Cristiano Rogério Lopes Xavier

Extração, Transformação e Carregamento Ágil de Dados

Dissertação de Mestrado

Mestrado em Informática, Especialização em Engenharia de Software

2º Ciclo de Estudos

Departamento de Inovação, Ciência e Tecnologia

Dezembro 2012

Dissertação apresentada à Universidade Portucalense para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Software, realizada sob orientação científica do Doutor Fernando Joaquim Lopes Moreira, Professor Associado do Departamento de Inovação, Ciência e Tecnologia da Universidade Portucalense.

DECLARAÇÃO

Nome: _____

Nº. do BI/CC: _____ Tel/Telem.: _____ e-mail: _____

Curso de Doutoramento/Mestrado em _____

Ano de conclusão: _____/___/___

Título da tese / dissertação

_____

Orientador (es):

_____
_____

Declaro, para os devidos efeitos, que concedo, gratuitamente, à Universidade Portucalense Infante
D. Henrique, para além da livre utilização do título e do resumo por mim disponibilizados, autorização, para esta arquivar nos respetivos ficheiros e tornar acessível aos interessados,
nomeadamente através do seu repositório institucional, o trabalho supra-identificado, nas
condições abaixo indicadas:

[Assinalar as opções aplicáveis em 1 e 2]

1. Tipo de Divulgação:

□ Total.

□ Parcial.

2. Âmbito de Divulgação:

□ Mundial (Internet aberta)

□ Intranet da Universidade Portucalense.

□ Internet, apenas a partir de □ 1 ano □ 2 anos □ 3 anos – até lá, apenas Intranet da UPT

Advertência: O direito de autor da obra pertence ao criador intelectual, pelo que a subscrição desta declaração não implica a renúncia de propriedade dos respetivos direitos de autor ou o direito de a usar em trabalhos futuros, os quais são pertença do subscritor desta declaração.

Assinatura: _____

Porto, ____/____/____

Agradecimentos

Quero agradecer ao Professor Doutor Fernando Joaquim Lopes Moreira pela competência com que orientou esta minha dissertação e o tempo que generosamente me dedicou transmitindo-me os melhores e mais úteis conselhos, com paciência e compreensão.

Agradeço à minha esposa e família pela sua compreensão, dedicação, e apoio ao longo destes anos do curso.

Um agradecimento à DevScope e em especial ao Engenheiro Rui Quintino e Engenheiro José António e Silva, com os conselhos e experiencia dada foram peças determinantes para a concretização da dissertação.

Quero também agradecer aos meus colegas de curso e restantes amigos cujo apoio e companheirismo foram determinantes.

Palavras-chave extração, transformação, carregamento, armazém de dados, conhecimento de negócio

Resumo O presente trabalho propõe a criação de uma ferramenta ágil de trabalho para técnicos especializados na área de conhecimento de negócio, que facilita a consolidação de informação num repositório central denominado por armazém de dados. Foram criados mecanismos de criação, controlo e monitorização dos processos de extração transformação e carregamento de dados, que visam dar uma resposta mais rápida quer na criação quer na monitorização dos processos.

Keywords extract, transform, load, data warehouse, business intelligence

Abstract This paper proposes the creation of an agile tool for technicians working in the area of business intelligence, which facilitates consolidation of information in a central repository called a data warehouse. Mechanisms have been established to create, control and monitoring of the processes of extraction transformation and loading of data, which are intended to give a faster response either in creating or monitoring processes.

Índice

Lista de Tabelas

Lista de Figuras

Lista de Abreviações e Acrónimos

API	Application Programming Interface
ASP.NET	Active Server Pages .NET
BI	Business Intelligence
C#	C Sharp
CI	Configuration Item
COM	Component Object Model
CPU	Central Processing Unit
DLL	Dynamic-link library
DW	Data Warehouse
ER	Entity–relationship model
ETL	Extract, Trasnform, Load
HTML	Hyper Text Marked Language
IIS	Internet Information Server
MS-DOS	MicroSoft Disk Operating System
OLAP	On-line Analytical Processing
OLTP	Online Transaction Processing
SP	Stored Procedure
SQL	Structured Query Language
SSAS	SQL Server Analysis Services
SSIS	SQL Server Integration Services
Stag	Staging
VB	Visual Basic
XML	Extensible Markup Language
XMLA	Extensible Markup Language for Analysis

1. Introdução

As organizações para se tornarem mais competitivas têm de responder de forma mais acelerada e ágil às necessidades face à sua concorrência e às constantes oscilações dos mercados. Segundo o estudo publicado pelo *The Data Warehouse Institute,* a 510 empresas, mostra que com a chegada do *Business Intelligence* houve uma economia no tempo em cerca de 61 % das empresas e que cerca de 57% das empresas tiverem melhores decisões estratégicas enquanto que 56% tiveram melhores decisões táticas, além disso foi inferido que 39% das empresas geraram uma economia nos custos. (ECKERSON, 2010)

Um dos processos mais críticos na construção de uma ferramenta de conhecimento de negócio (*Business Intelligence*) é a consolidação da informação dos ambientes operacionais em repositórios analíticos. A criação de um repositório central denominado como *Data Warehouse* ou *Data Mark*, é um processo complexo não só na construção do modelo que permite sustentar as necessidades do negócio, mas principalmente no processo de catalogação que passa por três fases: Extração, Transformação e Carregamento de dados. (Kimball & Caserta, 2004)

O sistema de *ETL (Extract Transform and Load)* é muito mais que uma ferramenta de obtenção de dados de um sistema de origem para um repositório central, ele remove erros e corrige a falta de dados, fornece métricas de confiança nos dados, captura fluxos de dados transacionais para proteção da informação, ajusta dados de várias fontes para serem analisados em conjunto e estrutura os dados para serem utilizados por ferramentas de utilizadores finais. (Kimball & Caserta, 2004)

Numa ótica de agilizar a criação de um ETL para minimizar tempos, recursos e custos da produção de um *Data Warehouse*, a solução proposta visa a construção de uma ferramenta que automatiza e monitoriza processos de extração, transformação e carregamento dos dados. É uma *framework* que apesar de segmentada para um público-alvo especializado não requer grandes conhecimentos técnicos na área de Business Intelligence, pois permite a

criação *Out Of The Box*[1] de processos de extração que colocam a informação numa área temporária, de componentes de transformação de dados e de duas metodologias de carregamento para o *Data Warehouse* detalhadas no capítulo seguinte.

Existem dois modos de operacionalizar a ferramenta, utilizando as funções disponibilizadas em PowerShell[2] ou então através do *Web Site em ASP .NET*[3].

1.1 Motivação

Na criação de um *Data Warehouse* gasta-se cerca de 70% do tempo em construção de mecanismos de ETL (Kimball & Caserta, 2004). A necessidade de minimizar este impacto na construção de repositórios centrais através de uma solução que permitisse responder de uma forma segura sem introdução de tarefas repetitivas, garantindo o foco dos consultores de *Business Intelligence* na criação do modelo analítico, é o desafio principal.

Com a criação de uma ferramenta ETL onde as regras já estão todas definidas e centralizadas, o grau de confiança fica mais acentuado, pois minimiza o erro das tarefas repetitivas, para além da garantia de qualidade dos dados, previsto no *roadmap*[4] da plataforma através de componentes de asserção[5].

Para conseguir abranger mais colaboradores especializados do que o da área de *Business Intelligence,* a disponibilização de funções já criadas com todos os mecanismos de asserção, automação, criação de extratores ou integradores, era importante, no entanto, desejava-se chegar a um nível acima, disponibilizando uma camada de interface que apenas através de um

[1] Característica ou funcionalidade de um produto que funciona imediatamente após a instalação, sem qualquer configuração ou modificação.
[2] É um novo interpretador de linha de comando mais poderoso que o nativo do MS-DOS, explicação detalhada no capítulo seguinte.
[3] Ferramenta de trabalho de aplicações Web desenvolvida e patenteada pela Microsoft que permite a programadores desenvolver sítios web dinâmicos, explicação detalhada no capítulo seguinte.
[4] É um plano que combina com objetivos de curto e longo prazo, com soluções específicas para ajudar a atingir metas propostas (Wikipedia, s.d.).
[5] Asserção (em inglês: assert) é um predicado inserido no programa para verificar uma condição que o programador supõe que seja verdadeira, em determinado contexto.

wizard, fosse possível criar novos processos de extração ou integração de dados.

Outro objetivo é garantir, paralelamente ao mecanismo de criação de ETL, um modelo de persistência de indicadores de análise que fosse ágil de forma a que a necessidade de inserção de novas métricas exigisse apenas a parametrização de uma tabela centralizada e a criação de um novo fluxo de extração, reutilizando todo o modelo analítico (*Online Analytical Processing)*, desta forma garantia-se um aumento da produtividade no desenvolvimento, uma diminuição do tempo de desenvolvimento, um aumento da interoperabilidade do sistema, desenvolver com menos pessoas alocadas, redução de custos com o desenvolvimento e manutenção, produção mais estável e com mais qualidade (Gaea Consulting, 2010).

A necessidade de armazenar novos dados está a crescer exponencialmente (Netcraft, 2012), criar instrumentos de agregação de dados para conseguir manter o maior número de registos possível no *Data Warehouse,* sem despender de custos em mais capacidade de armazenamento, era uma mais-valia pretendida, criando processos de agregação de dados, que visam na diminuição do volume de informação, diminuindo a sua granularidade dos factos através de um modelo de desempenho de indicadores (Parker, 2008).

Por fim, outro foco de interesse que levou ao desenvolvimento de uma ferramenta de ETL, foi o controlo e monitorização centralizada dos vários processos de extração e integração de dados, aliado ao controlo de *log de dados*[6], um sistema de notificações automático que conseguisse absorver a informação de uma forma simples e transparente, de forma a conseguir uma reposta mais rápida e mais assertiva sobre problemas de desempenho ou outros.

[6] Log de dados é uma expressão utilizada para descrever o processo de registro de eventos relevantes num sistema computacional.

1.2 Organização da Dissertação

Este documento está organizado em mais 4 capítulos:

- Capítulo 2, Trabalho relacionado – Descrição de tecnologias, ferramentas relacionadas com a solução proposta, divide-se nas seguintes partes:
    - Microsoft Integration Services – Descrição da tecnologia, utilizada no carregamento de informação nas origens
    - Microsoft Windows Service Applications – Modelo utilizado como orquestrador de todo o processo ETL
    - Microsoft SQL Server – Exposição da persistência dos dados e modelos de integração
    - Windows PowerShell – Breve descrição sobre tecnologia utilizada para automação de novos processos
    - ASP.NET – Apresentação da camada de front-end utilizada na arquitetura.
    - OLAP – Breve descrição do conceito On-line Analytical Processing.
- Capítulo 3, Framework ETL – Descrição do protótipo implementado e as suas componentes:
    - Arquitetura – Descrição da arquitetura utilizada em todo o ETL
    - Modelo Logico – Modelo logico da Framework
    - Modelo Físico – Modelo de Entidade Relação
    - Motor de orquestração – Explicação do funcionamento de orquestração dos vários módulos que compõem o ETL e restantes funcionalidades da Framework
    - Calendarização – Funcionamento da calendarização dos componentes
    - Extração – Descrição do funcionamento da extração dos dados
    - Integração – Detalhe dos processos de integração da informação e dos vários modelos aplicáveis
    - Modelo de indicadores – Explicação do modelo ágil para criação de novos indicadores de análise

    - Agregação de dados – Descrição do processo de agregação de dados históricos
    - Processamento OLAP – Exposição dos processos de processamento OLAP
    - Automação em PowerShell – Automação dos componentes existentes para criação de novos processos ETL
    - Interface Web – Componente de administração e exploração da Framework.
- Capítulo 4, Resultados obtidos – Análise do desempenho dos processos ETL.
- Capítulo 5, Conclusões – Conclusões gerais do trabalho efetuado e indicação de trabalhos futuros.

2. Trabalho Relacionado

Este capítulo descreva as tecnologias usadas para a construção da *framework*, detalhando cada ferramenta utilizada em subseções, para uma melhor compreensão do trabalho realizado.

2.1 Microsoft Integration Services

O *Microsoft Integration Services* é uma plataforma empresarial para extração, integração e transformação de dados (ETL). Esta solução para além do carregamento e transformação de dados para um *Data Warehouse* dá resposta a um grande número de requisitos com um conjunto de funcionalidades como envio de correio eletrónico, descarregar ficheiros, limpeza e mineração de informação, etc. (Knight, Veerman, Dickinson, Herbold, & Hinson, 2008).

Os pacotes de integração podem operar de forma isolada ou em colaboração com outros pacotes de integração, mediante a complexidade exigida pelo negócio.

O *Integration Services* pode extrair ou transformar dados de várias fontes como XML, ficheiros, base dados relacionais e carregar esta informação para um ou mais destinos. (Knight, Veerman, Dickinson, Herbold, & Hinson, 2008)

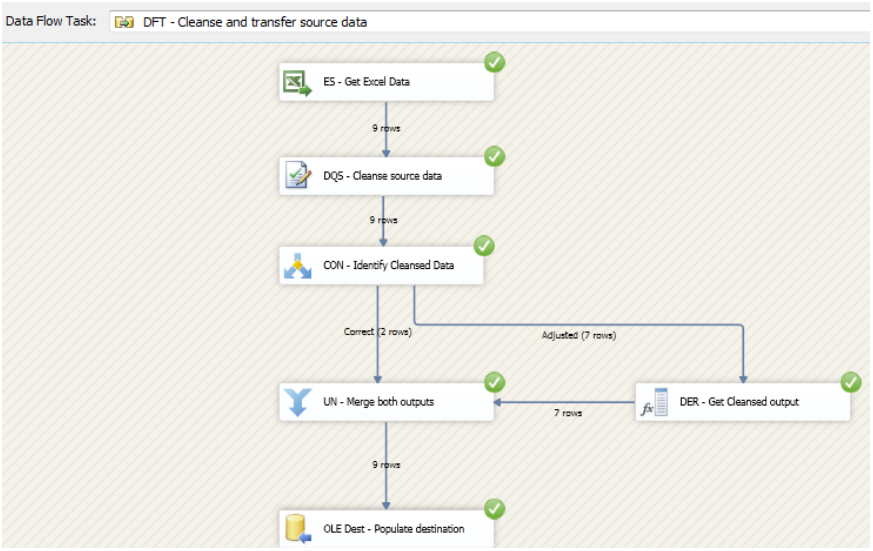


Figura 1 – SQL Server Integration Service

Podem ser usados os componentes gráficos de criação de fluxos de informação (Figura 1), conseguindo sem qualquer tipo de código fazer a informação passar de uma origem para um destino amigavelmente. O exemplo da Figura 1 representa um fluxo de transferência de dados criados desde um ficheiro *Excel*, até a um repositório de dados relacional.

2.2 Microsoft Windows Service Applications

Windows Services Applications, também conhecido por *NT Services*, permite criar aplicações de execução de longa duração em sessões de trabalho do Windows (Morin, 2000). Estes serviços podem ser executados de forma automática quando o computador reinicia e podem também ser colocados em pausa ou serem reiniciados (Microsoft, 2010). São aplicações que não têm interface gráfico, e que correm sem interação do utilizador.

Estas aplicações têm sempre um utilizador de serviço associado, que permite aceder a instâncias de base dados ou a outros ambientes, utilizando as suas credenciais na autenticação, como pode ser observando na coluna *Log On As* da Figura 2.

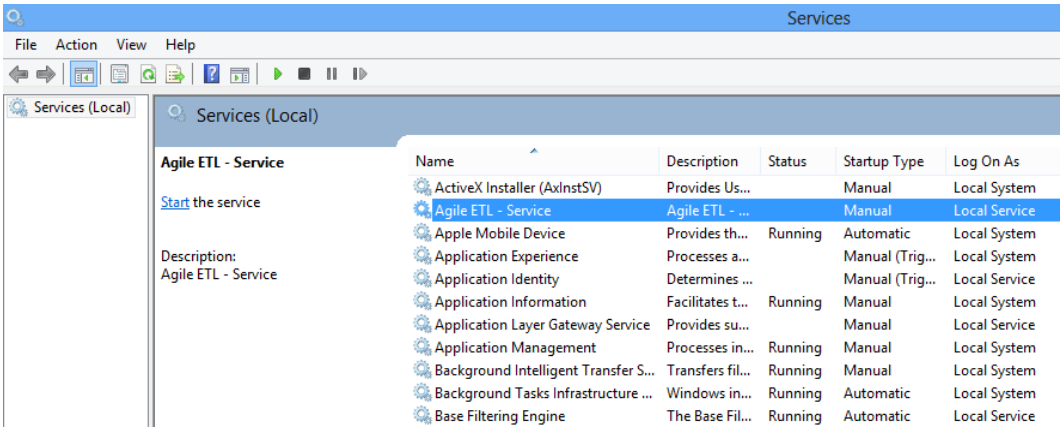


Figura 2 – Serviço Windows

2.3 Microsoft SQL Server

É um sistema de gestão de base de dados, da Microsoft criado em 1988 em parceria com a Sybase e inserido na altura no licenciamento do Windows NT (Mistry & Misner, 2012) . Em 1994 a Microsoft acabou com a parceria continuando a evoluir o produto, atualmente na versão Microsoft SQL Server 2012, com quatro tipos de licenciamento:

- *Enterprise Edition* – contém tudo o que as outras versões disponibilizam mais *SQL Server AlwaysOn*, *xVelociy*, *Data Quality Services*, *Distributed Relay*, *Enhanced Audit*;
- *Business Intelligence Edition* – contém para além do licenciamento do SGBD produtos como *PowerPivot*, *BI Semantic Model*, *Power View*, *Master Data Services*;
- *Standard Edition* – contém *Windows Server Core Support*, *User-Defined Server Roles*, *Distributed Replay*, *Distributed Replay*;
- *Express* Edition – é a versão gratuita que pode alocar até 10GB de informação.

Este sistema corporativo para além de um sistema robusto de armazenamento de informação tem algumas funcionalidades especiais que o distinguem de outros sistemas concorrentes como *Oracle* ou *IBM DB2*, verificado na Tabela 1 (Microsoft, 2012),

Tabela 1 – Comparação plataformas de Base de dados

	Oracle	SQL Server 2012	IBM DB2
Desempenho, escalabilidade e alta disponibilidade	X	X	X
Ferramentas *out-of-the-box* de BI *Self-service*		X	
Implementação flexível na nuvem		X	
Integração nativa com o Microsoft SharePoint e Microsoft Office		X	
Mais seguro (menos vulnerável)		X	
Ferramenta líder da indústria de desenvolvimento		X	
Custo Total de Propriedade		X	
Sistema Operativo Microsoft	X	X	X
Sistema Operativo Mac OS X	X		X
Sistema Operativo Linux / Unix	X		X

Existem outros sistemas sem custos de licenciamento, que podem dar algum contributo em soluções mais pequenas, mas contêm limitações como apresenta a Tabela 2 (Microsoft, 2012).

Tabela 2 – Comparação plataformas de Base de dados sem custo de licenciamento com SQL Server 2012

	MySQL	SQL Server 2012 Express	SQL Server 2012 Enterprise
Custo de Licenciamento			X
Limites de BP, CPU, RAM		X	
Sistema Operativo	Windows/Linux/Unix	Windows	Windows
Operações com várias cores na Base de dados	X	X	X
Vulnerabilidades de segurança	91	25	25
Backup Online		X	X
Relatórios		X	X
Versão Nuvem		X	X
Auto *Tunning* Performance		X	X
Sincronização		X	X
Alta disponibilidade			X
Base dados Analíticas (OLAP)			X
Data Warehouse			X

Devido à oferta dos componentes de *Business Intelligence*, com o licenciamento do SQL Server 2012, e um dos propósitos da ferramenta de ETL, ser comtemplar processos de integração com o modelo analítico SSAS[7], a escolha do SQL Server 2012 foi imprescindível para as aspirações e objetivos traçados para este trabalho.

[7] É uma ferramenta de OLAP (*Online Analytical Processing*) da Microsoft, para exploração de dados.

2.4 Windows PowerShell

É um novo interpretador de linha de comando (Figura 3) mais poderoso que o nativo do MS-DOS, que permite automatização via *script*[8] (Jones, 2011). Usado por muitos administradores como ferramenta de eleição para processos de manutenção e automação (Holmes, 2010). O Windows *PowerShell* permite acesso a todas as API's .NET disponíveis no sistema, além dos Objetos COM e outras API's Microsoft.

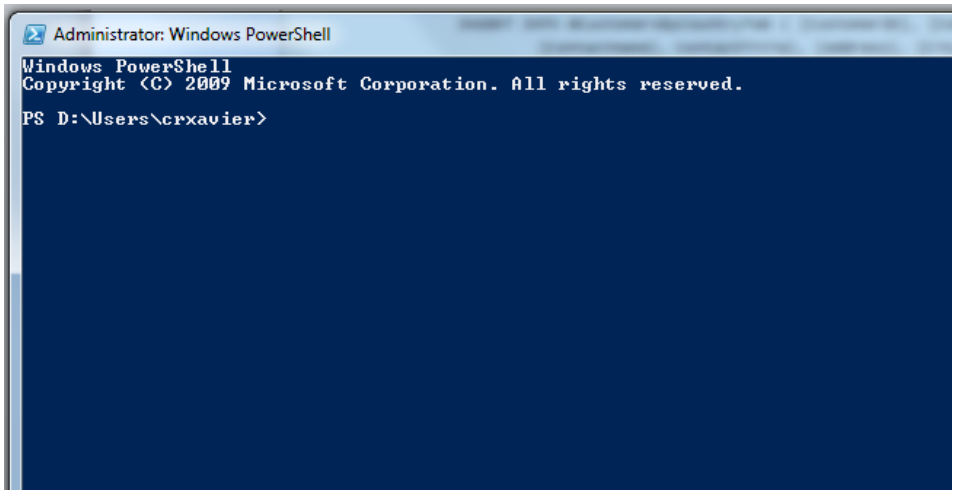


Figura 3 – Windows PowerShell

2.5 ASP.NET

Ferramenta de trabalho de aplicações Web desenvolvida e patenteada pela Microsoft que permite a programadores desenvolver sítios web dinâmicos, aplicações web ou serviços web (MacDonald, Beginning ASP.NET 4.0 in C#, 2010).

Para quem está familiarizado com o Visual Basic ou C#, linguagens de programação Microsoft, não será difícil perceber o código, uma vez que as rotinas de *scripting* de ambos os sistemas são quase idênticas e seguem as mesmas regras. Se além do VB ou C#, existirem conhecimentos de HTML, é possível programar sobre este sistema a um bom nível. (MacDonald, Beginning ASP.NET 4.0 in C#, 2010)

[8] Script, também conhecido como linguagem de scripting, ou linguagem de extensão, são linguagens de programação executadas do interior de programas ou de outras linguagens de programação, não se restringindo a esses ambientes.

