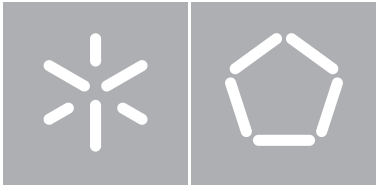
Universidade do Minho
Escola de Engenharia

Hugo Miguel Teixeira Lopes Guimarães

Geração de Esqueletos para Sistemas de ETL a partir de Redes de Petri Coloridas

Outubro de 2014



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Hugo Miguel Teixeira Lopes Guimarães

Geração de Esqueletos para Sistemas de ETL a partir de Redes de Petri Coloridas

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de
Professor Doutor Orlando Manuel de Oliveira Belo
Professora Doutor João Miguel Fernandes

Outubro de 2014

Anexo 3

DECLARAÇÃO

Nome

Hugo Miguel Teixeira Lopes Guimarães

Endereço electrónico: hmtlguimaraes@gmail.com Telefone: 934 672 986 / _____

Número do Bilhete de Identidade: 13744185

Título dissertação □/tese □

Geração de Esqueletos ETL a partir de Redes de Petri Coloridas

Orientador(es):

Orlando Manuel de Oliveira Belo e João Miguel Fernandes

Ano de conclusão: 2014

Designação do Mestrado ou do Ramo de Conhecimento do Doutoramento:

Mestrado em Engenharia Informática

Nos exemplares das teses de doutoramento ou de mestrado ou de outros trabalhos entregues para prestação de provas públicas nas universidades ou outros estabelecimentos de ensino, e dos quais é obrigatoriamente enviado um exemplar para depósito legal na Biblioteca Nacional e, pelo menos outro para a biblioteca da universidade respectiva, deve constar uma das seguintes declarações:

1. É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA TESE/TRABALHO APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;

2. É AUTORIZADA A REPRODUÇÃO PARCIAL DESTA TESE/TRABALHO (indicar, caso tal seja necessário, nº máximo de páginas, ilustrações, gráficos, etc.), APENAS PARA EFEITOS DE INVESTIGAÇÃO, , MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;

3. DE ACORDO COM A LEGISLAÇÃO EM VIGOR, NÃO É PERMITIDA A REPRODUÇÃO DE QUALQUER PARTE DESTA TESE/TRABALHO

Universidade do Minho, 31/10/2014

Assinatura: Hugo Miguel Teixeira Lopes Guimarães

Geração de Esqueletos para Sistemas de ETL a partir de Redes de Petri Coloridas

Hugo Miguel Teixeira Lopes Guimarães

Dissertação apresentada à Universidade do Minho para obtenção do grau de Mestre em Engenharia Informática, elaborada sob orientação do Professor Doutor Orlando Manuel de Oliveira Belo e do Professor Doutor João Miguel Fernandes.

2014

Agradecimentos

A todos aqueles que me acompanharam neste percurso de desenvolvimento e escrita da dissertação. Aos meus orientadores e professores, Orlando Belo e João Miguel Fernandes. Ao pessoal do laboratório que me acompanhou neste processo e à minha família que me proporcionou o percurso académico que me encontro a fazer.

Resumo

Geração de Esqueletos para Sistemas de ETL a partir de Redes de Petri Coloridas

As Redes de Petri Coloridas são uma linguagem gráfica com uma semântica bem definida, que permite o desenho, especificação, simulação e validação de sistemas, cujos processos a modelar exijam características específicas de comunicação, concorrência e sincronização entre si. A nível aplicacional, as Redes de Petri Coloridas surgem em áreas muito diferentes, tais como a especificação de protocolos de comunicação, sistemas de controlo, sistemas de hardware ou de sistemas de software. Devido às suas características as Redes de Petri Coloridas foram adotadas, também, na modelação de sistemas de ETL (Extract-Transformation-Load). Meta-tarefas como *Change Data Capture* ou *Surrogate Key Pipelining*, frequentemente encontradas em sistemas de ETL convencionais, foram modeladas e validadas através do uso de redes de Petri Coloridas. Tal sustenta, de forma bastante efetiva, o objetivo principal deste trabalho de dissertação: desenvolver e implementar um sistema para a geração de esqueletos para sistemas de ETL a partir da correspondente Rede de Petri Colorida.

