. Citado na página 18.

GBOSBAL, S.; KIM, S. K. Building effective intelligence systems for competitive advantage. *Sloan Management Review (1986-1998)*, Massachusetts Institute of Technology, Cambridge, MA, v. 28, n. 1, p. 49, 1986. Citado na página 20.

GROSSMANN, W.; RINDERLE-MA, S. *Fundamentals of business intelligence*. [S.l.]: Springer, 2015. Citado 2 vezes nas páginas 13 e 21.

JÚNIOR, M. C. Projetando sistemas de apoio a decisão baseados em data warehouse. *Rio de Janeiro: Axcel*, 2004. Citado na página 24.

KAKISH, K.; KRAFT, T. A. Etl evolution for real-time data warehousing. In: *Proceedings of the Conference on Information Systems Applied Research ISSN*. [S.l.: s.n.], 2012. v. 2167, p. 1508. Citado 2 vezes nas páginas 22 e 27.

LUIZ, A. *Dados Históricos com Change Data Capture*. 2016. Disponível em: <https://dataslight.blog/indexacao-de-series/dados-historicos-com-change-data-capture/>. Acesso em: 13 fev. 2019. Citado na página 29.

MICROSOFT. *Sobre o change data capture (SQL Server)*. 2017. Disponível em: <https://docs.microsoft.com/pt-br/sql/relational-databases/track-changes/about-change-data-capture-sql-server?view=sql-server-2017>. Acesso em: 13 fev. 2019. Citado 2 vezes nas páginas 14 e 29.

NGUYEN, T. M.; TJOA, A. M. Zero-latency data warehousing (zldwh): the state-of-the-art and experimental implementation approaches. In: *RIVF*. [S.l.: s.n.], 2006. p. 167–176. Citado na página 26.

SHARDA, R.; DELEN, D.; TURBAN, E. *Business intelligence, analytics, and data science: a managerial perspective*. [S.l.]: Pearson, 2016. Citado 4 vezes nas páginas 13, 22, 23 e 25.

SILBERSCHATZ, A.; KORTH, H.; SUNDARSHAN, S. *Sistema de banco de dados*. [S.l.]: Elsevier Brasil, 2012. Citado 2 vezes nas páginas 17 e 19.

TANK, D. M. Reducing etl load times by a new data integration approach for real-time business intelligence. *International Journal of Engineering Innovation and Research*, v. 1, n. 2, p. 1–5, 2012. Citado 3 vezes nas páginas 14, 22 e 27.

TURBAN, E. et al. *Business Intelligence: um enfoque gerencial para a inteligência do negócio*. [S.l.]: Bookman Editora, 2009. Citado na página 20.

VERCELLIS, C. *Business intelligence: data mining and optimization for decision making*. [S.l.]: John Wiley & Sons, 2011. Citado 5 vezes nas páginas 13, 20, 22, 24 e 26.

ZHOU, H.; YANG, D.; XU, Y. An etl strategy for real-time data warehouse. In: *Practical applications of intelligent systems*. [S.l.]: Springer, 2011. p. 329–336. Citado na página 28.