Este sistema foi o segundo[9] adotado pela Microsoft para a criação de *sites* dinâmicos com acesso a base de dados. Não existem requisitos em relação aos *browsers* a usar para aceder aos *sites* criados em ASP .Net, uma vez que todo o código é interpretado pelo servidor, e o que chega ao utilizador final é apenas HTML gerado (MacDonald, Freeman, & Szpuszta, Pro ASP.NET 4.0 In C#, 2010).

Existem outras tecnologias que permitem desenvolver aplicações web como apresentado na Tabela 3 (VelvetBlues, 2010).

A escolha do ASP .Net baseou-se no conhecimento técnico já existente relativamente à plataforma, e também pelo facto de com apenas um fornecedor de tecnologias em toda a *framework*, existir uma garantia de melhor interoperabilidade.

Tabela 3 – Comparação de Linguagens Programação

	PHP	**Java**	**ASP .net**
Custo com *Software*			X
Custo com a Plataforma			X
Velocidade	Alta	Baixa	Alta
Eficiência	Alta	Alta	Alta
Segurança	Alta	Alta	Alta
Plataforma	Várias	Apache	IIS
Código aberto	X	X	X
Comunidade	Alta	Baixa	Alta
Exceções		X	X
Programação Orientada Objetos		X	X

2.6 OLAP

On-line Analytical Processing é a capacidade para manipular e analisar um grande volume de dados sob múltiplas perspetivas (Thomsen, Spofford, & Chase, 1999).

As aplicações OLAP são usadas pelos gestores em qualquer nível da organização para lhes permitir análises comparativas que facilitem tomar decisões (Wikipedia, s.d.).

[9] O primeiro sistema foi o de IDC (Internet Database Connector) que usava um sistema de dois ficheiros, um com o código HTML (HTX) que continha rotinas de chamada de ficheiros IDC, que por sua vez tinham as declarações de SQL para acesso às bases de dados.

No modelo OLAP, a informação é concetualmente organizada em cubos que armazenam valores quantitativos ou medidas. As medidas são identificadas por duas ou mais categorias descritivas denominadas dimensões que formam a estrutura de um cubo (Thomsen, Spofford, & Chase, 1999). Uma dimensão pode ser qualquer visão do negócio que faça sentido para a sua análise, como produto, departamento ou tempo. Este modelo de dados multidimensional simplifica a utilização dos utilizadores no processo de formular pesquisas ou "*queries*" complexas, criar relatórios, efetuar análises comparativas, e visualizar subconjuntos (*slice*) de maior interesse. Por exemplo, um cubo que contém informação de vendas poderá ser composto pelas dimensões tempo, região, produto, cliente, cenário (orçamentado ou real) e medidas. Medidas típicas seriam valor de venda, unidades vendidas, custos, margem, etc.

Dentro de cada dimensão de um modelo OLAP, os dados podem ser organizados numa hierarquia que define diferentes níveis de detalhe. Por exemplo, dentro da dimensão tempo, poderá existir uma hierarquia que representa os níveis ano, mês e dia. Da mesma forma, a dimensão região poderá ter níveis como país, região, distrito e cidade. Assim, um utilizador pode visualizar dados num modelo OLAP navegando para cima (*drill up*) ou para baixo (*drill down*) entre níveis para visualizar informação com maior ou menor nível de detalhe sem a menor dificuldade (Webb, Ferrari, & Russo, 2009).

3. *Framework* ETL

Neste capítulo é apresentada a ferramenta criada para efetuar a extração, transformação e integração de dados no *Data Warehouse* através de um processo de orquestração Windows que inclui componentes para um modelo de indicadores, processos de agregação de dados, calendarização de processos ETL e processamento OLAP.

3.1 Arquitetura

Através de funções de automação em *scripts powershell* a *framework* consegue dar resposta na criação ágil de componentes que compõem todo o fluxo, desde a criação de extratores que funcionam como elo de ligação com as origens de dados até à integração dos dados já transformados em concordância com o modelo de negócio.

A arquitetura, como pode ser verificado na Figura 4, tem por base os componentes de extração que podem ser *Store Procedures*[10] ou pacotes *de SSIS,* que colocam a informação de vários repositórios em tabelas, numa área temporária, denominada como *"Staging"*. Estes dados são armazenados na sua forma original, não existindo qualquer alteração das regras do negócio em cada registo, de forma a não ser colocado em causa a operacionalidade dos repositórios de origem.

Após os dados estarem armazenados no repositório temporário, é o momento para trabalhar a informação de forma a consolida-la no *Data Warehouse* em conformidade com as regras do negócio.

Por último, o processamento dos dados é realizado num repositório multidimensional (OLAP), para fornecer um método de acesso, visualização, e análise de dados com elevada flexibilidade e desempenho.

[10] Coleção de comandos SQL utilizados num Sistema de Gestão de Base Dados. Encapsula tarefas repetitivas, aceita parâmetros de entrada e retorna um valor de estados (que indica sucesso ou falha na execução)

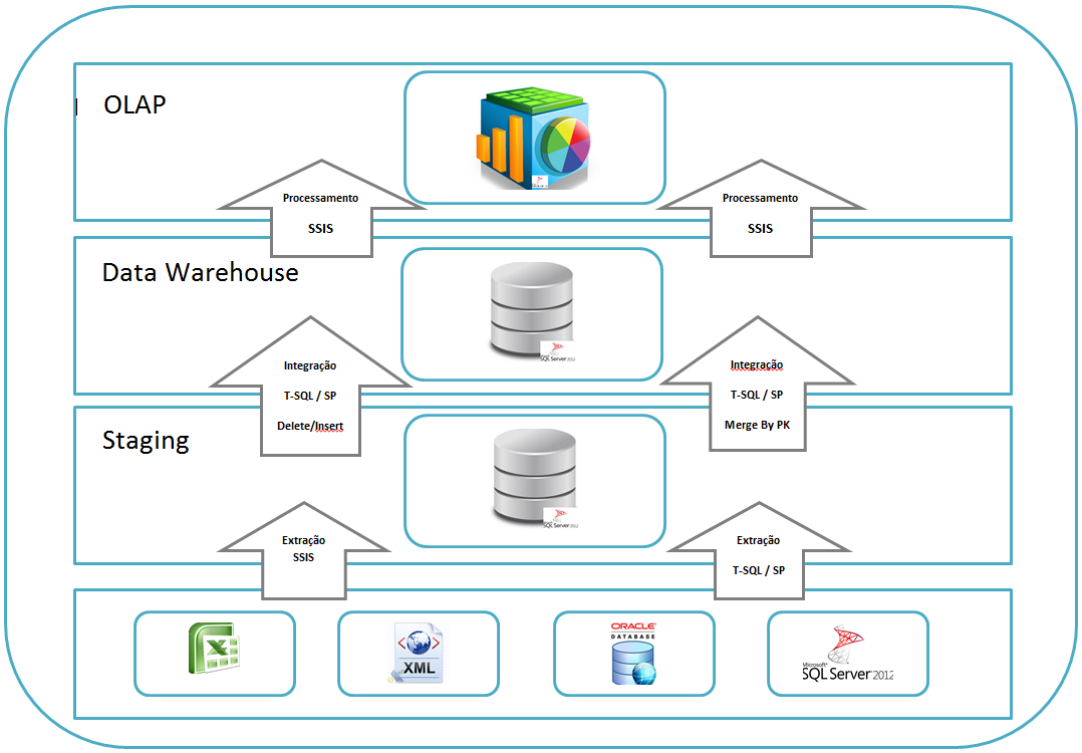


Figura 4 – Arquitetura da framework.

3.2 Modelo Lógico

Nesta seção será apresentada a relação entre as entidades no modelo ETL, para descrever os fluxos e a sua hierarquia de informação.

3.2.1 ETL

O modelo lógico ETL é visível na Figura 5, e representa o processo de orquestração, para a extração, transformação e carregamento de dados, apresentando as suas relações, hierarquias e dependências.

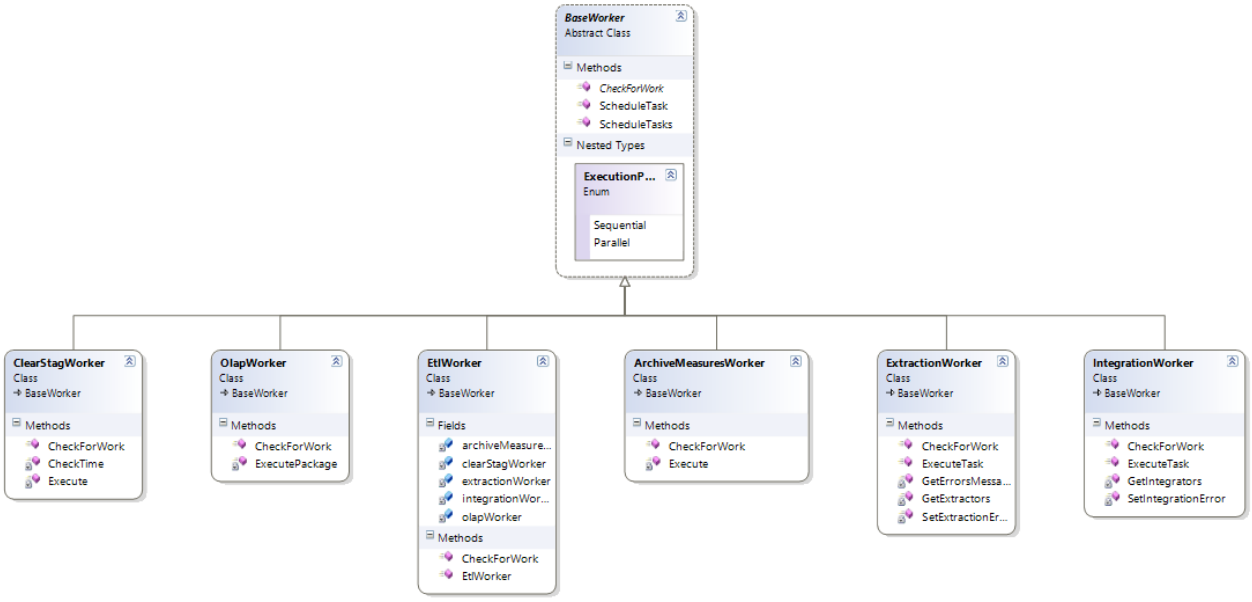


Figura 5 – Modelo Lógico ETL

A relação de dependência existente nas classes *ClearStagWorker*, *OlapWorker*, *ETLWorker*, *ArchiveMeasuresWorker*, *ExtractionWorker* e *IntegratorWorker* com o seu "*Pai*" *BaseWorker*, permite encapsular código e reutilizar em cada um dos nós as regras gerais, através de uma linguagem orientada a objetos (Gaea Consulting, 2010).

A *ClearStagWorker*, é responsável pela limpeza dos dados na área temporária, denominada também por área de *Staging*.

OlapWorker, executa os componentes de processamento OLAP, subscritos na tabela *OlapChangeLog* (ver Tabela 16), através do pacote SSIS da Figura 22.

ETLWorker, orquestra todos os outros componentes a serem executados sequencialmente ou em paralelo dependendo da parametrização escolhida.

ArchiveMeasuresWorker, responsável pela execução da agregação de dados do modelo de indicadores (ver seção 3.3.2).

ExtractionWorker, executa as extrações existentes na tabela de extratores (ver Tabela 17).

IntegrationWorker, executa as extrações existentes na tabela de extratores (ver Tabela 24).

3.3 Modelo Físico

Nesta seção será apresentado o modelo físico que representa a relação física entre as tabelas existente no modelo, bem como as suas chaves primárias e estrangeiras.

3.3.1 ETL

O modelo físico apresentado na Figura 6 é representativo da forma como os dados estão armazenados no *Data Warehouse* para responder às necessidades da *framework* de ETL.

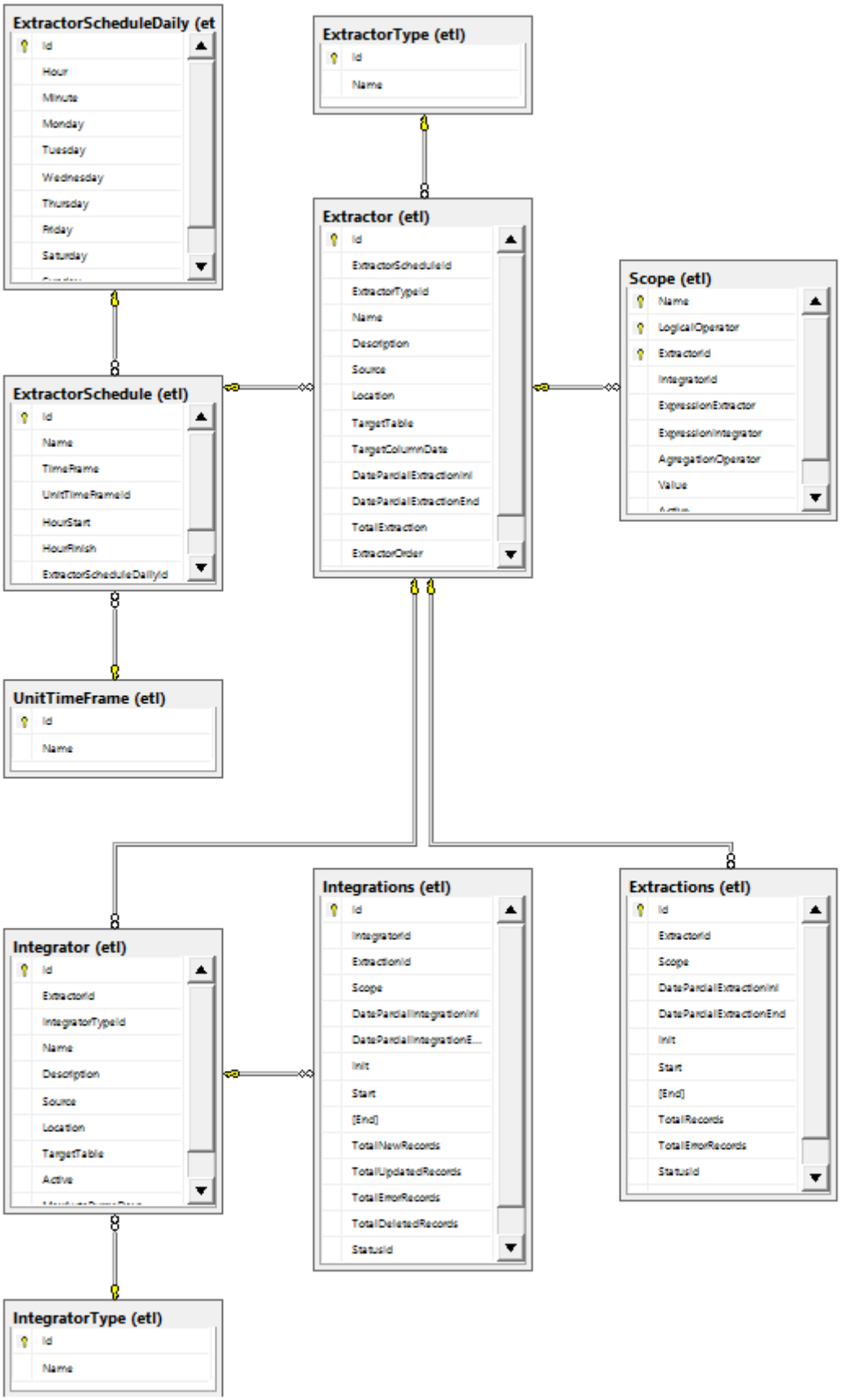


Figura 6 – Diagrama Entidade Relação da framework ETL

É visível na Figura 6, as duas principais entidades, pelas suas relações centralizadas. A tabela *Extractor* (ver Tabela 17) que representa os extratores e a tabela *Integrator* (ver Tabela 24) que representa os integradores, tem relação com as extrações e integrações (*Extractions em detalhe na* Tabela 21 e *Integrations ver* Tabela 26), para além da relação que representa os eventos do motor de ETL, existem outras tabelas normalizadas, como *UnitimeFrame* (ver Tabela 9), *ExtractorSchedule* em detalhe na Tabela 5 e *Scope* (ver Tabela 22).

3.3.2 Modelo de Indicadores

A Figura 7 representa o modelo de indicadores, responsáveis por garantir uma resposta flexível na criação de novas métricas de análise no OLAP. Este modelo físico representa as tabelas e as suas relações no *Data Warehouse*.

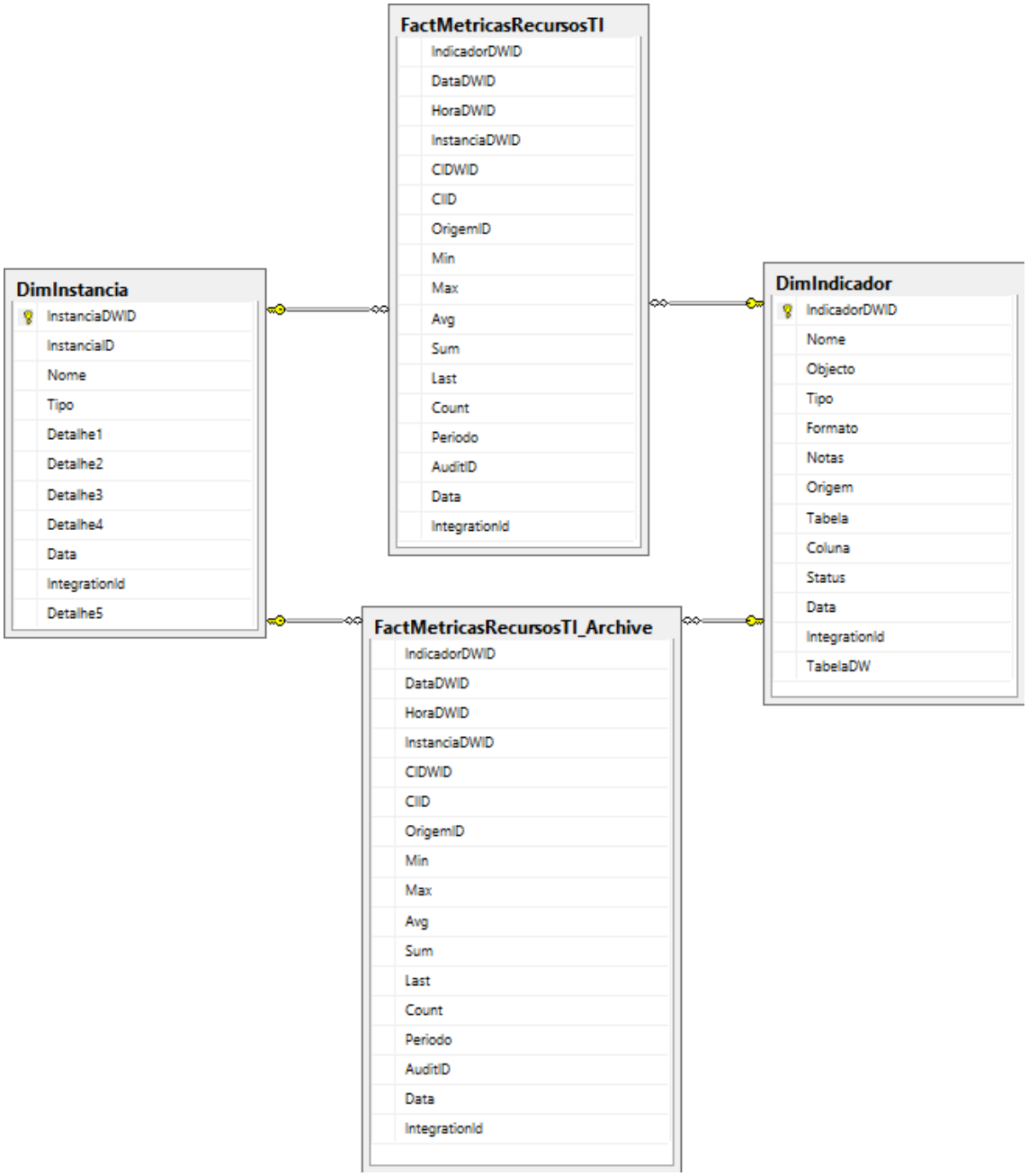


Figura 7 – Diagrama Entidade Relação do Modelo de Indicadores

A relação entre as dimensões *DimInstancia e DimIndicador* (ver Tabela 29 e Tabela 28, respetivamente), serve para garantir que os registos sejam dimensionados através de métricas em *FactMetricasRecursosTI* (ver Tabela 27), criando assim um modelo ágil de implementação de novos indicadores de análise. A tabela *FactMetricasRecursosTI*_Archive é igual à tabela *FactMetricasRecursosTI*, apenas foi criada por questões de desempenho e

manutenção dos indicadores agregados ao dia ou à hora, que são suportados pela *framework*.

3.3.3 Registos de Eventos

O modelo físico apresentado na Figura 8 é representativo de toda a informação guardada pelos mecanismos de registos de eventos da plataforma.

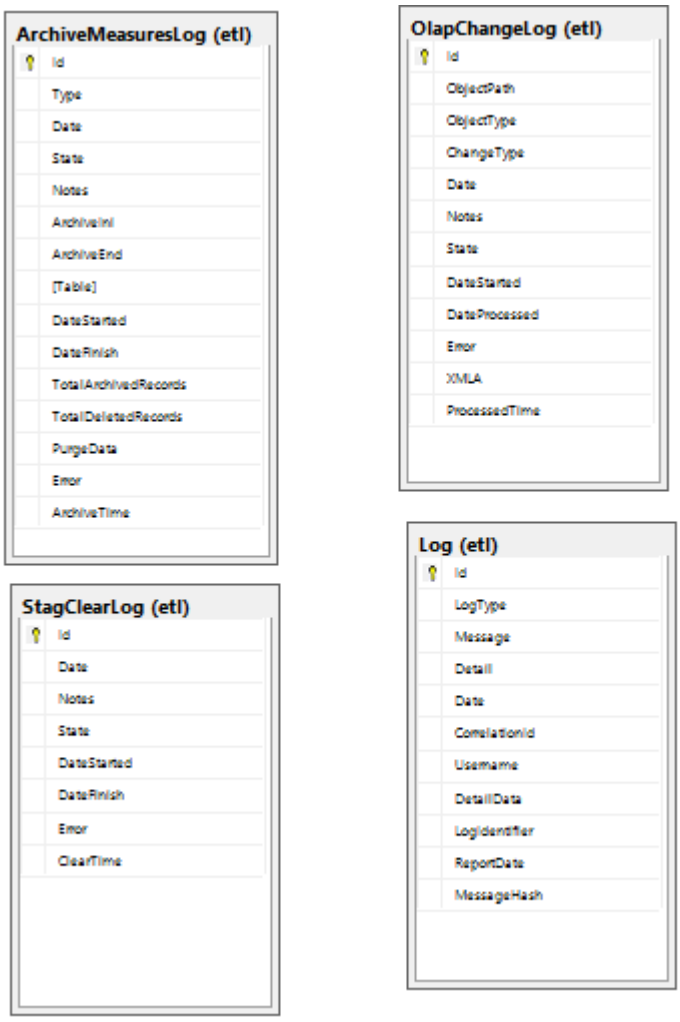


Figura 8 – Diagrama Entidade Relação dos Logs

A *ArchiveMeasuresLog* (ver Tabela 32) é a tabela onde ficam armazenados os eventos de agregação de indicadores descrito na secção 3.5.4. A tabela *StagClearLog* é a tabela onde ficam armazenados os eventos de limpeza de dados da área de *Staging* (ver Tabela 31). A tabela

OlapChangeLog é o repositório de eventos de *Log* dos processos OLAP (ver Tabela 16). Por fim a tabela *Log* é a tabela de eventos genéricos da *framework* (ver Tabela 30).