Abstract

Generation of Skeletons for ETL System based on Coloured Petri Nets

Coloured Petri Nets are a graphical language with a well-formed semantic, that allows the design, specification, simulation, and validation of systems, which specific characteristics such as, communication, concurrency and synchronization have a main role in the processes to model. At application level, Coloured Petri Nets are used in a wide variety of scientific areas, such as communication protocol, control systems, hardware systems or software systems. Due their characteristics Coloured Petri Nets were also adopted in modeling ETL (Extract-Transformation-Load) systems. Meta-tasks like Change Data Capture or Surrogate Key Pipelining, that are frequently founded in conventional ETL system, were modeling and validated using Coloured Petri Nets. All this support, quite effectively, the main propose of this dissertation work: develop and implement a system to generating skeletons to ETL systems from the corresponding Coloured Petri Nets.

Índice

Índice de Figuras

Índice de Tabelas

Siglas e Acrónimos

API	Application Programming Interface
BPMN	Business Pattern Model & Notation
CDC	Change Data Capture
DM	Data Mart
DQE	Data Qualite Enhancemente
DSA	Data Staging Area
DW	Data Warehouse
ETL	Extract-Transform-Load
IDL	Intensive Data Loading
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
KJB	Kettle Job
KTR	Kettle Transformation
LCR	Logique, Calcule et Raisonnement
LIP6	Laboratoire d'informatique de Paris 6
LIPN	Laboratoire d'informatique de Paris Nord
MoVe	Modélisation et Vérification
PDI	Pentaho Data Integration
PNML	Petri Net Markup Language
RPC	Redes de Petri Coloridas
SCD	Slowly Changing Dimension
SCD-H	Slowly Changing Dimension with History
SGML	Standard Generalized Markup Language

SKP	Surrogate Key Pipelining
SQL	Structured Query Language
UML	Unified Modeling Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language
YAWL	Yet Another Workflow Language

Capítulo 1

Introdução

1.1 Enquadramento

Numa era em que a informação gerada pelas organizações tem vindo a aumentar consideravelmente, e que para se subsistir no mundo empresarial é necessário usufruir de algumas vantagens em termos competitivos, aparece, naturalmente, a necessidade de melhorar o armazenamento e a exploração dos dados organizacionais. Os sistemas de *data warehousing* permitem colmatar algumas dessas lacunas, sobretudo ao nível da exploração dos dados, disponibilizando estruturas e serviços expeditos para a manipulação de dados, com especial perícia em termos de suporte a processos de tomada de decisão. Todavia, para serem úteis e eficazes estes sistemas têm que ser povoados com a informação adequada. Para povoar um *Data Warehouse* (DW) é necessário recorrer a um conjunto de programas um pouco particulares, que usualmente são reconhecidos como sistemas de ETL (Extract-Transform-Load). Os sistemas de ETL permitem que dados organizacionais, provenientes de diferentes fontes, possam ser acolhidos num único espaço, por forma a serem transformados e carregados para um repositório de dados especializado. Este processo garante, também, a qualidade dos dados importados, filtrando-os relativamente às suas possíveis impurezas (erros, formatos ambíguos, nulos, etc.), assegurando

que chegam nas condições requeridas ao DW destino. Como consequência, um sistema de ETL acrescenta valor ao próprio DW e consequentemente ao processo de tomada de decisão.

Um processo de ETL pode estar organizado em quatro fases distintas: extração, limpeza, transformação e carregamento, que ocorrem numa área denominada por *Data Staging Area* (DSA). Na primeira fase, extração, os dados são extraídos das diversas fontes de informação, de acordo com um plano de angariação previamente estabelecido, e colocados na DSA. Depois, durante a fase de limpeza, os dados extraídos são submetidos a processos de limpeza, nas quais se realizam alguns processos de validação de valores ou de remoção de registos duplicados, por exemplo. Com os dados retificados e homogeneizados, segue-se a sua transformação, que pode incluir, por exemplo, algumas tarefas de conversão de unidades de medida, geração de chaves de substituição (*surrogate keys*) ou processos de mapeamento de dados. Por último, é efetuado o carregamento dos dados para o DW.

Segundo Kimball e Caserta (2004), um sistema de ETL é o núcleo de um DW, em grande parte devido ao simples facto de a modelação conceptual de sistemas de ETL, por parte das organizações, ser feita de uma forma *ad-hoc* (Vassiliadis, Simitsis e Skiadopoulos, 2002). Isto, porém, pode originar uma má implementação do sistema de ETL, que irá resultar, naturalmente, numa perda na qualidade de informação dos dados e consequentemente, num aumento dos seus custos de implementação (English, 1999). De modo a tentar minimizar os custos de uma má implementação de um sistema de ETL, recomenda-se o desenho e a modelação conceptual de um sistema de ETL. É neste âmbito que, as *Redes de Petri Coloridas* (RPC), através das suas características, podem ser uma boa resposta à modelação de meta-tarefas em sistemas de ETL.

As RPC foram desenvolvidas pelo CPN Group da Universidade de Aarhus na Dinamarca, tendo Kurt Jensen como o seu principal impulsionador. São uma linguagem para modelação e validação de sistemas em que a concorrência, a comunicação e a sincronização desempenham um grande papel (Jensen et al., 2007). As RPC combinam as propriedades das Redes de Petri com a linguagem de programação funcional baseada em Standard ML. Os modelos de RPC descrevem os estados e eventos que podem levar o sistema a mudar o seu estado interno. Ao correr simulações de RPC é possível investigar diferentes cenários e explorar comportamentos do sistema. Através da capacidade hierárquica presente nas RPC, é possível especificar sistemas usando módulos. Estes podem ser simulados independentemente, oferecendo um melhor controlo sobre a exploração do comportamento do sistema. A capacidade formal aliada à capacidade de representação gráfica faz das RPC uma ferramenta poderosa em termos da modulação de sistemas.

Por essa razão é possível ver as RPC aplicadas em diferentes áreas, como os protocolos de comunicação (Billington & Han, 2007), os sistemas militares (Mitchell et al., 2007), os sistemas de controlo (Moncelet et al., 1998) ou em software. Grupos como a Peugeot, Citroën, a Nokia e a Hewlett-Packard possuem trabalhos de modelação usando RPC. Associado a tudo isto, existe uma ferramenta *open-source*, dominada por CPN Tools, também desenvolvida pelo CPN Group, que permite construir, simular e analisar modelos em RPC. Esta ferramenta está disponível para todos os sistemas operativos, apesar de se recomendar o uso da última versão estável para o sistema operativo Windows.

1.2 Motivação e Objetivos

Aplicar as RPC para modelar e especificar processos de ETL é algo que tem vindo a ser desenvolvido com sucesso. Duas meta-tarefas com grande peso num sistema de ETL - *Surrogate Key Pipelining* (Silva et al., 2012) e *Change Data Capture* (Silva et al., 2013) - foram modeladas e validadas recorrendo a modelos desenvolvidos com RPC. Estas são soluções para minimizar os custos relacionados com a implementação de sistemas de ETL. Isso é possível, porque havendo modelos pré-definidos de meta-tarefas de ETL é possível modelar e avaliar se os modelos refletem *a priori* o comportamento desejado para o sistema de ETL e, caso isso não aconteça, intervir já na fase de modelação, evitando-se, assim, intervenções mais sérias durante a fase de implementação e, consequentemente, gastar menos na generalidade do processo. Para isso, podemos, por exemplo, reutilizar os módulos das meta-tarefas que estão devidamente testadas e validades durante a modelação do sistema.

Este trabalho de dissertação tem como objetivo principal planear, desenvolver e implementar uma peça de software que seja capaz de pegar no modelo desenvolvido com RPC e gerar um "esqueleto" para um sistema de ETL, com vista ao seu acolhimento e execução numa ferramenta de construção de sistemas de ETL (Pentaho Kettel[1], MS-SQL Server Integration

[1] http://community.pentaho.com/projects/data-integration/

Services[2], ou outro). Através da geração destas estruturas é possível reaproveitar grande parte do trabalho realizado durante a fase de modelação conceptual do sistema ETL, o que de certa forma, implica que a geração de esqueletos para sistemas de ETL é uma mais valia no processo de construção de sistemas ETL, permitindo poupar tempo de implementação, bem como, por de parte as típicas implementações *ad-hoc*. Para tal, será necessário estudar aprofundadamente os ficheiros gerados pelo CPN Tools e pela ferramenta de construção de sistemas de ETL, de modo a que seja possível fazer a geração automática dos "esqueletos" para uma linguagem nativa de uma ferramenta de ETL a partir dos correspondentes modelos RPC.