3.4 Motor de Orquestração

A orquestração de todo o processo é responsável por uma aplicação do tipo Microsoft *Windows Service Applications* na *framework 4.0* designada por AgileETL.*Service*, como pode ser observado na parte que está realçada na Figura 9.

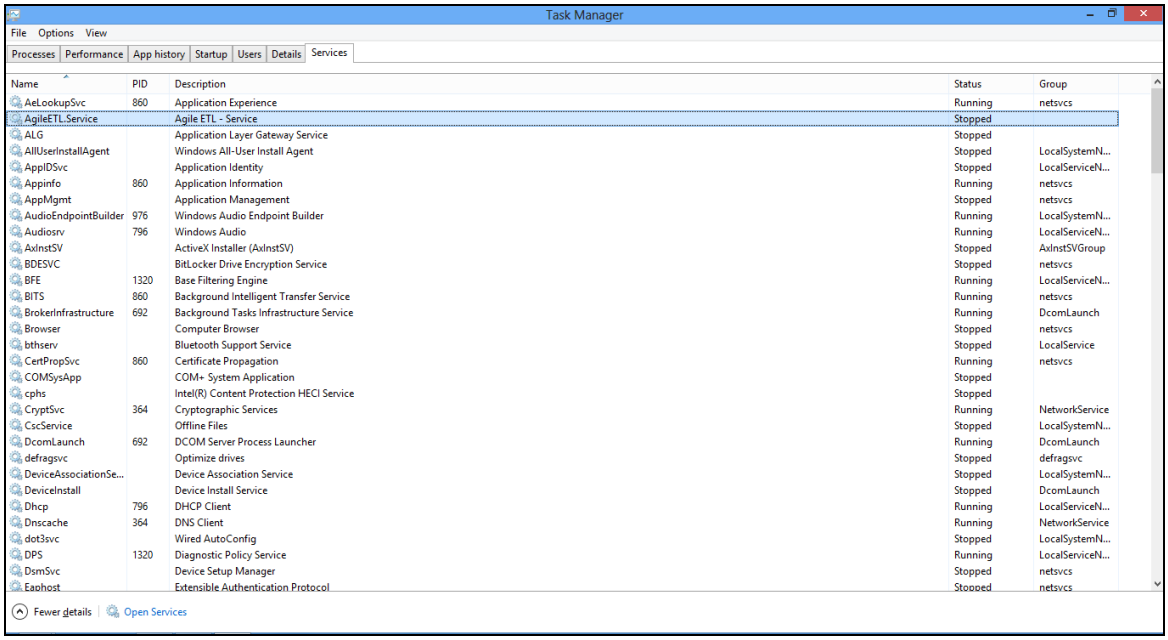


Figura 9 – Serviço Windows AgileETL.Service

No serviço Windows está centralizado todo o núcleo do motor ETL e toda a sua especificidade de controlo, bem como as suas regras.

A vantagem de um motor de orquestração é poder colocar tudo o que são regras de uma forma centralizada sem redundância nem desperdício (Rotem-Gal-Oz, 2012). Exemplo da centralização de regras é a possibilidade de ter vários tipos de componentes de extração de dados, que apenas têm como tarefa copiar dados das origens para o repositório de "*Staging*", sem existir a preocupação com as regras do motor de ETL. A Figura 10, marcado a

amarelo, é a marcação do início de uma extração, útil para controlo de tempos, não está nem no pacote de extração nem numa *store procedure* de extração, o que permite efetuar alterações ágeis e serem replicadas de imediato para todos os processos.

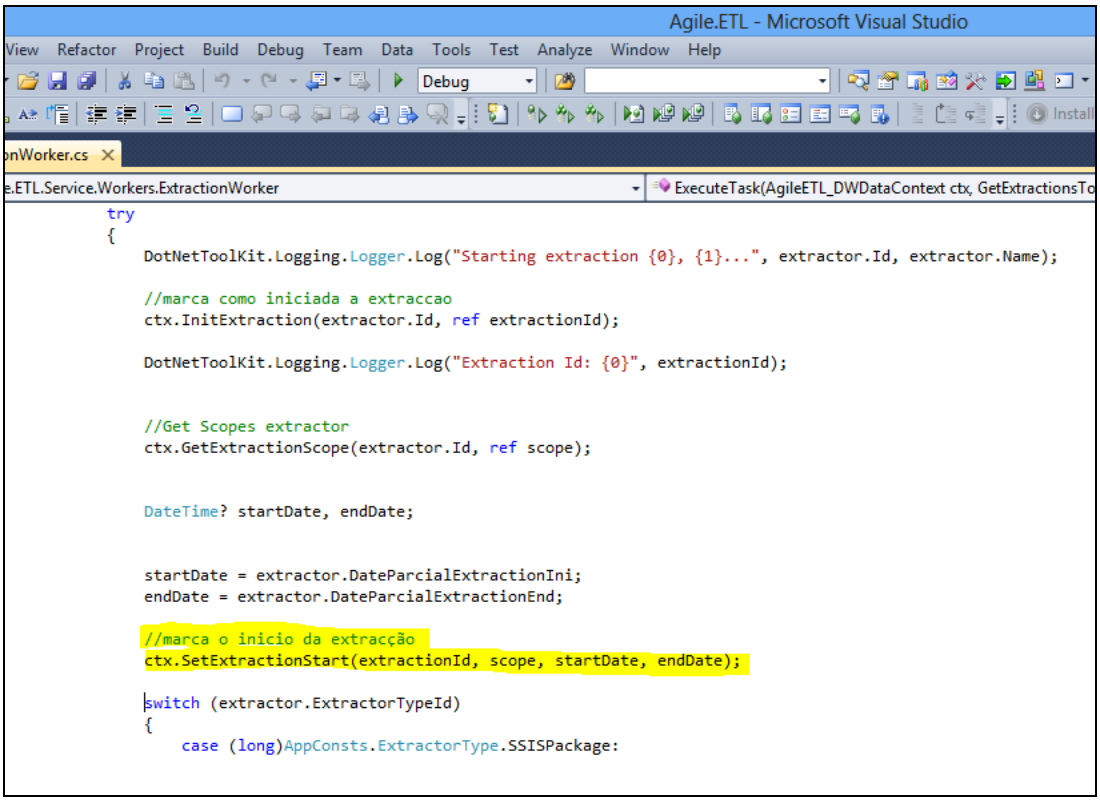


Figura 10 – Centralização de regras no orquestrador

O motor tem vários componentes intitulados como *Workers,* onde os mais importantes são:

- *ArchiveMeasusresWorker* – Agrega e apaga registos do modelo de indicadores, para minimizar a taxa de ocupação do DW;
- *ExtractionWorker* – Responsável pelo arranque, sinalização e *log* dos processos que efetuam a passagem da informação de um repositório na origem até ao repositório temporário (*Staging*);
- *IntegrationWorker* – Inicia, sinaliza e efetua os *log* dos processos que fazem a passagem da informação do repositório temporário (*Staging*) para o *Data Warehouse*;
- *ClearStagWorker* – Apaga todas as tabelas configuradas do repositório temporário numa periodicidade parametrizável;

- *OLAPWorker* – Efetua o carregamento dos pacotes necessários para processar no modelo multidimensional.

Cada módulo tem uma calendarização específica, e é responsável pela sua execução, o motor apenas pergunta a cada iteração a cada componente se tem tarefas para executar.

O serviço pode executar em paralelo ou sequencialmente (ver Figura 11), dependendo do objetivo do administrador de sistema e do próprio negócio. Se os dados forem completamente independentes de um processo para o outro a sua execução em paralelo pode ser uma mais-valia, mas se existirem precedências de informação como carregamento de dimensões a forma sequencial será a mais adequada. É ainda necessário ter em conta o tipo de servidor que se atribui ao projeto, pois o modo paralelo, sendo mais rápido, a execução global pode sobrecarregar mais os recursos e causar alguma indisponibilidade do sistema se os volumes de dados forem bastante elevados.

```
public virtual void ScheduleTasks(IEnumerable<Action> tasks, ExecutionPolicyEnum executionPolicy)
{
    switch (executionPolicy)
    {
        case ExecutionPolicyEnum.Parallel:
            //Parallel mode
            ParallelOptions p = new ParallelOptions();

            Parallel.ForEach(tasks, iAction =>
            {
                try
                {
                    iAction();
                }
                catch (Exception ex)
                {
                    //Parallel foreach stops executing tasks on error by default
                    //Ignore  error, assume that tasks are not dependent on each other

                    Logger.Error(ex);

                }
            });

            break;

        case ExecutionPolicyEnum.Sequential:
            //Sequential Mode
            {
                foreach (var iAction in tasks)
                {
                    try
                    {
                        iAction();
```

Figura 11 – Tipos de execução do motor

3.5 Calendarização

Cada módulo tem a sua calendarização específica. É da sua responsabilidade a execução mediante as suas configurações, que podem estar numa tabela no *Data Warehouse* ou em ficheiros de configuração.

3.5.1 Orquestrador

O motor de orquestração tem um relógio associado que executa num intervalo de tempo e que define a regularidade de cada iteração. A configuração é parametrizada no ficheiro de configurações do serviço *App.config* em milissegundos no atributo *Service.PollTimerInterval*.

3.5.2 Extratores

A calendarização dos extratores, pode ser realizada de duas formas distintas, extração recorrente, será executada mais do que uma vez por dia, ou extração diária que corre apenas uma vez ao dia, ambas configuradas são apresentadas na tabela *ExtractorSchedule* (ver Tabela 4).

Tabela 4 – Esquema ExtractorSchedule

Coluna	Tipo	Descrição
Id	bigint	Identificador único e incremental
Name	Nvarchar(100)	Nome associado à calendarização
TimeFrame	Int	Valor do tempo de execução
UnitTimeFrameId	Int	Id da unidade temporal para associar o valor, exemplo o Id 1 ser segundos e juntamente com o valor na coluna TimeFrame de 30, a execução será feira de 30 em 30 segundos.
HourStart	Int	Inicio de hora de execução
HourFinish	Int	Só executa até a hora assinalada
ExtractorScheduleDaily	Int	Id que identifica qual é a calendarização diária

Para criar uma calendarização diária é necessário criar um registo na tabela *ExtractorSchedule* fazendo a relação com a tabela *ExtractorScheduleDaily* (ver Tabela 6) que terá a hora e o dia em que deverá ser executado. O registo com *id 3* na Tabela 5 é exemplo da marcação diária de um processo de extração, não é necessário nesta parametrização assinalar

qualquer campo à exceção da chave estrangeira *ExtractorScheduleDailyId,* que faz ligação com a tabela *ExtractorScheduleDaily*

Tabela 5 – Exemplo de dados ExtractorSchedule

Id	Name	TimeFrame	UnitTimeFrame	HourStart	HourFinish	ExtractorScheduleDailyId
1	30 em 30 segundos	30	1	NULL	NULL	NULL
2	Hora a Hora	60	2	9	19	NULL
3	Diário Manhã	NULL	NULL	NULL	NULL	2

Tabela 6 – Esquema ExtractorScheduleDaily

Coluna	Tipo	Descrição
Id	Int	Identificador único e incremental
Hour	Int	Hora de execução
Minute	Int	Minuto de execução
Monday	Bit	Se é para executar à segunda-feira
Tuesday	Bit	Se é para executar à terça-feira
Wednesday	Bit	Se é para executar à quarta-feira
Thruday	Bit	Se é para executar à quinta-feira
Friday	Bit	Se é para executar à sexta-feira
Saturday	Bit	Se é para executar à sábado
Sunday	Bit	Se é para executar à domingo

Tabela 7 – Exemplo de dados ExtractorScheduleDaily

Id	Hour	Minute	Monday	Tuesday	Wednesday	Thruday	Friday	Saturday	Sunday
1	8	30	True	True	True	True	True	True	True
2	10	00	False	False	False	False	False	True	True
3	23	0	True	True	True	True	True	False	False

Na criação de um registo de calendarização na tabela *ExtractorSchedule* é necessário fazer corresponder um identificador para a tabela *UnitTimeFrame* (ver Tabela 8), que contém as unidades representadas na Tabela 9.

Tabela 8 – Esquema UnitTimeFrame

Coluna	Tipo	Descrição
Id	Int	Identificador único e incremental
Name	Nvarchar(100)	Nome da unidade de temporal

Tabela 9 – UnitTimeFrame

Id	Name
1	Segundos
2	Minutos
3	Horas
4	Dias

3.5.3 Integradores

Os processos de integração não são calendarizados individualmente, a premissa é quando existir dados extraídos que não foram integrados é iniciado o processo de integração. Este processo pode não ser executado quando existirem mais do que cinco integrações com erros no mesmo dia, o valor cinco pode ser parametrizado, foi escolhido por omissão no procedimento *SQL GetIntegrationsToRun()*.

A integração também pode ser interrompida com a extração ou qualquer outro componente que possa mexer com grandes volumes de dados, quando o sistema não está preparado para executar devidamente. Um exemplo disso é se o *Data Warehouse* estiver a efetuar um *backup* ou um restauro de uma base de dados, os processos de ETL entram em *stand-by* porque uma vez que o *transaction-log*[11] não é libertado quando está a correr este tipo de tarefas, poder-se-ia incorrer numa paragem brusca por falta de espaço em disco, instruções para inserir ou apagar registos ocupam muito espaço em grandes volumes de informação no repositório temporário.

3.5.4 Agregação dos Indicadores

A agregação dos indicadores é controlada através das tabelas *ArchiveMeasuresTable* (ver Tabela 12, para exemplo ver Tabela 13), *ArchiveMeasuresSchedule* (ver Tabela 10 e Tabela 11) e uma tabela intermédia que faz a ligação entre as duas, *ArchiveMeasuresScheduleTable* (ver Tabela 14 e Tabela 15).

[11] Repositório temporário que armazena histórico de ações executadas pelo sistema de gestão de base dados, para garantir as propriedades de atomicidade, consistência, isolamento e durabilidade sobre acidentes ou falhas de hardware (Wikipedia, s.d.).

Tabela 10 – Esquema ArchiveMeasuresSchedule

Coluna	Tipo	Descrição
Id	Int	Identificador único e incremental
Name	Nvarchar(100)	Nome do Agregador
AggregationType	Varchar(1)	Tipo de agregação, D – Dia H – Hora
MonthStart	Int	Valor pelo qual será subtraído em meses à data de hoje que corresponderá ao período de início da agregação
MonthFinish	Int	Valor pelo qual será subtraído em meses à data de hoje que corresponderá ao período de fim da agregação
PurgeArchiveRows	Bit	Campo de controlo que valida se informação agregada é para apagar, caso não exista valor em AggregationType apaga os registos compreendidos entre MonthStart e MonthFinish

Existem apenas dois tipos de agregação dos indicadores como apresentado na Tabela 11, coluna *AggregaionType*, que independentemente do nível de granularidade dos registos colocam a informação agregada à hora ou ao dia.

Tabela 11 – Exemplo de dados ArchiveMeasuresSchedule

Id	Name	AggregationType	MonthStart	MonthFinish	PurgeArchiveRows
1	Agregação Trimestral	H	3	6	True
2	Agregação Semestral	D	6	12	True
3	Dados Históricos		12	24	True

A coluna *PurgeArchiveRows* (ver Tabela 10 e Tabela 11), valida se os registos que foram previamente agregados devem ser apagados do sistema ou não, esta opção de apagar registos, quando não mencionado o tipo de agregação (coluna *AggregaionType* vazia, ver Tabela 11) funciona apenas como serviço de limpeza de dados, ou seja não agrega a informação por nenhum tipo, simplesmente os apaga, é muito recorrente para dados históricos.

Tabela 12 – Esquema ArchiveMeasuresTable

Coluna	Tipo	Descrição
Id	Int	Identificador único e incremental
TableName	Varchar(255)	Nome da tabela a agregar
IsTable	Bit	Campo de controlo que valida se o objeto descrito em TableName é uma tabela ou view. Bastante útil para flexibilizar o modelo porque podemos ter assim tabelas que não respeitem o modelo e mesmo assim serem executadas as funcionalidades pois as views de suporte o esquema necessário para dar resposta ao modelo de indicadores

Tabela 13 – Exemplo de dados ArchiveMeasuresTable

Id	TableName	IsTable
1	FactMetricasRecursosTI	True
2	FactMetricasRecursosTI_2	True
3	vwFactMetricasNetworking	False

Tabela 14 – Esquema ArchiveMeasuresScheduleTable

Coluna	Tipo	Descrição
IdSchedule	Int	Chave estrangeira para tabela de ArchiveMeasuresSchedule
IdTable	Int	Chave estrangeira para tabela de ArchiveMeasuresTable

Tabela 15 – Exemplo de dados ArchiveMeasuresScheduleTable

IdSchedule	IdTable
1	1
1	2
1	3
2	2
2	3

3.5.5 Limpeza da área temporária

A limpeza da área temporária ou área de *Staging* é agendada no ficheiro de configuração do serviço de orquestração.

A periodicidade pode ser agendada de duas formas, ou se define o número de dias a executar à data da última execução, ou seja se for colocado o valor “1” em *Stag.StagClearIntervalDays* será executada a limpeza todos os dias, enquanto que se for colocado o valor na propriedade a *Stag.StagClearDay*, por exemplo “*Sunday*”, será executada a tarefa todos os domingos.

Complementarmente a estas propriedades existe uma outra obrigatória para o funcionamento do módulo, a *Stag.StagClearHour* que define a que horas serão executados os processos, por exemplo se for atribuído o valor “15:00”, ter-se-ia uma limpeza da área temporária todos os Domingos às quinze horas.

3.5.6 Processamento OLAP

A *framework* ordena a execução de tarefas agendadas para processamento OLAP que estão armazenadas na tabela *OlapChangeLog* (ver Tabela 16) com estado igual a “0”.

A responsabilidade de agendamento do processamento é dos integradores que devem inserir registos na *OlapChangeLog* (ver Tabela 16) para serem processados. Este agendamento passa por criar um novo registo com informação da instrução, em código XMLA, que efetuará o processamento do objeto no OLAP.

A *framework* já disponibiliza procedimentos em SQL que registam e criam as entradas na tabela *OlapChangeLog*:

- *etl.OlapRegisterChange* – Serve para marcar e processar dimensões, tem como parâmetros o nome do objeto e o tipo. Exemplo: etl.*OlapRegisterChange* ‘*Dimension Name’* , ‘*DIM’*.
- *etl.OlapGeneratePartionsUpdate* – Serve para marcar partições ou *measure groups* para processar no OLAP. Tem como parâmetros o nome da partição ou do *measure group*, o *id* da integração e um parâmetro de controlo do tipo *booleano* para verificar se o particionamento é diário.
  Exemplo: etl.*OlapGeneratePartionsUpdate* ‘*MeasureGroup’*, *integrationid*, *1.*

Ao executar um destes comandos serão criadas entradas na tabela *OlapChangeLog* (ver Tabela 16), com o XMLA necessário para o processamento dos objetos OLAP.

Tabela 16 – Esquema OlapChangeLog

Coluna	Tipo	Descrição
Id	Int	Identificador único e incremental
ObjectPath	Varchar(200)	Nome do objeto OLAP
ObjectType	Varchar(20)	Tipo de objeto OLAP (dimensão/métrica)
ChangeType	Tinyint	Tipo de alteração
Date	Datetime	Data de registo do processamento
Notes	Varchar(MAX)	Notas referentes ao processamento
State	Tinyint	Estado do processamento
DateStarted	Datetime	Data de início do processamento
DateProcessed	Datetime	Data de fim do processamento
Error	Tinyint	Campo de sinalização de erros
XMLA	Varchar(MAX)	Campo onde fica armazenado o script em XMLA que será executado
ProcessedTime	Float	Campo calculado através da *Datestarted* e *DateProcessed*

3.6 Extração

Os extratores são os objetos responsáveis pela transição dos dados desde a origem, base de dados OLTP, ficheiros XML, ficheiros Excel, etc., até ao repositório temporário denominado por área de *Staging*.

É da responsabilidade de cada extrator saber onde e como colocar a informação nas tabelas de *Staging*. Nesta fase, tipicamente não existirá transformação dos dados, por questões de desempenho a conetividade com as origens deve ser minimizada o mais possível para evitar impactos negativos que possa existir nos sistemas.

Atualmente existem dois tipos de extratores: pacote de *SSIS* ou procedimentos de *SQL*.

Os extratores do ETL estão catalogados na tabela etl.*Extractor* (ver Tabela 17), onde se pode administrar todas as suas funcionalidades, como ativar ou desativar, alterar a ordem de extração, definir extrações parciais ou totais e alterar a sua calendarização.

Tabela 17 – Extrator

Coluna	Tipo	Descrição
Id	Bigint	Identificador único e incremental
ExtractorScheduleId	Bigint	Chave estrangeira para tabela de calendarizações
ExtractorTypeId	Bigint	Chave estrangeira para a tabela de tipos de extratores, atualmente existem dois tipos SSIS e Procedimentos SQL
Name	Nvarchar(100)	Nome do Extrator
Description	Nvarchar(MAX)	Campo livre para enriquecer informação sobre o extrator, muito utilizado para descrever qual é a origem.
Source	Nvarchar(100)	Nome do pacote de SSIS ou do Procedimento SQL
Location	Nvarchar(150)	Localização ou do pacote SSIS ou do Procedimento, quando NULL o valor de omissão será carregado, caso seja SSIS caminho que está definido no app. config do serviço, caso seja Procedimento SQL base de dados do DW.
TargetTable	Nvarchar(100)	Tabela de destino dos registos extraídos em Staging
TargetColumnDate	Nvarchar(100)	Coluna de data que será endereçada no caso de extrações parciais
DateParcialExtractionIni	Datetime	Valor máximo da última extração, utilizado como controlo para efetuar a próxima extração parciais, servindo este como data de início dessa extração.
DateParcialExtractionEnd	Datetime	Serve como data fim em caso de extração parcial, se for NULL o seu valor assume-se a data atual do sistema
TotalExtration	Bit	Campo de controlo, se a extração é parcial ou total na origem, uma vez assinalado o campo todos os dados da origem serão copiados para Staging.
ExtractorOrder	Int	Define a ordem como o motor chama para extrair
Active	Bit	Ativa ou desativa o extrator. Mesmo que esteja no período de execução calendarizado se estiver desativo não será executado.