1.3 Estrutura do Documento

Além do presente, este documento integra outros quatro capítulos. No capítulo 2 será abordado o "estado da arte" sustentando a forma como diferentes linguagens foram adaptas para modelar processos ETL, assim como, a exposição da modelação conceptual de sistemas de ETL, mais propriamente, expondo a forma como as RPC podem ser utilizadas na modelação conceptual deste tipo de sistemas. De seguida, será abordada a ferramenta que permite a criação dos modelos RPC de meta-tarefas ETL, o tipo de ficheiros exportados pela ferramenta, assim como as suas próprias características. Após a análise desse tipo de ficheiros será feita uma breve comparação entre os dois formatos exportados e uma sugestão relativamente ao formato a usar neste tipo de processos. Para fechar o capítulo, desenvolveu-se uma análise mais detalhada ao formato selecionado. De seguida, no capítulo 3, apresentar-se-á a descrição dos vários processos ETL modelados usando RPC para o efeito, apresentando-se também o caso de estudo utilizado como suporte aos processos de modelação realizados, bem como os modelos correspondentes. No capítulo 4 será apresentado o processo de geração de esqueletos ETL a partir dos modelos RPC que foi desenhado e implementado. Nesse capítulo, também é explicado o ciclo de vida do processo de desenvolvimento de esqueletos ETL e exposto o resultado final de todo o processo de

[2] http://www.microsoft.com/en-us/server-cloud/products/sql-server/default.aspx

transformação dos modelos para uma dada ferramenta de ETL. Por fim, no capítulo 5 serão apresentadas as conclusões acerca da trabalho realizado e apresentadas algumas linhas de trabalho para um futuro próximo.

Capítulo 2

Modelação Conceptual de Sistemas de ETL

2.1 Modelos Aplicados aos Sistemas de ETL

A modelação tem como objetivo identificar os conceitos básicos associados a um problema de grande dimensão. O modelo é então uma abstração do produto final, seja ele uma peça de software, um esquema de uma base de dados relacional ou um sistema de ETL. Sem dúvida que a modelação oferece enormes vantagens, no desenho, concepção e implementação do produto.

A modelação conceptual de sistemas de ETL não devia de ser exceção. O facto é que este tipo de modelação não tem um peso significativo na construção do sistema, sendo realizada usualmente de uma forma *ad-hoc* (Vassiliadis, Simitsis e Skiadopoulos, 2002), muito embora seja um processo crítico para o bom funcionamento de um sistema de *data warehouse* (Kimball e Caserta, 2004). Diversos autores defendem diferentes perspetivas em termos de impacto prático (Vassiliadis, Simitsis e Skiadopoulos, 2002; Kimball e Caserta, 2004), mas todos partilham a mesma ideia: a implementação e manutenção de um sistema de ETL é um processo que requer de muitos recursos. Contudo diversos autores têm contribuído, de forma significativa, com os seus trabalhos, para a evolução e enriquecimento da modelação conceptual de sistemas de ETL. Trabalhos como (Vassiliadis, Simitsis e Skiadopoulos, 2002), (Simitsis, 2003), (Simitsis e Vassiliadis, 2003), (Trujillo e Luján-Mora, 2003) e (Abelló, Samos e Saltor, 2006) são claros exemplos daquilo que foi realizado

durante os últimos anos no âmbito da modelação conceptual e lógica de sistemas de ETL. Algumas das propostas apresentam componentes próprios criados para satisfazer as necessidades que um dado modelo conceptual requer, como é o caso de Vassiliadis, Simitsis e Skiadopoulos (2003). Outras, como o caso da proposta apresentada por Trujillo and Luján-Mora (2003) usa a notação Unified Modeling Language (UML), em particular os diagramas de atividades. Mais recentemente, linguagens de *workflow* tais como BPMN (Business Pattern Model & Notation), Reo, YAWL (Yet Another Workflow Language) ou RPC surgiram com alguma expressividade ao serem empregues na modelação conceptual de sistemas de ETL. No caso do BPMN, por exemplo, os trabalhos realizados por Akkaoui (2009) e Akkaoui (2011) mostraram que é possível modelar tarefas ETL, e conduziram a forma como as linguagens de *workflow* poderiam ser usadas para modelar esse tipo de tarefas. Surge então a noção de modelação conceptual de processos padrão para sistemas de povoamento de DW (Oliveira e Belo, 2012) e (Oliveira e Belo, 2013), usando para isso a linguagem de workflow BPMN. Outras tentativas com outras linguagens de workflow, levaram à modelação de padrões ETL usando o Reo (Oliveira e Belo, 2013) e o YAWL (Oliveira e Belo, 2014). Os trabalhos realizados pelo (Silva et al., 2012) e (Silva et al., 2013) mostraram que a adaptação das RPC à modelação e validação de padrões ETL também é praticável, levando à criação de modelos com algum detalhe. As linguagens de *workflow* vieram trazer uma nova versatilidade à modelação de sistemas de ETL, pois permitem que a conceptualização do modelo não seja feita tão exaustivamente e possa ser realizada através de um conjunto de atividades permitindo uma maior abstração e aliando-se a uma modelação orientada a padrões ETL, tornam-se eficazes e permitem a criação de sistemas de ETL mais robustos e menos susceptíveis ao erro, aumentando a sua qualidade. Relativamente à geração automática de esquemas físicos ETL a partir de modelos conceptuais, não existem propostas relevantes que possam ser discutidas e que das quais, sejam retiradas informações relevantes para o processo de geração de esqueletos ETL.

Assim sendo, aqui abordamos a questão de como construir esqueletos correspondentes a um dado sistema de ETL, a partir dos modelos conceptuais gerados por uma dada ferramenta de modelação. Tal como acontece na modelação de sistemas de software, a partir de um diagrama de classes é possível gerar o código esqueleto correspondente. Com um modelo RPC o objetivo será idêntico.

2.2 CPN Tools

A ferramenta que utilizámos para a implementação, simulação e validação de um dado modelo ETL foi o CPN Tools. Esta é uma ferramenta *open-source* originalmente desenvolvida pelo CPN Group na Universidade de Aarhus. Começou a ser desenvolvida em 2000, por Kurt Jensen, Søren Christensen, Lars M. Kristensen e Michael Westergaard. Mais tarde, em 2010 passou a ser desenvolvida pelo AIS Group, da Universidade de Eindhoven. Atualmente, encontra-se na versão 4.0.0 e recomenda-se a sua utilização da versão para o sistema operativo Windows.

A ferramenta CPN Tools tem como principal objetivo construir e analisar os modelos RPC (Jensen, 2007). A partir da ferramenta é possível, editar, simular e analisar modelos RPC através do interface gráfico disponibilizado (Figura 2.1).

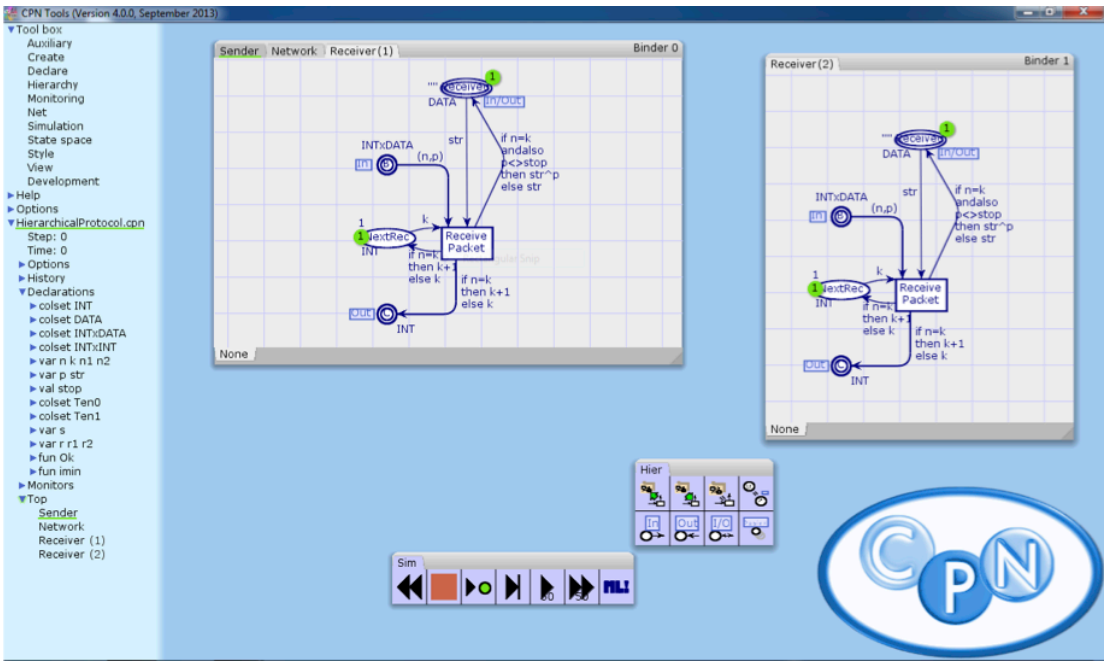


Figura 2.1 - Interface gráfico da ferramenta CPN Tools.