A Tabela 17 representa o modelo de configuração de um extrator, que pode ser parametrizado como os exemplos visíveis na Tabela 18.

Tabela 18 – Exemplo de dados de Extratores pela ordem da tabela acima

1	**2**	**1**	**CI**	**Data Center**	**CI.dtsx**	**NULL**	**CI**	**Data**	**2012-04-04**	**NULL**	**1**	**1**	**1**
2	2	1	Serviços	Data Center	Servicos.dtsx	NULL	Servicos	Data	2012-04-04	NULL	0	2	0

A *framework* disponibiliza um componente genérico de SSIS (ver Figura 12 e Figura 13) que alterando os valores das variáveis (ver Tabela 19) e da conetividade da "*Source",* pode ser reutilizável em mais do que um pacote de extração.

Tabela 19 – SSIS variáveis

Name	Scope	Data Type	Descrição
DateExtractionFim	Template	DateTime	Data de extração final em caso de extração parcial
DateExtractionIni	Template	DateTime	Data de extração inicial em caso de extração parcial
DateNameFim	Template	String	Nome da coluna que faz validação da data de extração parcial final
DateNameIni	Template	String	Nome da coluna que faz validação da data de extração parcial inicial
DateTimeFormat	Template	String	
ErroMensagem	Template	Int32	Mensagem de erro do pacote caso exista
ExtractionId	Template	Int32	Id da extração
ExtractorId	Template	Int32	Id do extrator
LastDateExtraction	Template	DateTime	
Query	Template	String	Query SQL que faz a consulta à origem dos dados
RegistosErro	Template	Int32	Número de registos que deram erro
RegistosExtraidos	Template	Int32	Número de registos extraídos
RowCount	Template	Int32	Número de registos retornados da query SQL na origem
Scope	Template	Int32	Scope aplicar na query é origem
StagTable	Template	String	Nome da tabela destino
TotalExtraction	Template	Boolean	Valida se a extração é parcial ou total, em caso de verdadeiro ignora todas as datas para extrações parciais e todos os scopes a aplicar

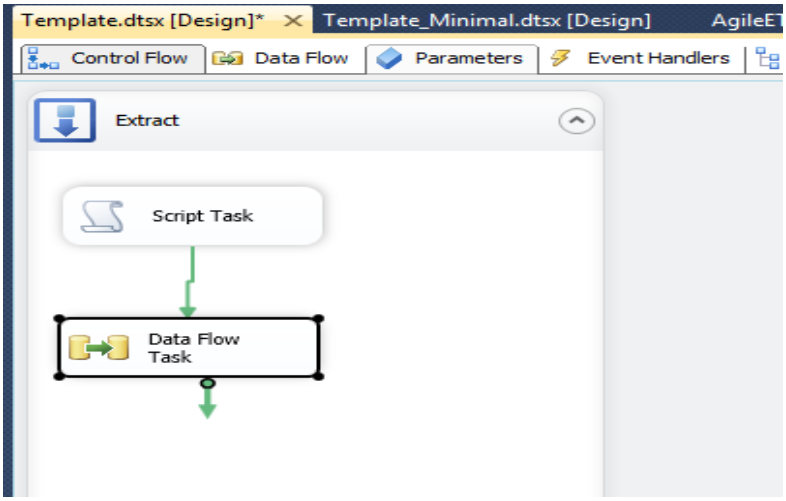


Figura 12 – Extrator Template SSIS

Na Figura 12 são visíveis os componentes que integram o pacote de SSIS, usado como referência na criação de novos processos de extração. Este pacote apenas tem um *Script Task*, que agrega as variáveis de data início e data fim, à consulta feita na origem, para obter os registos a extrair. O outro componente identificado como *Data Flow Task* na Figura 12, pode ser observado na Figura 13 em detalhe, é o componente responsável pelo fluxo de registos desde a origem até ao destino.

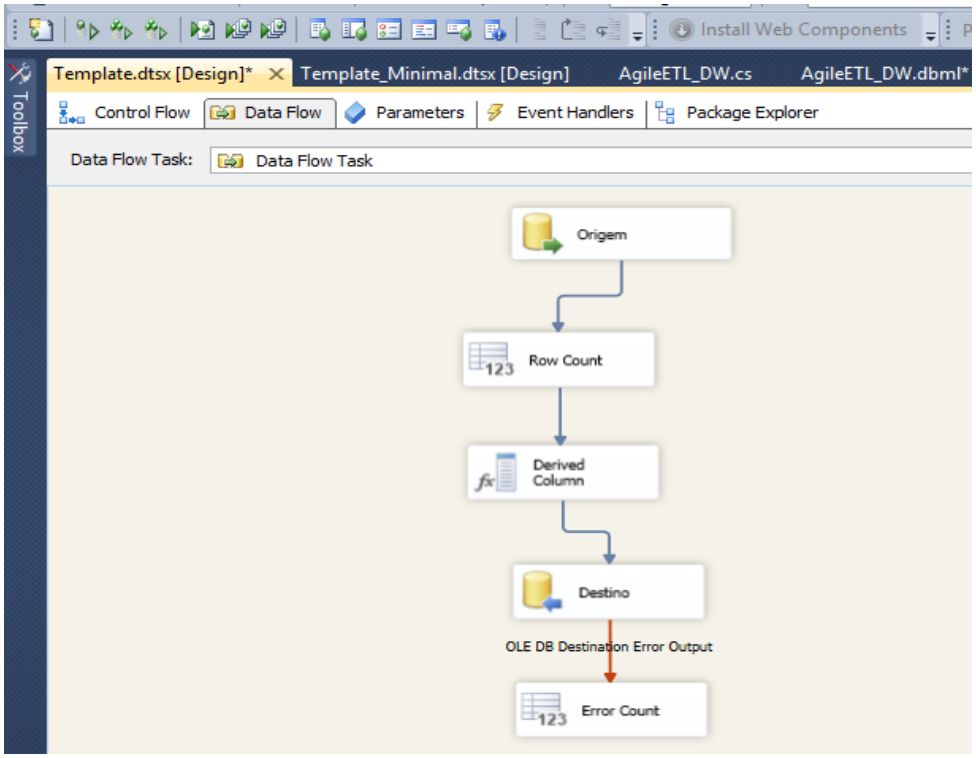


Figura 13 – Fluxo Extrator Template SSIS

3.6.1 Extrações

As extrações são eventos associados aos extratores, de cada vez que um extrator é executado é sinalizado no seu início o estado a 1 que significa que o processo vai arrancar, o identificador único do extrator, o *scope* (seção seguinte) caso exista, a data de extração parcial que o afeta quer de início quer de fim, caso exista, e a hora pelo qual está a ser iniciado. Após o arranque do processo começa a ser executada a extração de registos e o estado passa a 2 e é inserida a data e hora na Tabela 21, quando o processo de extração termina o orquestrador do ETL sinaliza a data e hora de finalização, altera o estado para 4 e marca o número total de registos extraídos. No caso do processo falhar é sinalizado o estado com 3 e a descrição do erro escrita na coluna *Error* da Tabela 21. Este erro não é trabalhado, mas colocado na sua forma original; isto para a informação fornecida ser o mais completa possível, uma vez que a ferramenta é para ser usada por técnicos e não para utilizadores finais.

A Tabela 20 representa os estados possíveis que uma extração pode ter, existindo três ciclos possíveis:

1. Iniciado -> A extrair -> Extraído
2. Iniciado -> A extrair -> Extraído -> Integrado
3. Iniciado -> A extrair -> Erro

Tabela 20 – Estado da Extração

Id	Nome	Descrição
1	Iniciado	O processo foi iniciado
2	A extrair	A extração está a ser executada
3	Erro	Erro na extração
4	Extraido	Extraído com sucesso
5	Integrado	Integrado com sucesso

A Tabela 21 é uma tabela de eventos onde fica armazenada toda a informação sobre as extrações efetuadas pela *framework,* como o número de registos extraídos, estado atual da extração, etc..

Tabela 21 – Extractions, tabela de eventos de extração

Coluna	Tipo	Descrição
Id	Bigint	Identificador único e incremental
ExtractorId	Bigint	Chave estrangeira para tabela de extratores
Scope	Nvarchar(MAX)	Clausura a acrescentar ao *Where* da consulta
DateParcialExtractionIni	Datetime	Caso seja extração parcial é a data de início da extração
DateParcialExtractionEnd	Datetime	Caso seja extração parcial é a data de fim da extração
Init	Datetime	Data de sinalização do extrator
Start	Datetime	Data de arranque dos trabalhos de extração
End	Datetime	Data de finalização da extração
TotalRecords	Bigint	Número de registos extraídos
TotalErrorRecords	Bigint	Número de registos que deram erro
StatusId	Tinyint	Id do estado da extração
Error	Nvarchar(MAX)	Descrição do erro, caso exista.

3.6.2 Scope

Scope foi um nome dado para o conceito elaborado na *framework* de ETL, para restringir o âmbito de extrações, e segmentar a informação não apenas pelas extrações parciais, funcionalidade que também reduz o âmbito da extração, mas através da data inicio e data fim do registo. O conceito de *Scope* existe na *framework* para dar resposta de uma forma ágil a pequenas alterações aos processos de extração. Exemplo de um caso prático, uma extração de alarmes num servidor onde está a ser recorrente eventos do tipo

aviso (“*Warning*”), uma vez que a excessiva informação pode causar o desinteresse pois a procura fica mais difícil, era útil filtrar os alarmes temporariamente a extrair, idealmente só extrair alarmes críticos (“*Critical*”).

Com o *Scope* pode-se facilmente acrescentar itens de restrição nas consultas de SQL, sem ter que ir alterar o pacote de SSIS e ainda desativar quando se quiser sem alteração ao código. Esta alteração é realizada numa tabela de parametrizações designada por *etl*.*Scope* (ver Tabela 22) que contém, o *id* do extrator, o *id* do integrador, a expressão a usar quer no integrador quer no extrator (a coluna pode ter nomes diferentes no ambiente de *Staging* e *Data Warehouse*), o modo de agregação com a restante consulta, o valor pelo qual será filtrado e um campo booleano para ativar ou desativar o *Scope*.

A Tabela 22 representa a estrutura onde fica armazenada o conceito de *Scope*, podendo parametrizar mais âmbitos a considerar em extrações ou integrações.

Tabela 22 – Scope

Coluna	Tipo	Descrição
Nome	Nvarchar(100)	Nome do Scope
LogicalOperator	Nvarchar(50)	Operador lógico a aplicar (>;<;>=;<=)
ExtractorId	Bigint	Id do extrator
IntegratorId	Bigint	Id do integrador
ExpressionExtractor	Nvarchar(MAX)	Expressão ou nome de coluna a utilizar à esquerda do operador lógico no processo de extração
ExpressionIntegrator	Nvarchar(MAX)	Expressão ou nome de coluna a utilizar à esquerda do operador lógico no processo de integração
AgregationOperator	Nvarchar(3)	Disjunção ou conjunção lógica (AND;OR)
Value	Nvarchar(MAX)	Valor a ser inserido na clausura à direita do operador lógico
Active	Bit	Ativa ou desativa o scope

A Tabela 23 representa dois exemplos de *Scopes* aplicados, o 1º exemplo tem como nome “Alarme Critico”, tem o valor lógico “=” para o *id* de extração “3” que faz ligação com a tabela de extratores (Tabela 17) e o *id* de integração “3” também que faz ligação com a tabela integradores (Tabela 24). O valor “1” na coluna “*Active*” representa que o *Scope* está em funcionamento. Estes valores parametrizados serão refletidos na consulta de extração, sendo adicionado o seguinte filtro, ao código de extração, “*AND Categoria = ‘Critical’ ”* e o filtro ao código de integração “AND Criticidade = ‘Critical’ ”.

Tabela 23 – Exemplo dados Scope

Nome	LogicalOp.	Extr.	Inte.	Expr. Ext.	Exp. Int.	AgregationOp.	Value	Active
Alarme Critico	=	3	3	Categoria	Criticidade	AND	Critical	1
Serviços Hoje	=	2	2	Data	Data	AND	GETDATE()	0

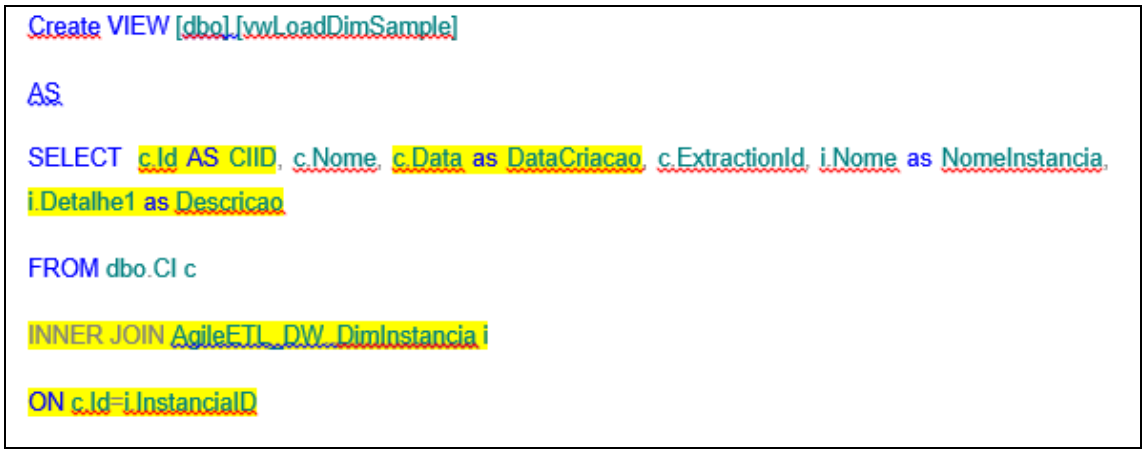
3.7 Integração

Os integradores têm como tarefa passar a informação da área de *Staging* para *Data Warehouse* e transformar os registos de uma forma consolidada para dar resposta ao modelo de negócio.

As integrações podem ser de dois tipos, pacotes de SSIS ou então procedimentos SQL, sendo que a *framework* tira mais partido dos procedimentos SQL (existem grandes limitações nos pacotes SSIS na reutilização de código), assim, foram construídos procedimentos genéricos através de SQL dinâmico que visam em dois tipos de integrações distintas. Estratégia de apagar registos antes de inserir, esta estratégia utilizada tipicamente para as métricas faz uma inserção massiva, apagando os registos que compreendem na sua extração parcial anteriormente, ou se a extração for total apaga toda a tabela e insere os novos registos.

Para a criação de um pacote de integração, aproveitando os componentes que a *framework* oferece, deve-se criar um procedimento em SQL específico para o integrador que chama o procedimento SQL genérico de integração *"[etl].[uspLoadDW_DeleteInsert]".* Este procedimento genérico tem como parâmetros de entrada o *id* da integração, o *id* da extração, o nome da tabela destino no *Data Warehouse*, o nome da tabela ou *view* de *Staging*, existe um requisito particular, o esquema da tabela ou *view* utilizada no carregamento tem que ter o mesmo esquema da tabela de *Data Warehouse* a ser carregada, ou seja todas as colunas tem que ter o mesmo nome e serem do mesmo tipo. Deve-se utilizar sempre que possível as *views* e fazer toda a transformação e enriquecimento necessário para satisfazer o modelo de negócio desenhado no *Data Warehouse*.

A Figura 14 apresenta uma *view* típica de transformação para ser usada no carregamento e transformação de dados para o *Data Warehouse*:



```
Create VIEW [dbo].[vwLoadDimSample]

AS

SELECT  c.Id AS CIID, c.Nome, c.Data as DataCriacao, c.ExtractionId, i.Nome as NomeInstancia,
i.Detalhe1 as Descricao

FROM dbo.CI c

INNER JOIN AgileETL_DW_DimInstancia i

ON c.Id=i.InstanciaID
```

Figura 14 – View de carregamento dados.

A Figura 15 ilustra um exemplo, de um extrator na estratégia de apagar registos antes de inserir.

```
CREATE procedure [dbo].[uspIntegrate_Sample_DeleteInsert]

        @integrationId  bigint,@extractionId  bigint,  @scope  nvarchar(max)  =  null  ,
        @totalNewRecords  int  output,  @totalUpdateRecords  int  output,  @totalErrorRecords  int
        output, @totalDeleteRecords int output

AS

        DECLARE  @dwColumns  nvarchar(max),@stagColumns  nvarchar(max),@mScope
        nvarchar(max),@stagTable  nvarchar(100)='dbo.vwLoadStag',@dwTable
        nvarchar(100)='dbo.DWTAble'

        EXEC  [etl].[uspLoadDW_DeleteInsert]  @integrationId,@extractionId  ,  @dwTable,
        @stagTable,  @stagColumns,  0,  'Data',  'Data',  @scope,  0,  @totalNewRecords  output,
        @totalUpdateRecords output, @totalErrorRecords output, @totalDeleteRecords output

        EXEC etl.OlapGeneratePartitionsUpdate 'MeasureGroup',@integrationId,1
```

Figura 15 – Extrator de métricas

O extrator da Figura 15 chama um procedimento SQL genérico *[etl].[uspLoadDW_DeleteInsert]* disponibilizado pela ferramenta, que consegue dar resposta generalizada através de SQL dinâmico. O procedimento SQL tem como parâmetros de entrada o *id* da integração, o *id* da extração, o nome da tabela no *Data Warehouse*, o nome da tabela em *Staging*, a lista de colunas de uma das duas tabelas (o esquema tem que ser o mesmo), uma coluna do tipo verdadeiro ou falso, chamada *@isTable*, para validar se o nome colocado em *@stagTable* é de uma *view* ou de uma tabela (deve ser *view* caso se pretenda efetuar transformação e consolidação de dados), o nome da coluna de data no

Data Warehouse e em *Staging* para filtragem em caso de extrações parciais, o *Scope* para filtragem mais específicas. Um dos parâmetros mais complexos é o *@variation,* este parâmetro tem como objetivo verificar se o número de registos extraídos está em conformidade com o número de registos integrados quando sinalizado a "0", esta validação no final da integração é desativada, senão caso seja colocado "1" o número de registos extraídos tem que ser igual ao número de registos integrados, caso seja "2" o número de registos integrados é o dobro dos registos extraídos, e assim sucessivamente, pois funciona como multiplicador ao número de registos:

Integração com Sucesso -> Nº Registos Integrados= Nº Registos Extraídos * @variation

O procedimento SQL tem como parâmetros de saída o número de novos registos inseridos, o número de registos apagados e, caso exista registos com erro também devolve o número de registos assinalados com erros.

Existe outra estratégia de carregamento de dados que nunca apaga registos no *Data Warehouse*, apenas insere se forem novos registos ou altera caso exista já o registo, este carregamento necessita de uma chave de verificação que controla através da chave primária se o registo é novo ou não. Esta estratégia de carregamento é utilizada quando os dados do *Data Warehouse* não podem ser apagados, ou seja, sempre que é inserido um registo ele verifica se existe, e se existir, faz a alteração, senão insere como novo, utilizando as funcionalidades de *Merge* do SQL Server. É utilizado nas dimensões de análise principalmente ou em métricas que têm alterações ao longo da sua existência.

Existe um procedimento em SQL, genérico, que disponibilizado para ser utilizado em extratores deste tipo chamado *[etl].[uspLoadDW_ByPk]* (ver Figura 16). O procedimento SQL tem como parâmetros de entrada o *id* da integração, o *id* da extração, o nome da tabela no *Data Warehouse*, o nome da tabela em *Staging*, a lista de colunas de uma das duas tabelas (o esquema tem que ser o mesmo), uma variável booleana chamada *@isTable*, para validar se o nome colocado em *@stagTable* é de uma *view* ou de uma tabela (preferencialmente

sempre *view* para ser lá efetuado a transformação e consolidação dos dados), o nome da coluna de data no *Data Warehouse* e em *Staging* para filtragem em caso de extrações parciais, o *Scope* para filtragem mais específicas, o nome da chave primária no *Data Warehouse* e em *Staging* para fazer ligação entre registos e inferir quais são novos ou não, o parâmetro de controlo de número de registos igual com o comportamento igual ao da função de apagar e inserir novos registos. O procedimento SQL tem como parâmetros de saída o número de novos registos inseridos, o número de registos alterados, o número de registos apagados e caso exista registos com erro também devolve o número de registos assinalados com erros.

```
CREATE procedure [dbo].[uspIntegrate_Sample_LoadDW_ByPk]

        @integrationId bigint,@extractionId bigint, @scope nvarchar(max) = null ,
        @totalNewRecords int output, @totalUpdateRecords int output, @totalErrorRecords int
        output, @totalDeleteRecords int output

AS

        DECLARE stagColumns nvarchar(max),

        @stagTable nvarchar(100)='dbo.vwLoadStag',

        @dwTable nvarchar(100)='dbo.DWTAble'

    EXEC [etl].[uspLoadDW_ByPk] @integrationId @extractionId, @dwTable

    , @stagTable, 0, @stagColumns, 'CIID', 'CIID', 'Data', 'Data', @scope, 0, 1, @totalNewRecords
output, @totalUpdateRecords output, @totalErrorRecords output, @totalDeleteRecords output

        EXEC etl.OlapRegisterChange 'Dimension Name', 'DIM'
```

Figura 16 – Integrador de Dimensões

Os integradores do ETL estão catalogados na tabela *etl.Integrator* (ver Tabela 24), onde se pode administrar toda as suas funcionalidades, como ativar ou desativar, alterar o extrator correspondente.

Tabela 24 – Integrator, tabela de integradores de dados

Coluna	Tipo	Descrição
Id	Bigint	Identificador único e incremental
ExtractorId	Bigint	Chave estrangeira para tabela de extratores
IntegratorTypeId	Bigint	Identificador do tipo de Integrador
Name	Nvarchar(100)	Nome do integrador
Description	Nvarchar(MAX)	Campo livre para enriquecer informação sobre o integrador, muito utilizado para descrever qual é a origem.
Source	Nvarchar(100)	Nome do pacote de SSIS ou do Procedimento SQL
Location	Nvarchar(150)	Localização do pacote SSIS ou do Procedimento, quando NULL o valor de omissão será carregado, caso seja SSIS caminho que está definido no app.config do serviço, caso seja Procedimento SQL base de dados do DW.
TargetTable	Nvarchar(100)	Tabela de destino dos registos extraídos em Data Warehouse
Active	Bit	Ativa ou desativa o integrador. Mesmo que exista já uma extração realizada para o integrador se estiver desativo não será executado.