O interface gráfico do CPN Tools encontra-se divido em duas grandes áreas: o índice (*index*) e o espaço de trabalho (*workspace*). O índice é a área retangular localizada à esquerda na Figura 2.1. Um dos componentes do índice é a Caixa de Ferramentas (*Tool box*), que contém ferramentas para a manipulação dos diferentes componentes dos modelos RPC. Estão disponíveis ferramentas

para a criação de componentes básicos, para a alteração do posicionamento dos componentes, para a simulação do modelo, entre outras. Outro componente do índice, a vista global (*overview*), permite visualizar algumas informações relacionadas com os modelos abertos no CPN Tools. Informações essas que incluem o nome do modelo, no caso da Figura 2.1, 'HierarchicalProtocol.cpn', as declarações, os módulos e as estruturas hierárquicas presentes no modelo RPC.

O espaço de trabalho é a área do CPN Tools, que está localizada à direita do índice. A partir do espaço de trabalho é possível construir e simular os modelos RPC. Esta área pode conter um conjunto de caixas retangulares, chamadas de encadernadores (*binders*). Cada encadernador pode ser arrastado a partir do índice para o espaço de trabalho. Existem dois tipos de encadernadores. Na Figura 2.1 é possível visualizar esses dois tipos. Um deles contém os elementos do modelo RPC, i.e., os módulos e as declarações (*binder* 0 e *binder* 1), e o outro as ferramentas para a construção e manipulação dos modelos, podendo também ser arrastado a partir do índex (Sim, Hier). Cada encadernador pode ter múltiplas abas, no caso apresentado na Figura 2.1, o *binder* 0, tem 3 abas, Sender, Network e Receiver (1), o que permite de alguma forma uma melhor organização do espaço de trabalho.

Exportação do Modelo RPC

A modelação conceptual de um sistema de ETL oferece inúmeras vantagens. Uma delas passa pelo redução dos custos associados à construção de um sistema de ETL, já que este não é realizado de uma forma *ad-hoc*, mas sim desenvolvido com método e rigor. A possibilidade de associar algo mais ao modelo, como a simulação e validação do seu funcionamento oferece uma enorme vantagem em relação a outro tipo de modelação estática. A modelação de diferentes meta-tarefas de sistemas de ETL usando as RPC tem por base estes princípios. Apesar das vantagens da modelação é sempre necessário implementar o sistema de ETL de raiz. Não havendo uma ferramenta que consiga a partir do modelo RPC gerar o respetivo esqueleto do sistema de ETL. Surge então a necessidade de implementar uma peça de software para o efeito. Para isso, será usada uma das ferramentas que o CPN Tools possui, a exportação dos modelos RPC de meta-tarefas de sistemas de ETL.

A exportação do modelo RPC é um processo simples. Através da área índice no CPN Tools, na componente vista global, no nome do modelo aberto, neste caso, 'HierarchicalProtocol.cpn', é

possível aceder a um menu circular com o botão secundário do rato (Figura 2.2). Nesse menu a opção '*Save Net As*', permite exportar o modelo RPC em dois formatos diferentes: *Petri Net Markup Language* (PNML) ou *Extensible Markup Language* (XML). A exportação do modelo RPC em PNML é uma característica nova implementada na versão 4.0.0 do CPN Tools. Os dois formatos serão, em seguida, alvos de um estudo, para que seja possível fazer uma análise comparativa sobre si, com o intuito de determinar qual o formato mais vantajoso para a implementação da peça de software.

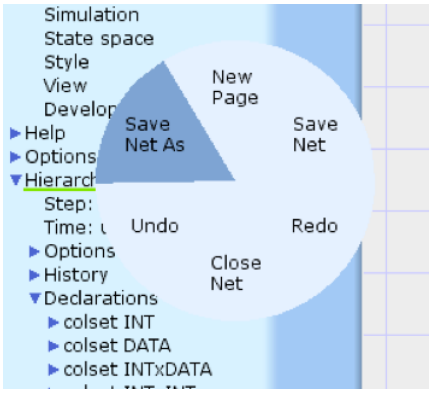


Figura 2.2 - Menu circular da ferramenta CPN Tools.

2.3 Petri Net Markup Language

Petri Net Markup Language (PNML) é um formato de permuta baseado em XML para redes de Petri. A ideia principal para o desenvolvimento deste formato partiu da premissa que, seria útil um formato capaz de transferir modelos redes de Petri entre as diferentes ferramentas que possam existir (Billington et al., 2003). Com este tipo de formato é possível usufruir de vantagens que outras ferramentas recentemente desenvolvidas possam dispor, já que o modelo pode ser exportado por uma ferramenta e importado por outra sem que o conteúdo do modelo altere. Tarefas como simulação, análise e implementação não estão cingidas a uma única ferramenta, mas sim a um conjunto variado delas. Seria possível então, desenvolver um modelo rede de Petri numa ferramenta, simular noutra ferramenta e analisar os resultados numa terceira. Para que a permuta entre ferramentas possa ocorrer é necessário adotar um formato standard. No ano 2000 durante a realização da Conferência Internacional sobre a Aplicação e Teoria de Redes de Petri foram propostos vários formatos de permuta baseados em XML, sendo o ponto de partida para o

desenvolvimento de um formato *standard* (Billington et al., 2003). Atualmente o formato PNML é um *standard* reconhecido internacionalmente pelo comité da International Organization for Standardization e a International Electrotechnical Commission (ISO/IEC) pela norma ISO/IEC 15909. Esta norma encontra-se dividida em duas partes: a parte 1, ISO/IEC 15909-1, que contém os conceitos, definições e notação gráfica para as redes de Petri; e, a parte 2, ISO/IEC 15909-2, que aponta para o formato de transferência. Uma terceira parte está a ser desenvolvida e visa o uso de extensões no formato, i.e., inclusão de construtores de tempo e de modularidade.

O formato PNML foi desenhado com o objetivo de permitir a permuta de redes de Petri independentemente de qualquer tipo de ferramenta ou plataforma. Sendo assim, o desenho de PNML está orientado aos seguintes princípios:

- flexibilidade: qualquer rede de Petri deve ter representação no formato PNML com as respetivas extensões e características;
- compatibilidade: entre diferentes tipos de redes de Petri a informação permutada deve ser o máximo possível.

Na Figura 2.3 podemos ver a organização dos meta-modelos PNML em notação UML.

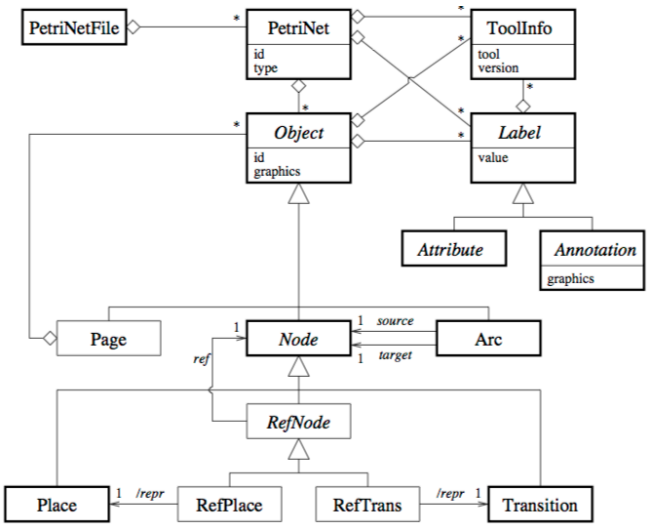


Figura 2.3 - Organização estrutural do formato PNML em notação UML.

Os objetos com os quais é possível modelar uma rede de Petri são: o arco (*arc*), a transição (*transition*) e o local (*place*). Estes são os objetos básicos que irão compor um ficheiro PNML, que por convenção são designados por nodos (*nodes*). Por sua vez, cada objeto tem associado um identificador único, que pode ser usado para referenciar outro objeto. Além disso, possui também um rótulo (*label*), que acrescenta algum significado ao próprio objeto. Quanto aos rótulos, estes podem ser de dois tipos: anotação (*annotation*) ou atributos (*attributes*). Um rótulo do tipo anotação serve essencialmente para mostrar informações acerca dos objetos correspondentes. Tendo em conta a Figura 2.1, na aba Reciever(1) é possível observar as diferentes anotações presentes. São exemplo, as anotações: INTxDATA, (n,p), DATA, entre outras. Ao rótulo de atributos estão associadas as propriedades gráficas do objeto, i.e. cor, estilo, forma, entre outras. O atributo gráfico (*graphics*), que está presente tanto nos objetos como nos rótulos do tipo atributo, dizem respeito os diferentes posicionamentos que os vários objetos e rótulos podem assumir na área de trabalho. Em cada objeto e em cada rótulo pode estar especificado o tipo de ferramenta que criou o ficheiro PNML para a rede de Petri correspondente. Além disso, ainda existe um outro objecto denominado página (*page*) que acolhe vários nodos em si.

Ferramentas para a Interpretação do Formato PNML

Para desenvolver uma peça de software capaz de transformar modelos RPC de meta-tarefas ETL em esqueletos para sistemas de ETL, é necessário extrair a informação contida no formato PNML referente ao modelo. Para isso, existe uma série que ferramentas[3] que são capazes de trabalhar sobre o formato standard PNML. Entre as diferentes ferramentas disponíveis, a que de alguma forma é capaz de responder ao problema em questão, extrair a informação contida no formato PNML, é a PNML Framework (Hillah et al., 2010).