3.7.1 Integrações

As integrações são eventos associados aos integradores, cada vez que são executados é sinalizado no seu início o estado a 1 que significa que o processo vai arrancar, o identificador único do integrador, o identificador da extração associada, o Scope caso exista, a data de extração parcial que o afeta quer de início quer de fim caso também exista e a hora pelo qual está a ser iniciado. Depois quando o processo de ETL arranca e começa a ser executada a integração dos registos no *Data Warehouse* o estado passa a 2 e é inserida a data e hora na tabela de integrações (ver Tabela 26) à qual deu início esta fase, depois quando o processo de integração acaba o orquestrador do ETL sinaliza a data e hora de finalização, altera o estado para 5 e marca o número total de registos inseridos, total de registos atualizados, total de registos apagados. Caso o processo falhe é sinalizado o estado com 3 e a descrição do erro escrita na coluna Error da tabela de integrações (ver Tabela 26). À semelhança com as extrações este erro não é trabalhado.

A Tabela 25 representa os estados possíveis que uma integração pode ter, existindo dois ciclos possíveis:

1. Iniciado -> A integrar -> Integrado
2. Iniciado -> A integrar -> Erro

Tabela 25 – Estado da Integração

Id	Nome	Descrição
1	Iniciado	O processo foi iniciado
2	A integrar	A integração está a ser executada
3	Erro	Erro na integração
5	Integrado	Integrado com sucesso

Tabela 26 – Integrations, tabela de eventos de integração de dados

Coluna	Tipo	Descrição
Id	Bigint	Identificador único e incremental
IntegratorId	Bigint	Chave estrangeira para tabela de integradores
ExtractionId	Bigint	Chave estrangeira para a tabela de extrações
Scope	Nvarchar(MAX)	Clausura a acrescentar ao *Where* da consulta
DateParcialIntegrationIni	Datetime	Caso seja integração parcial é a data de início da integração
DateParcialIntegrationEnd	Datetime	Caso seja integração parcial é a data de fim da integração
Init	Datetime	Data de sinalização do integrador
Start	Datetime	Data de arranque dos trabalhos de integração
End	Datetime	Data de finalização da integração
TotalNewRecords	Bigint	Número de novos registos inseridos
TotalUpdatedRecords	Bigint	Número de registos alterados
TotalDeletedRecords	Bigint	Número de registos apagados
TotalErrorRecords	Bigint	Número de registos que deram erro
StatusId	Tinyint	Id do estado da integração
Error	Nvarchar(MAX)	Descrição do erro, caso exista.

3.8 Modelo de indicadores

O modelo de indicadores é uma arquitetura ágil de armazenamento de métricas que permite ao sistema incorporar novos indicadores sem ter que criar novas tabelas ou novos grupos de métricas no OLAP. Com recursos e estruturas distintas em várias origens de dados, a consolidação passa por transformar toda a informação recolhida e centralizada numa tabela, com um esquema genérico, *FactMetricasRecursosTI* (ver Tabela 27), onde toda a informação servirá um único grupo de métricas no OLAP. Existem outras três tabelas no *Data Warehouse* que complementam a restante arquitetura.

A tabela *FactMetricasRecursoTI* (ver Tabela 27), onde ficam armazenados os registos de métricas, tem relação com a dimensão data, hora e pode conter um recurso associado, este recurso será uma dimensão que será associado ao registo, usando a nomenclatura da certificação ISO 20000 ITIL (Van Bon & de Jong, 2007), seriam os *configuration items* os valores associados a esta dimensão. Os campos *Min, Max, Avg, Sum* e *Last*

inicialmente serão armazenados todos com o mesmo valor, valor do indicador na origem. A coluna *Periodo* estará preenchida a vazio e o *Count* a "1", existe ainda atributos de relação com a dimensão indicador que terá dados específicos sobre o indicador da análise, e como podem existir métricas que necessitam de mais informação para analisar, a dimensão *DimInstancia* tem campos de detalhe que podem armazenar essa informação personalizada. Para ficarem marcados todos os registos existe uma coluna designada por *IntegrationId* que sinaliza qual foi a integração que fez a sua transição para o *Data Warehouse.*

Tabela 27 – FactMetricasRecursosTI, tabela genérica de métricas

Coluna	Tipo	Descrição
IndicadorDWID	bigint	Chave estrangeira para dimensão DimIndicador
DataDWID	int	Chave estrangeira para dimensão DimData
HoraDWID	int	Chave estrangeira para dimensão DimHora
InstanciaDWID	bigint	Chave estrangeira para dimensão DimInstancia
CIID	bigint	Valor do Configuration Item, caso exista em regime da nomenclatura ISO 20000
OrigemID	varchar(50)	Id do facto na origem
Min	float	Valor min dos factos em caso de agregação, senão todos os valores serão preenchidos de igual modo
Max	float	Valor max dos factos em caso de agregação, senão todos os valores serão preenchidos de igual modo
Avg	float	A média dos factos em caso de agregação, senão todos os valores serão preenchidos de igual modo
Sum	float	
Last	float	Último valor do facto em caso de agregação, senão todos os valores serão preenchidos de igual modo
Count	bigint	1 se o facto não for agregado, caso seja agregado é o número de registos agregados
Periodo	char(1)	Tipo de agregação, 'H' – agregação à hora 'D' – agregação ao dia. Valor Nulo quando o facto não é nenhuma agregação
Data	datetime	Data do facto
IntegrationId	bigint	Id da integração que colocou o facto no Data Warehouse

Na agregação de dados é colocado na coluna *Periodo* o tipo de agregação "D" para agregação ao dia, e "H" para agregação à hora, depois o motor de agregação com a gama de valores do período de agregação preenche a coluna de *Min* com o menor valor da amostra de análise, preenche o *Max* com o maior valor, faz a média de todos os valores e coloca em *Avg*, faz o somatório e preenche a coluna *Sum*, coloca o último valor em *Last,* finalmente é preenchido o *Count* com o número de registos agregados. Desta forma consegue-se dar resposta à análise com mais longevidade, mas com menos nível de granularidade, garantindo mais espaço em disco no servidor.

A tabela DimIndicador (ver Tabela 28) armazena informação sobre o indicador para dar contexto à métrica armazenada na tabela *FactMetricasRecursosTI* (ver Tabela 27), estas duas tabelas complementam-se na análise e uma não pode ser analisada sem a outra. Exemplo disso é o valor "0,6" armazenado em *Sum* na tabela *FactMetricasRecursosTI* (ver Tabela 27), *com* o *Count* a "1", *Periodo* "Nulo" e *Data* de hoje, não se consegue inferir nada com a informação. Contudo, com esta informação, se indicado que o indicador é a taxa de ocupação de CPU então a informação pode mostrar que o valor assinalado pode ser critico, porque o valor refere-se a uma excessiva ocupação de CPU num determinado recurso.

Tabela 28 – DimIndicador

Coluna	Tipo	Descrição
IndicadorDWID	bigint	Identificador único e incremental
Nome	varchar(50)	Nome do indicador
Objecto	varchar(50)	Tipo de objeto, criado para agrupar indicadores do mesmo tipo
Tipo	varchar(50)	Tipo de indicador é para sinalizar se é percentagem, inteiro, decimal, moeda
Formato	varchar(50)	Formato que deve ser aplicado o valor
Notas	varchar(MAX)	Campo de descrição
Origem	varchar(50)	Armazenar o servidor de origem do facto
Tabela	varchar(50)	Armazenar a tabela de origem do facto
Coluna	varchar(50)	Armzenar a coluna de origem do facto

DimInstancia (ver Tabela 29) tem a funcionalidade de enriquecer a métrica com informação útil, ou seja se no exemplo anterior se guardavam informações relativas às taxas de ocupação de CPU num dado recurso, na tabela *DimInstancia* (ver Tabela 29), pode-se guardar informação sobre esse registo que daria mais detalhe na análise. Como a localização do recurso no momento do extração dos dados, o utilizador que fez disparar o CPU, ou até mesmo a aplicação que teve maior peso na taxa de ocupação.

Tabela 29 – DimInstancia

Coluna	Tipo	Descrição
InstanciaDWID	bigint	Identificador único e incremental
Nome	varchar(50)	Nome da instancia
Tipo	varchar(50)	Tipo de instancia
Detalhe1	varchar(255)	Campo livre para inserir detalhe sobre o facto
Detalhe2	varchar(255)	Campo livre para inserir detalhe sobre o facto
Detalhe3	varchar(255)	Campo livre para inserir detalhe sobre o facto
Detalhe4	varchar(255)	Campo livre para inserir detalhe sobre o facto
Data	datetime	Campo data para controlo de extrações parciais
IntegrationId	bigint	Id da integração que colocou o membro da dimensão no Data Warehouse

3.9 Agregação de dados

Como o armazenamento em sistemas corporativos tem um custo elevado e a análise de dados históricos não tem muitas vezes grande necessidade de detalhe, o modelo de indicadores tem a funcionalidade nativa de agregação de indicadores, como forma de disponibilizar mais espaço em disco dos servidores onde o *Data Warehouse* está alojado.

A calendarização descrita no ponto 3.5.4 tem duas opções distintas de agregação, agregação à hora, que agrega os registos independentemente do nível de detalhe ao nível da hora, ou seja, na análise de um indicador de desempenho "*Taxa de Ocupação de CPU*", o agente do sistema está a recolher informação de minuto a minuto num servidor em concreto e o processo de ETL guarda essa informação no *Data Warehouse*, com esse detalhe. Com o processo de agregação e com a calendarização específica o modelo de indicadores substitui no período selecionado os dados que estavam com um nível de detalhe ao minuto, agregando pelas suas médias, somatórios e número de registos inseridos, ao nível da hora. Com esta agregação o nível de detalhe que era ao minuto perde-se, e passa o detalhe a ser à hora, diminuindo com isto o número de registos no *Data Warehouse*. Com a agregação, o nível de detalhe máximo de informação que pode ser irrelevante para muitos dos casos, porque saber a "*Taxa de Ocupação de CPU*" num determinado minuto à 5 anos atrás não traz uma mais-valia significativa em comparação com o custo de armazenamento.

Atualmente existe dois tipos de agregação dos dados, a agregação à *Hora* e ao *Dia.* A agregação à *Hora* agrupa toda a informação respeitante ao nível da hora independente do nível de detalhe existente, ao *Dia,* que depois da agregação à hora, coloca a informação num patamar de granularidade inferior, somando todas as horas do dia e respeitantes métricas.

Esta informação fica armazenada na tabela do *Data Warehouse FactMetricasRecursosTI_Archive* do modelo de indicadores, tabela igual à *FactMetricasRecursosTI* (ver Tabela 27), apenas foi separada por questões de desempenho.

A *framework* nativamente contém procedimentos em SQL de agregação, um desses procedimentos genéricos é o *etl.uspArchiveMeasures* (ver Figura 17), que contém como parâmetros de entrada a data de início e fim do período de agregação, a tabela a agregar, o tipo de agregação para hora é 'H' para o dia é 'D', e tem como parâmetro de saída o total de registos arquivados.

```
Create PROCEDURE [etl].[uspArchiveMeasures] @dateIni date, @dateEnd date, @table nvarchar(50), @archiveType
nvarchar(50)='H' , @totalRowCount bigint output

As declare @data datetime,@msgLog as nvarchar(max),@msgErro as nvarchar(max),@rowCount as bigint,@cmd as
nvarchar(max),@scope as nvarchar(max),@indicadoresDWID as nvarchar(max), @archiveTable as nvarchar(255);

SET @archiveTable='dbo.FactMetricasRecursosTI_Archive'

IF (@archiveType='D') BEGIN

SELECT @indicadoresDWID=[etl].[fn_GetTableIndicadores] (@table)END ELSE

BEGIN SET @cmd='select @dateIni=DATEDIFF(dd, 0, (SELECT MIN(Data) FROM '+@table  +' ))' END

EXEC sp_executesql @cmd,N' @dateIni datetime output', @dateIni = @dateIni output

SET @totalRowCount=0;

declare cs cursor for  select data from dimdata where data between @dateIni and @dateEnd

open cs fetch next from cs into @data while @@FETCH_STATUS<>-1

BEGIN declare @status nvarchar(500)

 IF(@archiveType='D'  BEGIN

EXEC [etl].[uspArchiveMeasuresDay] @table,@data,NULL,NULL,@archiveTable,@indicadoresDWID,@rowCount
output

          SET @totalRowCount=@totalRowCount + @rowCount

     END ELSE BEGIN

    EXEC [etl].[uspArchiveMeasuresHour] @table ,@data , NULL , NULL ,@archiveTable , @rowCount output

     SET @totalRowCount=@totalRowCount + @rowCount END

     declare @d2 datetime

     set @d2=@data+1

     fetch next from cs into @data

ENDclose cs

deallocate cs
```

Figura 17 – Procedimento SQL para Agregação de métricas

O procedimento SQL *uspArchiveMeasures* (ver Figura 17) chama dois outros procedimentos SQL mais específicos, que mediante o parâmetro de entrada do período a agregar, executa o procedimento *uspArchiveMeasuresHour* (ver Figura 18**Error! Reference source not found.**), se for para agregar à hora, e com os parâmetros de entrada do nome da tabela agregada, a data inicio e fim do período que se pretende agregar a informação, um nome de um índice (caso valorize a pesquisa de dados na tabela de agregações), o nome da tabela para onde a informação irá ser agregada e é devolvido no procedimento o número de registos agregados.

```
CREATE PROCEDURE [etl].[uspArchiveMeasuresHour] (@table as nvarchar(100), @dateIni datetime,
@dateEnd datetime=null, @indexName nvarchar(100)=null, @archivetable
nvarchar(100)='dbo.FactMetricasRecursosTI_Archive', @rowCount bigint output) as

DECLARE @date1 date, @date2 date, @datadwid int, @cmdstring nvarchar(max),@cmdIdx
nvarchar(200)='',@msgLog nvarchar(300)='';

 SET @date1=@dateIni

   IF(@dateEnd IS NULL)

    BEGIN

      SET @date2=dateadd(day,1,@date1)

    END

   IF(@indexName IS NOT NULL AND @indexName <> '')

   BEGIN

     SET @cmdIdx=' With (Index (' + @indexName + '))'

   END

 SELECT @datadwid=etl.fn_DataDWID(@date1)

EXEC [etl].[uspDeleteBeforeArchiveRows] @table , @date1 , @date2 , 'H' , @archivetable

 SET @cmdstring ='INSERT INTO ' + @archivetable +' (IndicadorDWID, DataDWID, HoraDWID,
InstanciaDWID, CIDWID, CIID, OrigemID, [Min], [Max], [Avg], [Sum],[Count],Periodo,Data,IntegrationId)

 SELECT f.IndicadorDWID,'' + cast(@datadwid as nvarchar(50))+ '' as DataDWID, h.HoraBaseId as
HoraDWID, f.InstanciaDWID,f.CIDWID,f.CIID ,0 as OrigemID,MIN(f.[Min]) as [Min],MAX(f.[Max])as
[Max],AVG(f.[Avg]) as [Avg],SUM(f.[Sum]) as [Sum],COUNT(*) as [Count],''H'' as Periodo,'' + cast(@date1
as nvarchar(100)) + ''' + cast(h.Hora as nvarchar(50)) + ':00:00' as Data,0 as IntegrationId FROM (select
* from ' + @table + @cmdIdx + 'where data>='' + cast(@date1 as nvarchar(100)) + '' and data < '' +
cast(@date2 as nvarchar(100)) + '' ) f INNER JOIN dbo.DimHora h ON f.HoraDWID = h.HoraDWID
GROUP BY f.IndicadorDWID ,f.CIDWID, f.CIID,f.InstanciaDWID,h.HoraBaseId, h.Hora '

EXEC sp_executesql @cmdstring ; SET @rowCount=@@rowcount

exec uspCheckMeasuresArchive @table,'H', @date1,@date2,@indexName
```

Figura 18 – Procedimento SQL para agregação de métrica à hora

No caso do objetivo ser a agregação ao dia, então é chamado o procedimento SQL etl.*uspArchiveMeasuresDay* (ver Figura 19) que contém como parâmetro de entrada o nome da tabela a agregar, data de início e fim do período de agregação, o nome de um índice (caso valorize a pesquisa de dados na tabela de agregações), o nome da tabela onde os dados agregados

serão inseridos, o *id* do indicador de análise a agregar e será devolvido o número de registos que foram agregados.

```
Create PROCEDURE [etl].[uspArchiveMeasuresDay] (@table as nvarchar(100),@dateIni
datetime,@dateEnd datetime=null,@indexName nvarchar(100)=null,@archivetable
nvarchar(100)='dbo.FactMetricasRecursosTI_Archive',@indicadoresDWID nvarchar(max),@rowCount
bigint output) as

DECLARE @date1 date,@date2_date,@datadwid int,@cmdstring nvarchar(max),@cmdIdx
nvarchar(200)='',@msgLog nvarchar(300)=''; SET @date1=@dateIni

IF(@dateEnd IS NULL) BEGIN SET @date2=dateadd(day,1,@date1) END IF(@indexName IS NOT
NULL AND @indexName <>'')

        BEGIN SET @cmdIdx=' With (Index (' + @indexName + '))'END

SELECT @datadwid=etl.fn_DataDWID(@date1) EXEC
[etl].[uspDeleteBeforeArchiveRows] @table,@date1,@date2,'D',@archivetable,@indicadoresDWID

SET @cmdstring ='INSERT INTO ' + @archivetable + '
(IndicadorDWID,DataDWID,HoraDWID,InstanciaDWID,CIDWID,CIID,OrigemID,[Min],[Max],[Avg],[Sum],[C
ount],Periodo,Data,IntegrationId) SELECT f.IndicadorDWID ,'" + cast(@datadwid as nvarchar(50))+"' as
DataDWID,0 as HoraDWID,f.InstanciaDWID,f.CIDWID,f.CIID,0 as OrigemID,MIN(f.[Min]) as
[Min],MAX(f.[Max])as [Max],AVG(f.[Avg]) as [Avg],SUM(f.[Sum]) as [Sum],COUNT(*) as [Count],"D" as
Periodo,'" + cast(@date1 as nvarchar(100)) +"' as Data,0 as IntegrationId FROM

(select * from ' + @archivetable + @cmdIdx + ' where data>='" + cast(@date1 as nvarchar(100))"' and
data < '" + cast(@date2 as nvarchar(100)) +"' and IndicadorDWID IN (' + @indicadoresDWID + ')' + '
AND Periodo="H") f

GROUP BY f.IndicadorDWID,f.CIDWID,f.CIID,f.InstanciaDWID '

EXEC sp_executesql @cmdstring;

SET @rowCount=@@rowcount

EXEC uspCheckMeasuresArchive @table,'D', @date1,@date2,@indexName
```

Figura 19 – Procedimento SQL para agregação de métrica ao dia

Depois do processo de agregação de dados a *framework* de ETL executa uma tarefa de limpeza, que retira a informação previamente agregada no *Data Warehouse*. Este processo é executado num procedimento SQL chamado *etl.uspPutrgeAggregatedRows* (ver Figura 20) e tem como parâmetros de entrada o nome da tabela que foi agregada, a data de início e fim do período de agregação, o tipo de agregação “D” ou “H”, o nome do índice caso exista na tabela agregada e é devolvido o número de registos a apagar.

```
CREATE PROCEDURE [etl].[uspPurgeAggregatedRows]

@table nvarchar(50), @currentAggregateDate datetime=null, @lastAggregateDate datetime=null, @archiveType
nvarchar(50)=null, @indexName nvarchar(50)='', @totalRowCount bigint output AS BEGIN DECLARE @cmd
nvarchar(max),@rowCount bigint @reCall bit=0,@msgLog nvarchar(max),@msgErro
nvarchar(max),@indicadoresDWID nvarchar(max),@archiveTable nvarchar(255),@scope nvarchar(max) ;SET
@totalRowCount=0; SET @cmd='select @currentAggregateDate=DATEDIFF(dd, 0, (SELECT MIN(Data) FROM'
+ @table + ' ))' EXEC sp_executesql @cmd,N' @currentAggregateDate datetime output', @currentAggregateDate =
@currentAggregateDate output DECLARE @currentAggregateNextDay datetime
IF(REPLACE(@table,'dbo.','')='FactDisponibilidadeAplicacional') BEGIN SET
@archiveTable='dbo.FactDisponibilidadeAplicacional_Archive' END ELSE BEGIN SET
@archiveTable='dbo.FactMetricasRecursosTI_Archive' SELECT @indicadoresDWID=[etl].[fn_GetTableIndicadores]
(@table) END

  WHILE @currentAggregateDate <= @lastAggregateDate

        BEGIN SET @cmd='' SET @rowCount=0

 SET @currentAggregateNextDay=DATEDIFF(dd, 0, DATEADD(day, 1 , @currentAggregateDate ) )

declare @status nvarchar(500), @msg nvarchar(max)='Não é possivel continuar a purge (uspPurgeAggregatedRows) .
data ' + cast(@currentAggregateDate as nvarchar(50)) + ' pois a base dados ' + db_name() + ' está em';

exec [etl].[AssertReadyToRun] @msg ,@status output IF(@archiveType IS NOT NULL) BEGIN

   IF(@reCall=0) BEGIN EXEC uspCheckMeasuresArchive @table,@archiveType, @currentAggregateDate ,
@currentAggregateNextDay, @indexName END

 SELECT @indicadoresDWID=[etl].[fn_GetTableIndicadores](@table)

SET @cmd= 'IF EXISTS(SELECT fac.data FROM '+ @archiveTable +' fac WHERE fac.data >= '''+
cast(@currentAggregateDate as nvarchar(50)) + ''' AND fac.data < ''' + cast(@currentAggregateNextDay as
nvarchar(50)) + '''+ @scope + ' ) AND EXISTS (SELECT facLast.data FROM ' + @table + ' facLast WHERE
facLast.data >= ''' + cast(@currentAggregateDate as nvarchar(50)) +''' AND facLast.data < '''+
cast(@currentAggregateNextDay as nvarchar(50)) +''') BEGIN delete top(5000000) FROM ' +
REPLACE(@table,'dbo.vw','dbo.') + 'WHERE Data >= '''+ cast(@currentAggregateDate as nvarchar(50)) +''' AND Data
< ''' + cast(@currentAggregateNextDay as nvarchar(50)) +'''END'

END ELSE BEGIN SET @cmd= ' delete top(5000000) FROM ' + REPLACE(@table,'dbo.vw','dbo.') + 'WHERE Data >=
'''+ cast(@currentAggregateDate as nvarchar(50)) ''' AND Data < ''' + cast(@currentAggregateNextDay as nvarchar(50))
+ '''' END

EXEC sp_executesql @cmd ; SET @rowCount=@@rowcount @totalRowCount = @totalRowCount + @rowCount

SET @msgLog='Foram apagados ' + cast(@rowCount as nvarchar(255)) +' registos da tabela ' + @table + ' devido à
agregação do Tipo ' + @archiveType + ' na data ' + cast(@currentAggregateDate as nvarchar(255)) EXEC
[etl].[WriteLog] 'Purge de metricas' ,@msgLog,null,6,0

IF(@rowCount<>5000000) BEGIN SET @currentAggregateDate=DATEDIFF(dd, 0, DATEADD(day,1,
@currentAggregateDate))SET @reCall=0

END
```

Figura 20 – Procedimento SQL para limpeza de dados agregados

Para garantir o processo de agregação de dados e a sua qualidade, existe um outro conjunto de procedimentos que visam garantir se os dados que estão apagados já foram agregados corretamente. A garantia do processo é

conseguida através do cálculo dos valores agregados com os valores de nível granular mais baixo que têm que ser iguais ou bastante aproximados, com a média ou o somatório dos valores o número de registos de análise tem que ser igual, e a contagem de registos existentes tem que ser menor à dos agregados. Caso estes pressupostos não estejam verificados é iniciado uma exceção e o processo de agregação ou de limpeza é interrompido. Este procedimento designa-se por *etl.uspCheckMeasuresArchive* (ver Figura 21) e tem como parâmetro de entrada o nome da tabela agregada, o tipo de agregação “H” ou “D”, a data inicio e fim do período de agregação, e o nome do índice da tabela caso exista para melhorar o desempenho da execução do controlo.

```
CREATE proc [etl].[uspCheckMeasuresArchive]

(@table as nvarchar(100), @archiveType nvarchar(50), @date1 datetime, @date2 datetime=null , @indexName
nvarchar(100)=null)

asdeclare @somatorio1 float, @somatorio2 float, @numReg1 int , @numReg2 int , @countReg1 int, @countReg2 int,
@media1 float, @media2 float, @cmd nvarchar(max), @cmdIdx nvarchar(255)='', @msg nvarchar(max) ;

IF(@indexName IS NOT NULL AND @indexName <> '')BEGIN SET @cmdIdx=' With (Index (' + @indexName + '))'
END SET @cmd='select
@somatorio1=Sum([SUM]),@countReg1=Count(*),@numReg1=Sum([Count]),@media1=Avg([Avg]) from ' +  @table +
@cmdIdx +'where data >= ''' + cast(@date1 as nvarchar(50)) + ''' and data < ''' + cast(@date2 as nvarchar(50)) + '''+'
and Periodo IS NULL' EXEC sp_executesql @cmd,N' @somatorio1 float output, @countReg1 int output, @numReg1
int output, @media1 float output', @somatorio1 = @somatorio1 output, @countReg1 = @countReg1 output,
@numReg1 = @numReg1 output, @media1 = @media1 output

Select @somatorio2=Sum(f.[SUM]),@countReg2=Count(*) , @numReg2=Sum(f.[COUNT]) ,
@media2=Sum(f.[SUM])/Sum(f.[COUNT]) from dbo.FactMetricasRecursosTI_Archive f inner join dbo.DimIndicador I
ON  i.IndicadorDWID=f.IndicadorDWID where f.data>=@date1 and f.data <@date2 and f.Periodo=@archiveType and
i.TabelaDW=Replace(@table,'dbo.','')

SET @msg='Num reg differ - Arquivamento de metricas tipo ' + @archiveType + ' para a tabela' +  @table + ' entre ' +
cast(@date1 as nvarchar(50)) +' e ' + cast(@date2 as nvarchar(50))

if abs(isnull(@numReg2,0)-isnull(@numReg1,0))>0 RAISERROR (@msg, 16, -- Severity.1); -- Second substitution
argument if @countReg1=0 begin return end select @date1 as dataIni,@countReg1 as countReg1,@numReg1 as
numReg1,@media1 as media1,@somatorio1 as somatorio1 select @date1 as dataIni,@countReg2 as
countReg2,@numReg2 as numReg2,@media2 as media2,@somatorio2 as somatorio2--, @count2*1.0/@count3*1.0,
@count1/@count3*1.0 SET @msg='Avg differ - Arquivamento de metricas tipo ' + @archiveType + ' para a tabela'

if abs(isnull(@media2,0)-isnull(@media1,0))>1 RAISERROR (@msg, 16, -- Severity.1); -- Second substitution
argument  SET @msg='Sum differ  - Arquivamento de metricas tipo ' + @archiveType + ' para a tabela' +  @table

if abs(isnull(@somatorio2,0)-isnull(@somatorio1,0))>.01

 RAISERROR(@msg,16, -- Severity.1); -- Second substitution argument.

SET @msg='Count error  - Arquivamento de metricas tipo' + @archiveType + ' para a tabela' +  @table + ' entre' +
cast(@date1 as nvarchar(50)) +' e ' + cast(@date2 as nvarchar(50))

if isnull(@countReg1,0)<isnull(@countReg2,0) RAISERROR (@msg,16, -- Severity.1 -- State.);  END
```

Figura 21 – Procedimento SQL para garantir qualidade da agregação

A *framework* tem outros procedimentos como *Etl.AssertBetwenn* que analisa se o valor real está entre dois valores espectáveis, *Etl.AssertEqual* que verifica se o valor real é igual ao esperado *e Etl.AssetIntegrationRowCount* que verifica se o número de registos extraídos e integrados são iguais. Estes procedimentos de asserção servem para validar afirmações e caso estas não sejam verdadeiras causar exceções que serão trabalhadas individualmente. Tipicamente têm como parâmetro de entrada o valor a analisar, o valor esperado de retorno da função e a mensagem de erro, para o caso de o valor real ser diferente do espectável.

3.10 Processamento OLAP

O ETL contempla o processamento OLAP executando um pacote de SSIS sempre que estiver registado um pedido de processamento na tabela *OlapChangeLog* (ver Tabela 16). Este processo de calendarizarão, explicado no capítulo 3.5.6, é efetuado por cada integrador que coloca na tabela de eventos do OLAP o estado, se for "0" sinaliza que ainda não foi processado, e um código de processamento que a *framework* ajuda à construção através de SQL dinâmico em XMLA. Criado o registo na tabela *OlapChangeLog* (ver Tabela 16), o motor executa o pacote de processamento em SSIS (ver Figura 22) para cada um dos registos encontrados, e coloca o seu estado a "1" caso exista erros ou então a "2" caso seja executado com sucesso.

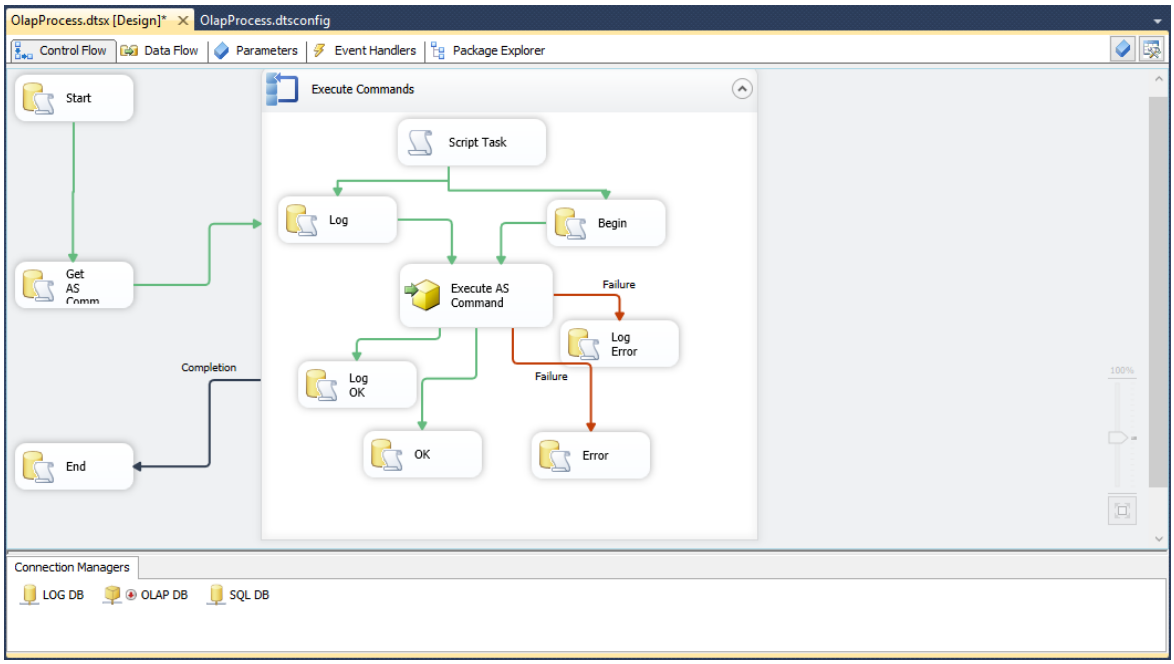


Figura 22 – Pacote SSIS Processamento OLAP

A *framework* disponibiliza os seguintes procedimentos para centralizar os processos e reutilizar o código:

- *OlapCommandBegin* – Marca na tabela de eventos que o processamento foi iniciado;
- *OlapCommandError* – Sinaliza com erro e escreve o erro na tabela de eventos;

- *OlapCommandOk* – Finaliza o processamento com sucesso e regista na tabela;
- *OlapGeneratePartitionsUpdate* – Calendariza partições para processar;
- *OlapPendingChange* – Recolhe todos os objetos marcados para processar;
- *OlapRegisterChange* – Agenda um objeto para processar.

3.11 Monitorização e Eventos

Existe uma tabela de eventos centralizada que pretende agregar toda a informação e estado da *framework* ETL. A *etl.Log* (ver Tabela 30) deve conter informação de todos os processos como mensagens de monitorização, erros ou avisos da aplicação.

Tabela 30 – etl.Log

Coluna	Tipo	Descrição
Id	int	Identificador único e incremental
LogType	varchar(50)	Descritivo do tipo de evento
Message	varchar(500)	Mensagem associada ao evento
Detail	varchar(MAX)	Detalhe da mensagem associada ao evento
UserName	varchar(50)	Nome do utilizador que despoletou o evento
DetailData	XML	Campo de detalhe em xml
LogIdentifier	Nchar(10)	Identificação do log caso tenha id associado ao evento
ReportDate	datetime	Data que foi reportado
MessageHash	int	Código da mensagem

Existem outras tabelas de registos de eventos mais específicos como a *etl.StagClearLog* (ver Tabela 31), onde fica armazenada toda a informação de limpeza dos dados das tabelas da área temporária e a *etl.ArchiveMeasuresLog* (ver Tabela 32) que guarda a informação dos processos de agregação de indicadores.

Tabela 31 – etl.StagClearLog

Coluna	Tipo	Descrição
Id	int	Identificador único e incremental
Date	datetime	Data de arranque do processo
Notes	varchar(MAX)	Campo descritivo para armazenamento de notas específicas
State	tinyint	Estado de limpeza
DateStarted	datetime	Data início
DateFinish	datetime	Data Fim
Error	varchar(MAX)	Descrição do erro caso exista

Tabela 32 – etl.ArchiveMeasuresLog

Coluna	Tipo	Descrição
Id	int	Identificador único e incremental
Type	varchar(255)	Tipo de Agregação 'D' – dia , 'H' -hora
Date	datetime	Data de sinalização
State	tinyint	Estado da agregação 1- Iniciado ; 2 - a agregar ; 3 – Erro ; 4 – Agregado com sucesso
Notes	varchar(max)	Campo descritivo usada para enriquecer com detalhe da agregação, como qual é a tabela agregada que período de agregação está a ser usado, etc.
ArchiveIni	datetime	Data de início do período a ser agregado
ArchiveEnd	datetime	Data de fim do período a ser agregado
Table	varchar(255)	Nome da tabela agregada
DateStarted	datetime	Data de início da agregação
DateFinish	datetime	Data fim da agregação
TotalArchiveRecords	int	Número de registos agregados
TotalDeletedRecords	int	Número de registos apagados depois da agregação
PurgeData	bit	Campo de sinalização caso tenha existido limpeza dos dados agregados
Error	varchar(max)	Descrição do erro em caso de falha no processamento das agregações

3.12 Automação em *PowerShell*

A *framework* apesar de disponibilizar um conjunto de componentes que centralizam o código, regras e outros artefactos, a criação de novos processos ETL era exclusivamente uma replicação manual, através de *copy/paste* de componentes já existentes. Para garantir uma resposta mais ágil, na criação de novos processos ETL, minimizando possíveis erros em recriações manuais, foram criadas funções em *PowerShell* que automatizam a criação, possibilitam gerir pacotes de integração ou de extração, isto sem ter de executar tarefas manuais ou codificação de código.

Esta camada de automação dos processos é responsável pela criação de novos componentes de extração, replicando um pacote de SSIS genérico para um novo pacote SSIS, parametrizando apenas o nome, a conetividade à origem e a consulta a efetuar, ficando a criação de um novo extrator concluída.

A *framework* tem um conjunto de funções para serem reutilizáveis na criação de novos componentes ETL, fazendo para isso referência a algumas *DLL*[12] externas como Microsoft.SQLServer.SMO (Jorgensen, Wort, LoForte, & Knight, 2012) camada de interface de ligação com as base dados relacionais,

[12] Dynamic-link library (biblioteca de vínculo dinâmico) ou DLL, é a implementação feita pela Microsoft para o conceito de bibliotecas compartilhadas nos sistemas operacionais Microsoft Windows e OS/2

Microsoft.SQLServer.ManagedDTS (Tok, Parida, Masson, Ding, & Sivashanmugam, 2012), Microsoft.SQLServer.DTSPipelineWrap (Tok, Parida, Masson, Ding, & Sivashanmugam, 2012), Microsoft.SQLServer.PSProvider (Tok, Parida, Masson, Ding, & Sivashanmugam, 2012), estas últimas três responsáveis pelos processos de automação dos novos componentes de extração em SSIS.

As funções atuais existentes na plataforma são:

- **ExecuteSQLCommandbyConnection**
  - Parâmetros
    - $connectionString – Texto de conexão, responsável por estabelecer a ligação através de um *provider*[13];
    - $cmd – Comando SQL a ser executado;
  - Ação – Executa um comando SQL num dado servidor e base dados através de um *provider* dado por um texto de conexão;
- **ExecuteSQLCommand**
  - Parâmetros
    - $server – nome do servidor;
    - $database – nome da base de dados;
    - $cmd – comando SQL a ser executado;
  - Ação – Executa um comando SQL num dado servidor e base dados dado pelas variáveis $server e $database através do *provider* nativo do MS SQL Server ;
- **GetSQLDataTableByConnection**
  - Parâmetros
    - $connectionString – Texto de conexão, responsável por estabelecer ligação através de um *provider* declarado;
    - $query – Query SQL;

[13] Provider é um fornecedor de acesso que permite a comunicação com um sistema de gestão de base de dados

- Ação – Faz uma consulta ao servidor e base dados especificados no texto de conexão e retorna uma tabela com os dados
- Retorno – Objeto do tipo DataTable com resultado da pesquisa efetuada.

- **GetSQLDataTable**
  - Parâmetros
    - $server – nome do servidor
    - $database – nome da base de dados
    - $query – Query SQL
  - Ação – Faz uma consulta ao servidor e base dados dado pelas variáveis $server e $database através do *provider* nativo do MS SQL Server
  - Retorno – Objeto do tipo DataTable com resultado da pesquisa efetuada.
- **CreateSQLTable**
  - Parâmetros
    - $connectionOrigem – Texto de conexão, responsável por estabelecer a ligação através de um *provider;*
    - $connectionDestino – Texto de conexão, responsável por estabelecer a ligação através de um provider;
    - $query – Query SQL;
    - $newTableName – Nome da nova tabela a criar;
    - $columnIdentity – Nome da chave primária da tabela;
  - Ação – Cria uma nova tabela num servido MS SQL Server através de um resultado de um consulta.
- **CreateLoadDWView**
  - Parâmetros
    - $server – nome do servidor;
    - $database – nome da base de dados;
    - $query – Query SQL;

        - $newViewName – Nome da nova vista SQL;
    - Ação – Cria uma vista SQL que faz a transformação dos dados de *Stag* para ser resultante do esquema final da tabela em DW.
- **CopyTemplatePackage**
    - Parâmetros
        - $filePath – Caminho físico onde ficará armazenado o ficheiro de SSIS;
        - $packageTemplatePath – Caminho físico onde está armazenado o ficheiro *Template* SSIS que servirá como base aos novos pacotes de extração;
    - Ação – Copiar pacote SSIS genérico para um novo pacote SSIS.
- **CreateSPIntegrateDW**
    - Parâmetros
        - $server – Nome do servidor;
        - $database – Nome da base de dados;
        - $spName – Nome do procedimento SQL a ser criado;
        - $stagView – Nome da *view* de carregamento;
        - $dwTable – Nome da tabela destino do processo de integração;
        - $integrateType – Tipo de estratégia do integrador, "*DeleteInsert*" ou "*ByPrimaryKey*";
        - $stagPk – Nome da coluna que é chave primária em Staging;
        - $dwPk – Nome da coluna que é chave primária em DW;
        - $dateParcialIni – Nome da coluna de início que irá fazer de controlo em caso de extração parcial;
        - $dateParcialFim – Nome da coluna de fim que irá fazer de controlo em caso de extração parcial;

        - $isDimension – Variável de controlo para marcação de processamento OLAP, caso seja verdadeiro o objeto a processar será uma *Dimensão;*
        - $olapObjectName – Nome do objeto Olap a ser processado em cada integração,
        - $asserRowsVariation – Valor de controlo que multiplicado com o número de registos integrados tem que dar o número de registos extraídos, em caso de ser “0”, a opção fica desabilitada,
    - Ação – Criar um objeto de procedimentos SQL para associar a um integrador.
- **InsertNewExtractor**
    - Parâmetros
        - $server – Nome do servidor;
        - $dataBase – Nome da base dados;
        - $extractorScheduleId – Id da calendarização;
        - $extractorTypeId – Id do tipo de extração;
        - $name – Nome do extrator;
        - $description – Descrição do extrator;
        - $source – Origem do extrator, em caso de ser vazio; é assumido o por omissão configurado;
        - $location – Localização do extrator, em caso de ser vazio é assumido o por omissão configurado
        - $targetTable – Tabela de destino de extração;
        - $targetColumnDate – Nome da coluna data para extrações parciais;
        - $dateParcialExtractionIni – Data de início para a próxima extração parcial;
        - $dateParcialExtractionEnd – Data de fim para a próxima extração parcial;
        - $totalExtraction – Variável booleana que valida se é para executar a extração com filtros ou se é para extrair todos os registos da origem;
        - $extractorOrder – Ordem de extração;

        - $active – Ativar ou desativar o extrator;
    - Ação – Cria um novo registo na tabela *Extractor,* tabela de catálogo dos extratores.
- **InsertNewIntegrator**
    - Parâmetros
        - $server – Nome do servidor;
        - $dataBase – Nome da base dados;
        - $extractorId – Id do extrator associado à integração,
        - $integratorTypeId – Id do Tipo de integração;
        - $name – Nome do integrador;
        - $description – Descrição do integrador;
        - $source – Origem do integrador, em caso de ser vazio é assumido por omissão o configurado no ficheiro XML;
        - $location – Localização do integrador, em caso de ser vazio é assumido por omissão o configurado no ficheiro XML;
        - $targetTable – Tabela de destino da integração;
        - $active – Ativar ou desativar o integrador;
    - Ação – Cria um novo registo na tabela *Integrator,* tabela de catálogo dos integradores.

- **CreateNewPackage**
    - Parâmetros
        - $fileName – Nome do ficheiro SSIS;
        - $packagesPath – Caminho físico onde ficará armazenado o novo pacote SSIS;
        - $query – Query SQL que fará consulta à origem a extrair dados;
        - $tableDestiny – Tabela de destino na área de Staging;
        - $dataSourceConnectionName – Nome da conetividade para relacionar com o ficheiro de configuração do SSIS;

- $dataSourceConnectionString – Conetividade para a origem de dados;
- $serverSource – Servidor da Origem;
- $dataBaseSource – Base dados de Origem;
- $packageTemplatePath – Caminho físico para o pacote SSIS Template;

- Ação – Cria um novo pacote de extração de dados do tipo SSIS.

3.13 Interface Web

Sem nunca perder o âmbito da *framework*, e sempre segmentando a aplicação para um público-alvo especializado, como ferramenta de auxílio nas tarefas de construção e manutenção de repositórios centralizados denominados muitas das vezes por *Data Warehouse*, é disponibilizado uma interface web que através de invocações dos *scripts Powershell* cria novos processos ETL. Para além da criação de novos pacotes de SSIS, novas tabelas, vistas e procedimentos em SQL, também possibilita a manutenção e configuração dos processos de extração e integração. A calendarização, monitorização e automação de processo com portabilidade, foram objetivos atingidos com esta interface (ver Figura 23).

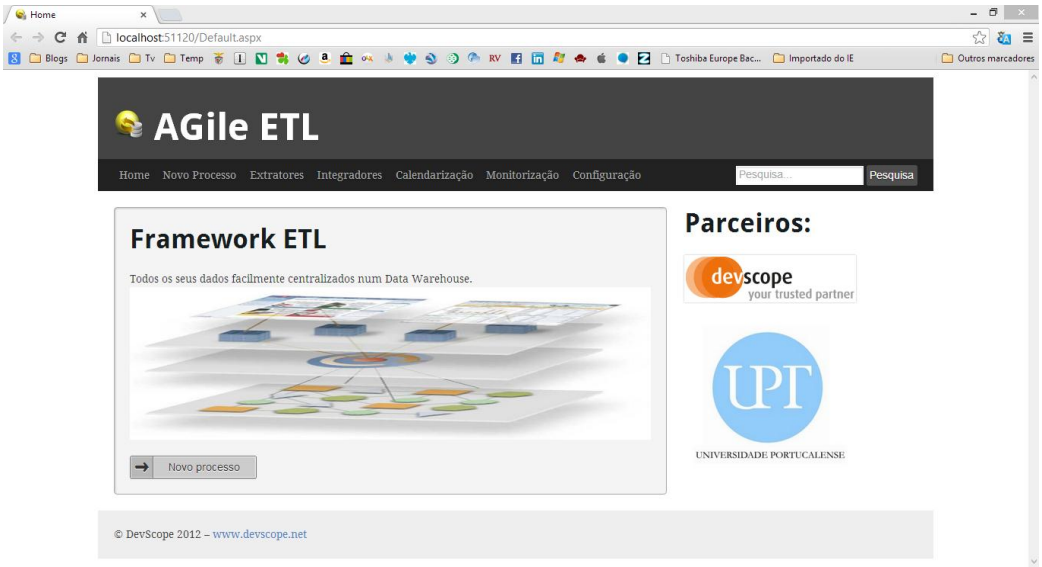


Figura 23 – Página Inicial

3.13.1 Novo Processo

A Criação de um novo processo ETL consiste inicialmente no preenchimento de um formulário dividido por três etapas.

No painel "*Origem Dados*" (ver Figura 24), deve ser preenchida a caixa de texto "*Conetividade*" com o texto de conexão que servirá para efetuar a ligação com a base de dados de origem. Na caixa de texto "*Query Extração*" deve-se colocar a *query* de extração, que servirá para extrair a informação do repositório de origem. A inserção do "*Nome*" na caixa de texto, servirá para o pacote de extração, e para facilitar o utilizador preenche automaticamente os restantes itens do formulário com o valor, ou seja se o pacote tiver preenchido com "*Serviço"* como mostra a Figura 24, então o nome da tabela em *Staging* fica preenchido como "*Serviço"* e o nome da *view* de integração fica "*vwLoadDimServicos"*, enquanto a *query* de transformação inicialmente será "*select * from Serviço"*. Esta funcionalidade serve como propósito inicial para agilizar a criação, mas não impede que o administrador faça, numa fase posterior, as alterações que entenda mais adequadas.

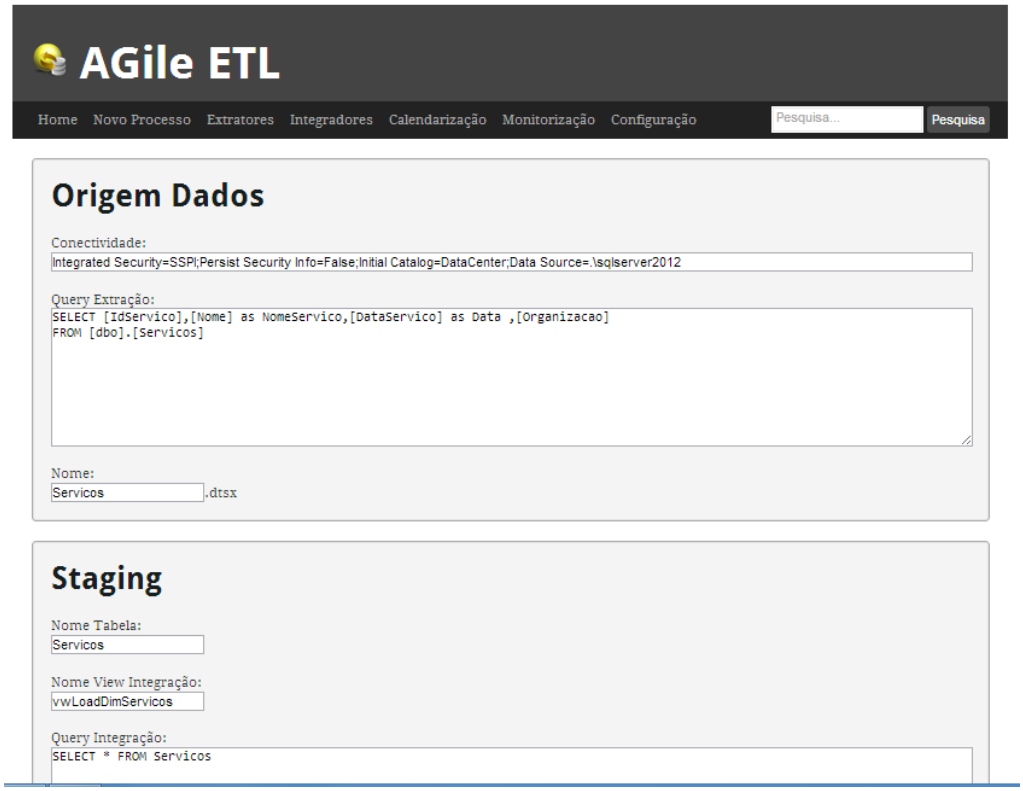


Figura 24 – Formulário de novo Processo ETL Parte 1

Os campos da seção de *Data Warehouse* (ver Figura 25) também serão iniciados, com o valor escolhido no campo "*Nome*" da seção "*Origem de Dados*", para agilizar o processo de criação, sendo que estes têm um prefixo associado ao tipo de objeto, mas pode ser alterado antes de gravar.

Os elementos a preencher são o "*Nome Tabela*", nome para a nova tabela no *Data Warehouse* (ver Figura 25), uma caixa de verificação, designada por "*Indicador*", onde se seleciona caso o novo processo seja para ser contemplado no "*Modelo de Indicadores*" (ver seção 3.3.2), sendo selecionado, a caixa de texto "*Nome Tabela*" fica em modo de leitura. O nome do procedimento de integração é outro dos componentes a inserir "*Nome SP Integração*", e também o valor de controlo de registos entre a integração e a extração "*Controlo Registos Extraídos*". Na seção *Data Warehouse,* é também contemplado o processamento OLAP, para isso deve ser preenchido o nome do objeto OLAP na caixa de texto "*Nome Objeto OLAP*", e selecionar a caixa de verificação "*Dimensão*" para novos objetos do tipo dimensão de análise. Finalmente a escolha da estratégia de integração dos dados é realizada através de botões de seleção, onde se seleciona se os registos serão

apagados e criados de novo em "*Apaga e Cria registos*" ou atualizados se serão atualizados pela chave primária "*Atualiza por chave primária*"; quando se seleciona "*Atualiza por chave primária*" tem que se preencher o nome da chave primária em "*Campo Lookup*".

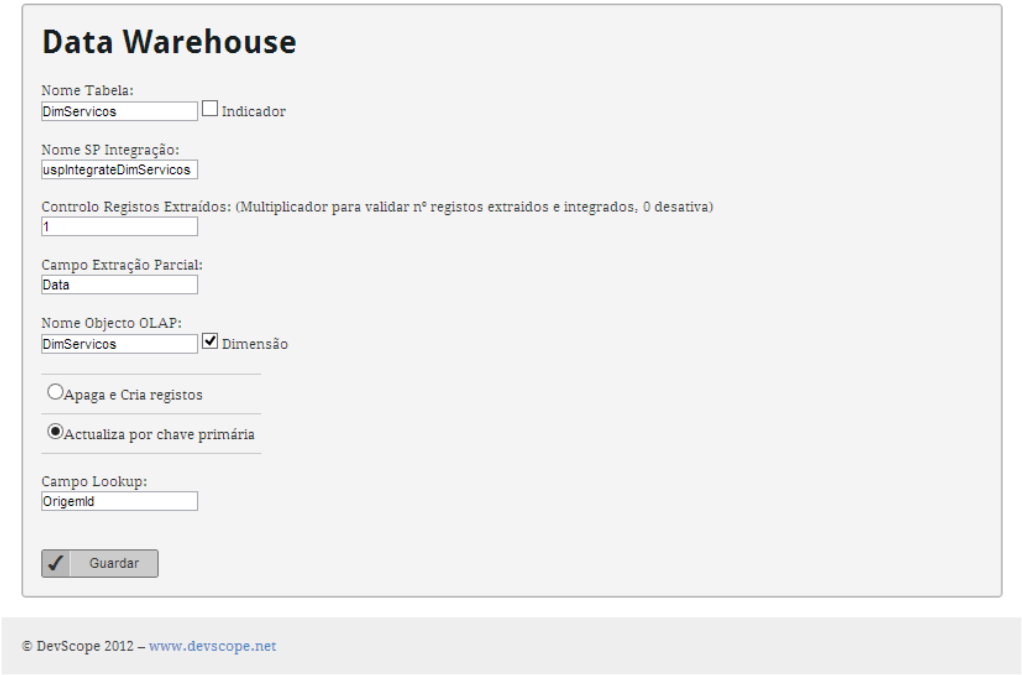


Figura 25 – Formulário de novo Processo ETL Parte 2

Ao gravar, a aplicação cria todos os artefactos que garantem o funcionamento quer da extração dos dados quer da integração dos mesmos já consolidados no repositório final.

Na fase seguinte, é necessário preencher o agendamento do processo criado. Nesta interface (ver Figura 26) configuram-se os dados de extratores e integradores (ver Figura 27) para serem inseridos nas tabelas *etl.Extractor* (ver Tabela 17) e *etl.Integrator* (ver Tabela 24) respetivamente. A configuração passa por selecionar na caixa de seleção "*Calendário de Extração*", o período ao qual se pretende que o processo de ETL seja executado, por preencher o nome do extrator em "*Nome Extrator*", e a sua descrição em "*Descrição*", existe uma caixa de seleção para escolher o tipo de extração "*Tipo de Extrator*", que pode ser entre pacotes de SSIS "*Packages SSIS*" ou procedimentos em SQL "*SQL Store Procedure*". A caixa de texto "*Nome Objeto Extração"* será preenchida automaticamente pelo nome inserido no campo "*Nome*" do painel

de “*Origem de Dados*” da Figura 24. A “*Tabela Destino*”, por sua vez será preenchida pelo campo “*Nome Tabela*” da seção “*Staging*” da Figura 24. No campo de texto “*Coluna Data Destino*” deve ser inserido o nome da coluna do tipo data, ao qual será responsável pela extração parcial dos dados. Os campos “*Extrair a partir de*”, “*Ordem Extração*” e “*Extração Total*”, são atributos de controlo para extrair os dados na origem, o “*Extrair a partir de”,* faz com que a próxima extração seja parcial e que a data de extração seja a inserida na coluna, a “*Ordem Extração*” é para garantir que todos os extratores executados de forma sequencial têm um controlo pela forma e ordem de inserção, quer em *Staging,* quer no *Data Warehouse.* O último atributo é o “*Estado"*, que representa se o extrator está ativo ou desativo.

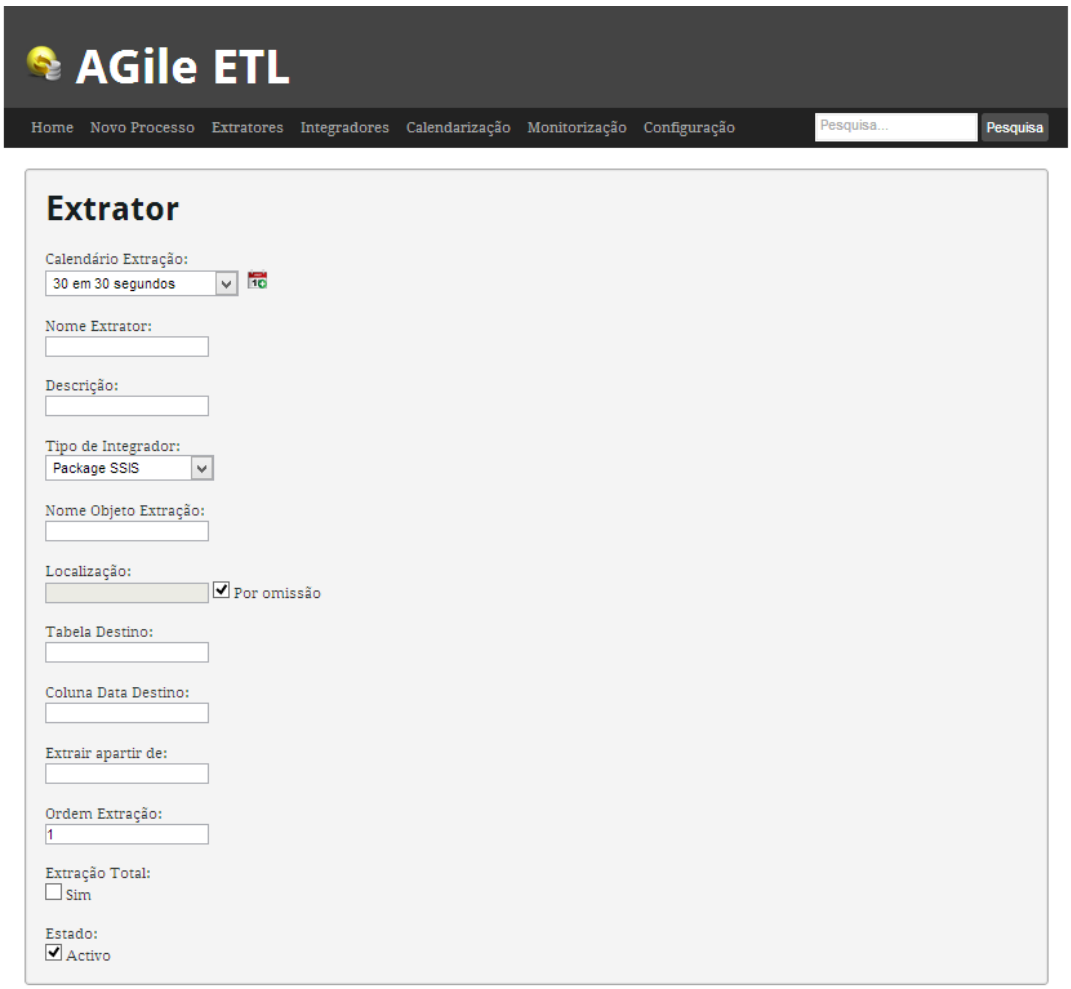


Figura 26 – Formulário de novo Extrator

A seção “*Integrador*” (ver Figura 27) tem o campo “*Nome Integrador*”, este atributo é preenchido pelo valor inserido no campo “*Nome*” da Figura 24, e o campo “*Descrição*”, que é um atributo de texto livre, existe à semelhança com a seção de “*Extrator*” uma caixa de seleção onde existe as duas opções de integração da *framework,* pacotes de SSIS “*Packages SSIS*” e procedimentos em SQL “*SQL Store Procedure*”. Para finalizar o componente de integração existe um campo para designar a tabela de destino onde os dados serão armazenados no *Data Warehouse* o campo “*Tabela Destino*”, e uma caixa de verificação para ativar ou desativar o integrador.

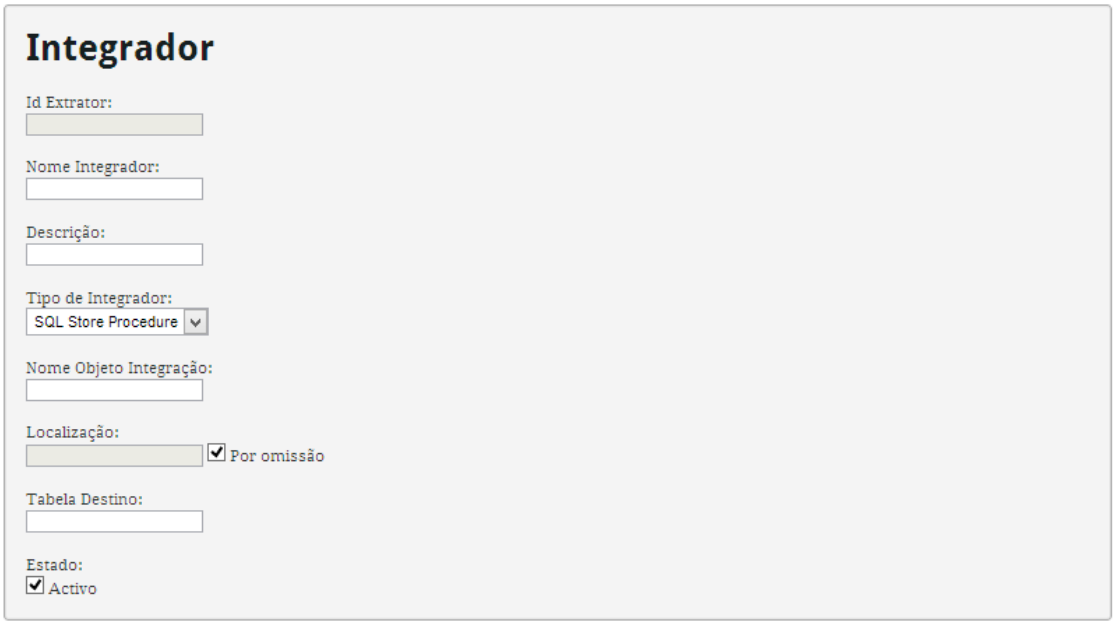


Figura 27 – Formulário de novo Integrador

A seção *Scope,* permite nas extrações restringir as consultas à origem de dados (ver seção 3.6.2), preenchendo o “*Nome*” do *Scope* e os valores que o caracterizam, como “*Operador Lógico*” (por exemplo “>” “<”), “*Expressão de Extração*” e “*Expressão de Integração”*, estes dois campos são preenchidos pelo nome da coluna ao qual se executará a filtragem, o “*Operador de Agregação*” que tem como opção apenas os valores “*And*” ou “*Or*”, o “*Valor Filtragem*” que é o valor ao qual será aplicado a restrição e para finalizar a caixa de verificação “*Estado*” que ativa ou desativa o *Scope* (ver Figura 28).

Para finalizar a criação do processo ETL basta "*Gravar*" e todos os artefactos serão criados para as extrações e integrações serem executadas (ver Figura 28).

Figura 28 – Formulário de criação de Scope

A execução da primeira extração pode não ser imediata, uma vez que depende sempre da configuração do calendário, o extrator pode ter o período definido para executar, mas a hora atual não está compreendida nessa configuração, assim, os processos ficaram à espera do período escolhido para serem executados.

Todas as calendarizações podem ser alteradas e criadas novas no separador de "*Calendarização"* da aplicação (ver Figura 29), neste separador é possível manipular todos os períodos que sejam pertinentes à execução dos processos ETL.

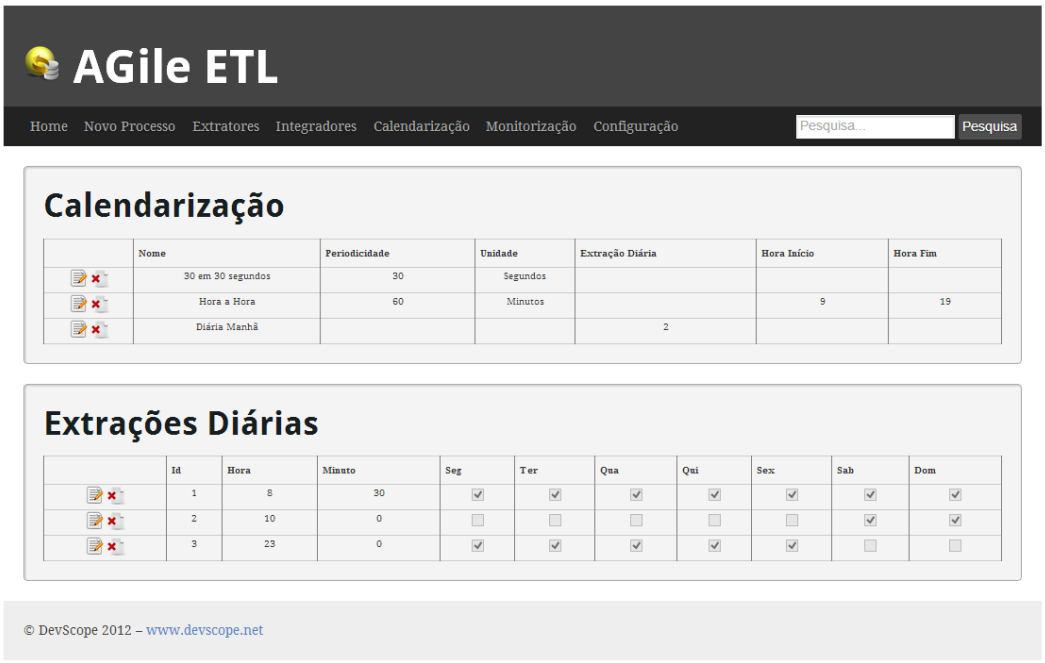


Figura 29 – Calendarização

Existem páginas para a manipulação de extratores (ver Figura 30) e integradores (ver Figura 31). Funcionalidades como alterar de ativo para inativo ou manipular extrações totais, ou executar extrações com datas parciais são componentes garantidos pela aplicação quer no separador de *Extratores* quer no separador de *Integradores*, apenas basta selecionar os botões das tabelas de edição e alterar os dados pretendidos e voltar a clicar no botão de "*Gravar*".

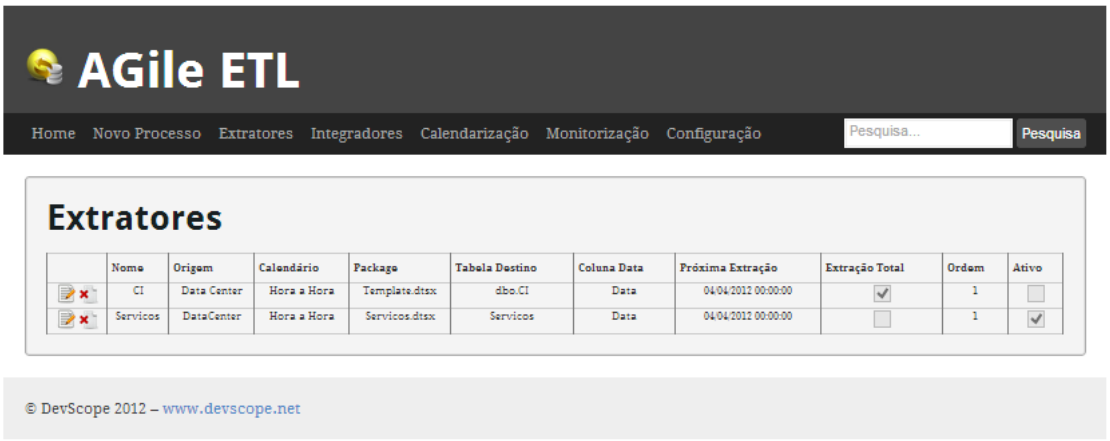


Figura 30 – Extratores

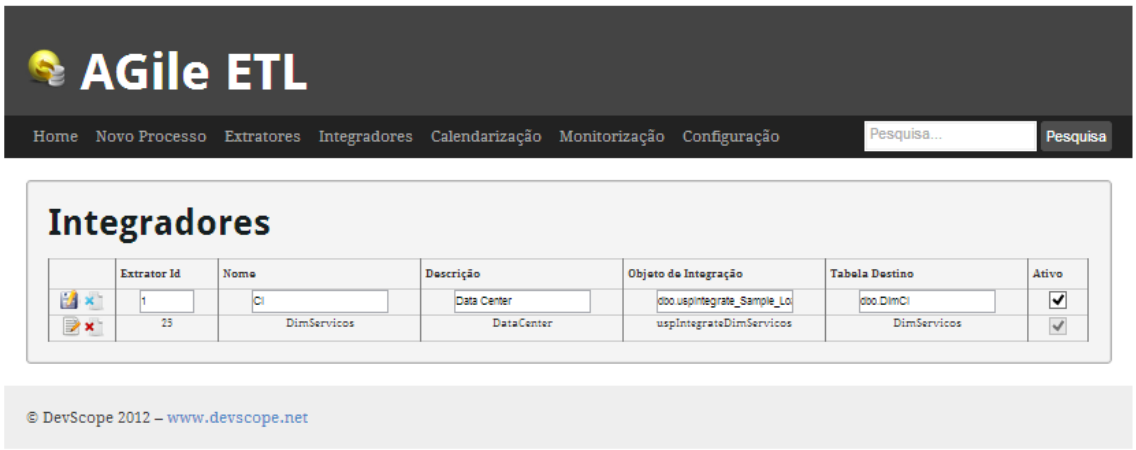


Figura 31 – Integradores

Para dar uma resposta mais adequada, foi construída uma página de controlo e monitorização de eventos de extrações e integrações (ver Figura 32), de forma a que os administradores possam controlar todos os processos que foram executados e que estão a ser executados no momento. Para além disso, é nesta página que se consegue verificar e controlar as extrações ou integrações que resultaram com erros e a descrição dos erros, como se pode observar na Figura 32. É nesta interface que também se controla os tempos de execução e a ordem de grandeza dos processos ETL.

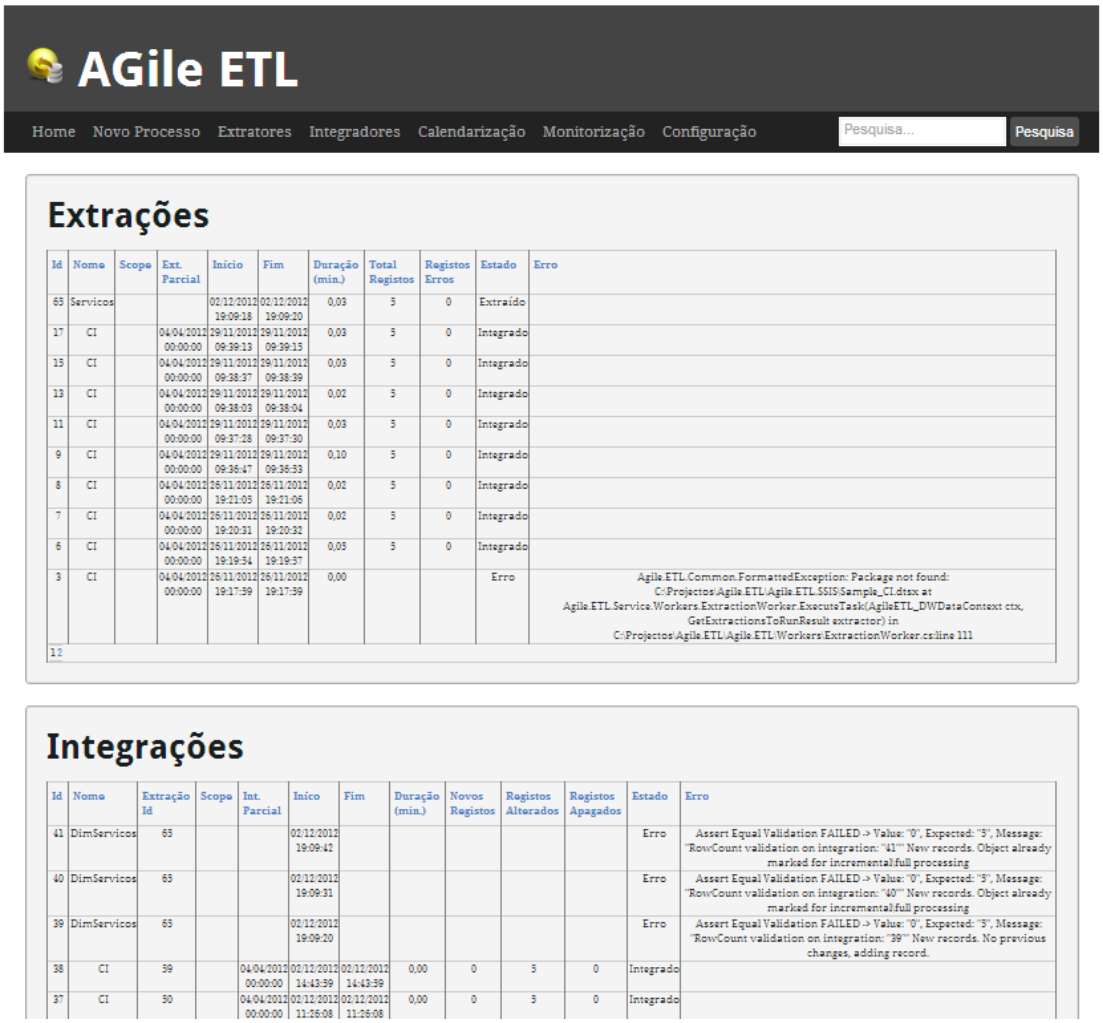


Figura 32 – Extrações e Integrações

4. Resultados Obtidos

Sabendo que a agilidade de construção ou a facilidade de manutenção dos processos ETL têm visibilidade mais acentuada, é necessário demonstrar as mais-valias também no desempenho de execução. Como tal realizou-se uma análise de desempenho da execução das tarefas de extração e integração. Nesse sentido, realizou-se um levantamento de um período de amostra compreendido entre 1 de Novembro de 2012 a 1 de Dezembro de 2012, e efetuou-se vinte e nove extratores e respetivos integradores, cada um com a sua calendarização.

Os resultados obtidos, foram analisados comparativamente a outras tecnologias de ETL, com base num estudo de *benchmark*[14] (Infosphere, 2011).

As ferramentas ETL utilizadas na comparação de desempenho são:

- TOS – *Talend Open Integration Solution;*
- PDI – *Pentaho Data Integration;*
- DataStage – *IBM InfoSphere DataStage;*
- Informatica – *Informatica PowerCenter.*

4.1 Extrações

Os processos de extração analisados (ver Tabela 33) são todos efetuado através da *framework* ETL utilizando pacotes de SSIS. Estes pacotes têm como objetivo a transição dos registos desde as origens até à área temporária (*Staging*).

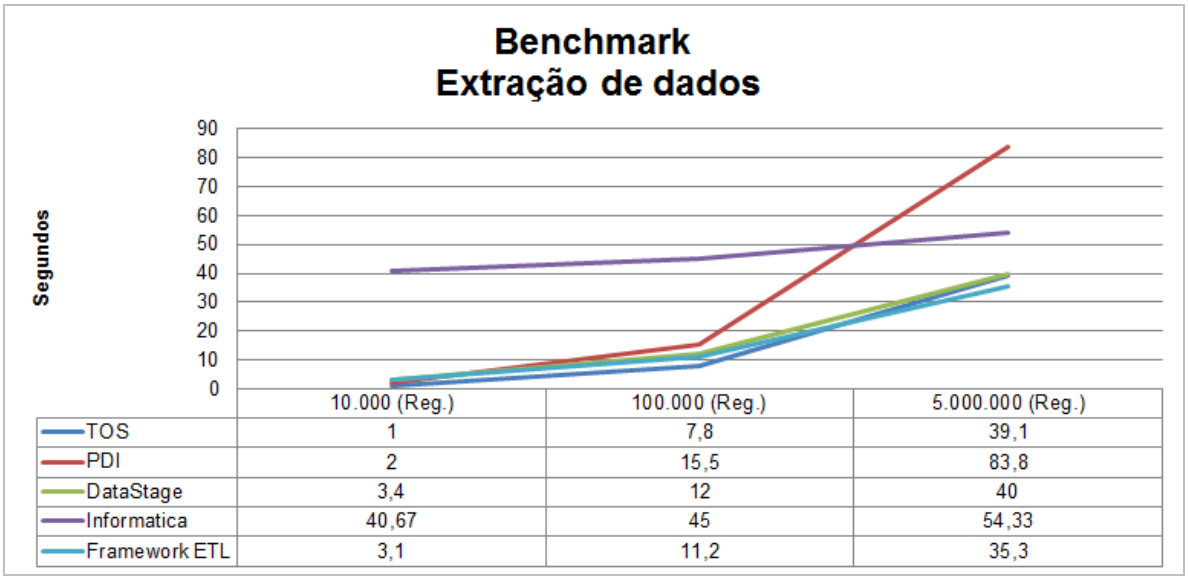
Existem várias origens distintas na análise, diversos servidores que utilizam base de dados em MS SQL Server 2005, MS SQL Server 2008, MS SQL Server 2008 R2, Oracle 10 e Oracle 11.

[14] É o ato de executar um conjunto de operações, com a finalidade de avaliar a performance relativa a um objeto, normalmente é executado uma série de testes para serem verificados os seus resultados.

Tabela 33 – Resultados Extrações

Extrator	Duração (seg.)	Número de registos extraídos
Backups	41,76	5.000.000
CI	8,97	100.000
Disponibilidade	10,37	100.000
Disponibilidade Aplicacional Protocolo	2,75	10.000
Disponibilidade de Rede	2,57	10.000
Alteração	4,29	10.000
Chamada	3,29	10.000
Serviço	2,03	10.000
Grupos Trabalhos	3,16	10.000
Incidentes	2,45	10.000
Instância VLAN	11,37	100.000
Interação	3,77	10.000
Interação Chamadas	3,26	10.000
Intervenções	3,27	10.000
Log Acessos	3,06	10.000
Métricas CPU	11,13	100.000
Métricas Disco	27,69	5.000.000
Métricas Disco Físico	34,78	5.000.000
Métricas Disco Info	34,75	5.000.000
Métricas File System	11,13	100.000
Métricas Memória	12,81	100.000
Métricas SWAP	11,25	100.000
Métricas NetWorking	37,27	5.000.000
Métricas Processador	12,94	100.000
Organização	3,42	10.0000
Serviços Instaldo por máquina	11,13	100.000

Como pode ser verificado na Figura 33, a *framework* ETL, na extração de dados através de pacotes de SSIS, teve resultados bastantes satisfatórios, ficando em segundo lugar nas extrações superiores a cem mil registos e em terceiro lugar nas extrações de dez mil registos.



	10.000 (Reg.)	100.000 (Reg.)	5.000.000 (Reg.)
TOS	1	7,8	39,1
PDI	2	15,5	83,8
DataStage	3,4	12	40
Informatica	40,67	45	54,33
Framework ETL	3,1	11,2	35,3

Figura 33 – Benchmark de Extração

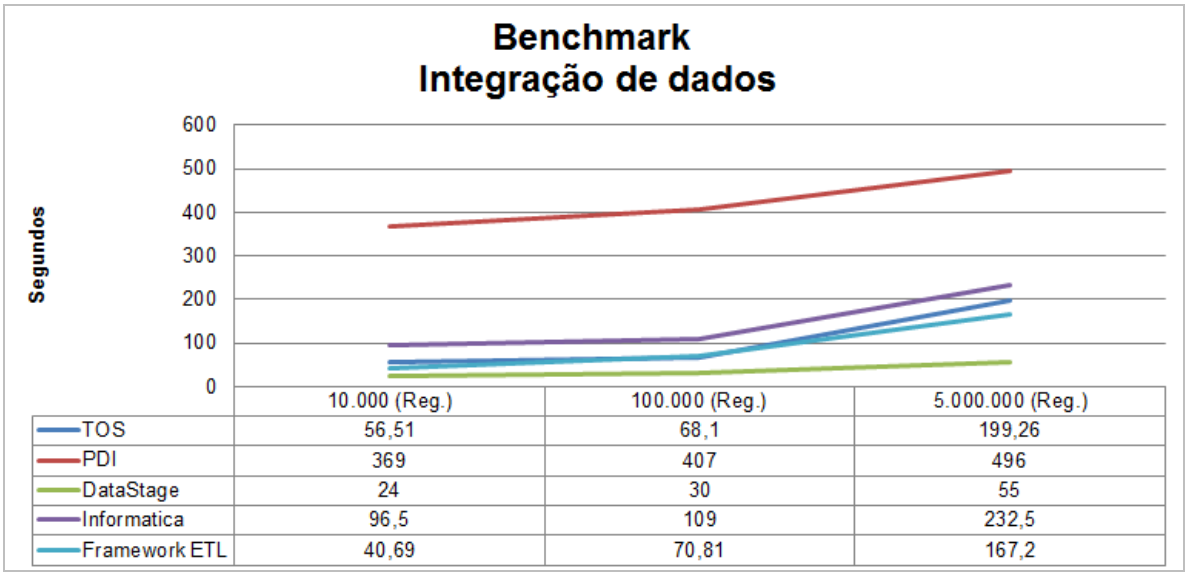
4.2 Integrações

As integrações analisadas (ver Tabela 34) são todas efetuadas através da *framework* ETL, utilizando procedimentos em SQL. Estes procedimentos têm como objetivo consolidar a informação no *Data Warehouse*, fazendo para isso transformações aos dados armazenados em *Staging*, que visam responder ao esquema desenhado e arquitetado no *Data Warehouse*. Aos processos de integração foi efetuado um ajustamento de desempenho com a criação de índices, criação de partições em tabelas, garantindo melhores resultados na transformação de dados (ver Figura 34).

Tabela 34 – Resultados das Integrações

Integrador	Duração (seg.)	Número de registos integrados
Backups	119,98	5.000.000
CI	63,9	100.000
Disponibilidade	74,57	100.000
Disponibilidade Aplicacional Protocolo	25,26	10.000
Disponibilidade de Rede	39,09	10.000
Alteração	46,13	10.000
Chamada	42,03	10.000
Serviço	42,13	10.000
Grupos Trabalhos	22,45	10.000
Incidentes	30,48	10.000
Instância VLAN	71,74	100.000
Interação	20,53	10.000
Interação Chamadas	20,53	10.000
Intervenções	85,52	10.000
Log Acessos	61,23	10.000
Métricas CPU	63,03	100.000
Métricas Disco	240,58	5.000.000
Métricas Disco Físico	131,55	5.000.000
Métricas Disco Info	178,26	5.000.000
Métricas File System	86,23	100.000
Métricas Memória	86,23	100.000
Métricas SWAP	86,23	100.000
Métricas NetWorking	165,61	5.000.000
Métricas Processador	66,77	100.000
Organização	53,23	10.0000
Serviços Instaldo por máquina	68,47	100.000

Na Figura 34, pode-se verificar que *framework* ETL, na integração de dados através de procedimentos SQL, teve resultados bastantes positivos, ficando sempre nos três primeiros lugares em todos os testes realizados.



	10.000 (Reg.)	100.000 (Reg.)	5.000.000 (Reg.)
TOS	56,51	68,1	199,26
PDI	369	407	496
DataStage	24	30	55
Informatica	96,5	109	232,5
Framework ETL	40,69	70,81	167,2

Figura 34 – Benchmark de Integração

5. Conclusão

Construir uma *framework* que disponibiliza-se funções com todos os mecanismos de criação para extratores e integradores de um *Data Warehouse*, foi importante, no entanto, desejou-se chegar a um patamar superior, disponibilizando uma camada de interface que apenas através de um *wizard*, é possível criar novos processos ETL.

O principal objetivo do trabalho era conseguir propor uma solução para agilizar o processo de carregamento de dados num repositório central, *Data Warehouse*. Para tal foi criado um protótipo, implementado com vários componentes distintos, tais como tabelas SQL, um serviço Windows, funções em *Powershell* para automação de processos, interface *web* para responder mais agilmente ao pedidos de alteração e monitorização de extrações ou integrações de dados e pacotes SSIS.

O protótipo implementado centraliza a criação e automação dos processos ETL, viabilizando a construção de repositórios centrais de dados a empresas com menor capacidade financeira.

Com soluções como esta o paradigma de “Business Intelligence” tende transforma-se no paradigma “Data Intelligence” (O'Reilly, 2012). Isto porque com soluções como a que foi apresentada, a capacidade de adquirir ferramentas de análise não estará segmentada apenas para as grandes empresas e o caminho tenderá para que o consumidor final consiga, mesmo para fins privados, analisar e retirar padrões sobre os seus dados para inferir e fazer opções mais assertivas (Minelli, Chambers, & Dhiraj, 2012).

5.1 Trabalho Futuro

No protótipo implementado foi possível detetar que alguns ajustes facilitariam ainda mais os processos de migração de informação. A ideia seria não fazer apenas a migração quase de tabela para tabela desde a origem até ao *Data Warehouse*, mas conceber um novo componente à semelhança do

"*Data Source View*" do *SQL Server Analises Services*, onde se define toda a estrutura pretendida, através de "*drag-and-drop"* dos vários objetos a importar, de origens distintas, e o modelo seria automaticamente ajustado respondendo ao esquema desenhado para o *Data Warehouse*.

Outro objetivo futuro é integrar a *framework,* com uma interface móvel, que possibilite monitorizar e criar novos componentes, através de um *smartphone, tablet* seja ele *IOS*[15]*, Android*[16] *ou Windows Phone*[17].

A longo prazo existe a intenção de adicionar uma nova arquitetura à *framework* ETL*, Event-driven architecture*[18] utiliza um sistema de subscrição de eventos, que ao contrário do sistema atual, não é uma arquitetura de pedido e resposta entre servidores, sejam eles de origem, *Staging* ou *Data Warehouse.* A arquitetura *Event-driven*, trabalha através de subscrição de eventos, que possibilita na sua subscrição, a realização de ações em resposta aos eventos criados. A mais valida para a *framework*, seria deixar de calendarizar os processos de extração, fazendo apenas subscrições nos servidores de origem, que nas entradas de novos registos ou alterações nos dados, seriam automaticamente encaminhados para a área de *Staging* e em seguida transformados e inseridos no *Data Warehouse*.

[15] Sistema Operativo, móvel da *Apple Inc*. desenvolvido originalmente para o *iPhone*, também é usado em *iPod Touch, iPad* e *Apple TV*.
[16] Sistema Operativo baseado no *Linux*, para dispositivos móveis, utilizado em dispositivos como *Google Nexus One, Nexus S, Galaxy Nexus*,
[17] Sistema Operativo móvel, desenvolvido pela *Microsoft*, sucessor da plataforma *Windows Mobile.*
[18] Event-Driven Architecture é uma arquitetura de software que promove a produção, a deteção, o consumo de, e a reação a eventos.

Bibliografia

ECKERSON, W. (2010). *Smart Companies in the 21st Century.* Seattle: The Data Warehousing Institute.

Fellows, B. (Novembro de 2011). *PowerShell manipulation of SSIS packages*. Obtido de Bill Fellows Blog: http://billfellows.blogspot.pt/2011/11/powershell-manipulation-of-ssis.html

Gaea Consulting. (Julho de 2010). Obtido de Reutilização de Código: http://www.gaea.com.br/br/artigos/8-software/8-reutilizacao-de-codigo

Gu, S. (06 de Julho de 2007). *LINQ to SQL*. Obtido de ScottGu's Blog: http://weblogs.asp.net/scottgu/archive/2007/07/16/linq-to-sql-part-5-binding-ui-using-the-asp-linqdatasource-control.aspx

H. N. Laursen, G., & Thorlund, J. (2010). *Business Analytics for Managers.* New Jersey: Jonh Wiley & Sons, Inc.

Haselden, K. (2009). *Microsoft SQL Server 2008 Integration Services.* Indianapolis: Sams.

Holmes, L. (2010). *Windows PowerShell Cookbook.* Sebastopol: O'Reilly Media.

Infosphere. (2011). *ETL Benchmark Favours Datastage and Talend*. Obtido de ETL Benchmark: http://it.com/blogs/infosphere/etl-benchmark-favours-datastage-and-talend-28695

Jones, D. (2011). *Learn Windows PowerShell in a Month of Lunches.* Greenwich: Manning Publications.

Jorgensen, A., Wort, S., LoForte, R., & Knight, B. (2012). *Professional Microsoft SQL Server 2012 Administration.* Indianapolis: Wiley.

Kimball, R., & Caserta, J. (2004). *The Data Warehouse ETL Toolkit.* Indianapolis: Wiley Publishing, Inc.

Knight, B., Veerman, E., Dickinson, G., Herbold, D., & Hinson, D. (2008). *Professional SQL Server® 2008 Integration Services.* Indianapolis: Wiley Publishing, Inc.

MacDonald, M. (2010). *Beginning ASP.NET 4.0 in C#.* New York: APress.

MacDonald, M., Freeman, A., & Szpuszta, M. (2010). *Pro ASP.NET 4.0 In C#.* New York: APress.

Microsoft. (10 de Outubro de 2010). *Introduction to Windows Service Applications*. Obtido de MSDN Microsoft: http://msdn.microsoft.com/en-us/library/d56de412(v=vs.80).aspx

Microsoft. (Junho de 2011). Analysis Services Operations Guide. *SQL Server White Paper*, p. 108.

Microsoft. (2012). *Compare Competitors*. Obtido de Microsoft Sql Server: http://www.microsoft.com/sqlserver/en/us/product-info/competitor-compare.aspx#Oracle

Minelli, M., Chambers, M., & Dhiraj, A. (2012). *Big Data, Big Analytics.* New Jersey: John Wiley & Sons, Inc.

Mistry, R., & Misner, S. (2012). *Introducing Microsoft SQL Server 2012.* Redmond, Washington: Microsoft Press.

Morin, R. C. (2000). *Programming Windows Services.* Indianapolis: Wiley Publishing, Inc.

Ndlovu, S. W. (02 de Junho de 2011). *Programmatically Create Data Flow Task in SSIS Package Using C#*. Obtido de select Sifiso: http://www.selectsifiso.net/?p=288

Netcraft. (Dezembro de 2012). Obtido de Web Server Survey: http://news.netcraft.com/archives/2012/12/04/december-2012-web-server-survey.html#more-7480

Nooijer, H. d. (17 de Janeiro de 2011). *SSIS : Adding a Derived Column to a SSIS package with C#*. Obtido de BI Future Blog:

http://bifuture.blogspot.pt/2011/01/ssis-adding-derived-column-to-ssis.html

O'Reilly. (2012). *Big Data Now.* Köln: O'Reilly Media, Inc.

Parker, R. (2008). *Performance Management Scorecards and Dashboards for IT Operations Data.* Microsoft.

Rotem-Gal-Oz, A. (2012). *SOA Patterns.* New York: Manning.

RSSBus. (2012). *PowerShell SSIS Components*. Retrieved from rssbus: http://www.rssbus.com/ssis/powershell/

Sheldon, R. (01 de Março de 2012). *Working with Variables in SQL Server Integration Services*. Obtido de Simple-talk: http://www.simple-talk.com/sql/ssis/working-with-variables-in-sql-server-integration-services/

Sojo, E. (03 de Janeiro de 2012). *Creando paquetes de SSIS con .NET*. Obtido de Blog de Eduardo Sojo: http://eduardosojo.com/2012/01/03/creando-paquetes-ssis-con-net-creando-data-flow-task-y-elementos-internos/

T. Moss, L., & Atre, S. (2012). *Business Intelligence Roadmap.* Boston: Addison-Wesley.

Thomsen, E., Spofford, G., & Chase, D. (1999). *Microsoft OLAP Solutions.* Indianapolis: Wiley Publishing, Inc.

Tok, W.-H., Parida, R., Masson, M., Ding, X., & Sivashanmugam, K. (2012). *Microsoft SQL Server 2012 Integration Services.* Sebastopol: O'Reilly Media, Inc.

Turban, E., Sharda, R., Delen, D., & King, D. (2010). *Business Intelligence (2nd Edition).* London: Prentice Hall.

Van Bon, J., & de Jong, A. (2007). *IT Service Management: An Introduction Based on ISO 20000 & ITIL V3.* Amersfoort: Van Haren Publishing.

VelvetBlues. (2010). *Website Performance*. Obtido de Web Development Blog: http://www.velvetblues.com/web-development-blog/php-vs-asp-which-is-best-for-your-application/

Webb, C., Ferrari, A., & Russo, M. (2009). *Expert Cube Development with Microsoft SQL Server 2008 Analysis Services.* Birmingham: Packt Publishing Ltd.

Wikipedia. (s.d.). *Wikipedia Free Encyclopedia*. Obtido de Wikipedia Free Encyclopedia: http://en.wikipedia.org/wiki/Main_Page