A PNML Framework foi desenvolvida e é mantida em parceria entre MoVe do LIP6[4] e a LCR do LIPN[5]. Esta ferramenta disponibiliza uma *Application Programming Interface* (API),

[3] http://www.pnml.org/tools.php

[4] Laboratoire d'informatique de Paris 6

[5] Laboratoire d'informatique de Paris Nord

implementada em Java, para escrever e ler ficheiros com o formato standard PNML de acordo com a norma ISO/IEC 15090-2. Uma das vantagens do uso da *framework* passa por libertar o utilizador de uma programação usando construtores XML, passando a usar os conceitos das redes de Petri (Hillah et al., 2010), o que resulta num menor esforço de programação para obter as informações acerca dos modelos. A *framework* disponibiliza um pacote de documentação bastante interessante que pode ser consultado para uma melhor compreensão acerca dos seus serviços e funcionalidades.

2.4 Extensible Markup Language

O formato *Extensible Markup Language*, também conhecido por XML é um outro tipo de formato exportado pela ferramenta CPN Tools. Este permite descrever, parcialmente, o comportamento de programas que os processam (Bray et al., 2006). Um documento em XML encontra-se conforme as normas do *Standard Generalized Markup Language* (SGML). A norma SGML define as regras gerais para a construção de um documento numa linguagem *markup* (Goldfarb, 1991). O formato XML foi desenvolvido por um grupo de trabalho, originalmente conhecido por SGML *Editoral Review Board*, formado em 1996 sobre o *World Wide Web Consortium* (W3C). O formato em si é composto por unidades de armazenamento denominadas entidades. Uma entidade pode conter dois tipos de informações: dados analisados e dados por analisar. Aquando da sua conceção, o formato XML seguiu uma série de características que abordam aspectos como:

- ser facilmente partilhado e usado pela comunidade web;
- suportar um grande número de aplicações;
- ser compatível com SGML;
- ser fácil e simples escrever programas que processem documentos XML;
- o número de características adicionais do XML deve ser mantido no mínimo valor possível;
- um documento XML deve ser legível por um humano;
- o desenho de um documento XML deve ser preparado rapidamente;
- o desenho de um documento XML deve ser formal e conciso;
- um documento XML deve ser fácil de criar;
- a concisão nas etiquetas do XML é algo com importação mínima.

A leitura do formato é realizada através da construção de pequenos módulos de software usualmente chamados de processadores de XML. Processar XML pode ser feito recorrendo a diversos tipos de linguagens de programação - Java, Python, PHP, Ruby, Perl são alguns dos exemplos de linguagens capazes de ler ficheiros do formato XML. A escolha da linguagem recai na necessidade de velocidade de processamento, gestão de memória ou outro tipo de necessidade que o programador possa ter em particular. Além da linguagem de programação, existe um número considerável de APIs capazes de ler e escrever ficheiros em formato XML, facilitando de certa forma o trabalho do programador.

A leitura e posterior extração de informação de modelos RPC exportados pela ferramenta CPN Tools para o formato XML, não possuí a vantagem que o formato PNML oferece: uma API bem definida com uma documentação associada. Como tal, para extrair a informação inerente ao modelo RPC exportado será necessário uma análise mais cuidada do ficheiro XML. Para isso, naturalmente, é preciso encontrar os elementos chave para a construção dos esqueletos de sistemas de ETL. Parte desses elementos chave são: os lugares, as transições, os arcos e as páginas. No ficheiro XML exportado pelo CPN Tools são identificados pelos seguintes elementos:

- <place id="string"></place>
- <trans id="string"></trans>
- <arc id="string"></arc>
- <page id="string"></page>

Para representar um lugar, uma transição, um arco e uma página, são utilizadas, respetivamente, as etiquetas *place*, *trans*, *arc* e *page*. A cada um destes elementos estão associadas mais informação na sua subárvore, tais como informações gráficas, anotações, entre outras, exceto para o caso da página que, na sua subárvore, estão associados os lugares, as transições e os arcos. No caso do arco tem associado os identificadores dos dois objetos que interliga, neste caso um lugar a uma transição ou vice-versa. Resumindo, o esforço de leitura de um ficheiro no formato XML é maior que o esforço despendido através do uso da PNML Framework para o formato PNML.

2.5 Análise Comparativa entre PNML e XML

Analisados os dois tipos de formatos exportados pelo CPN Tools, o passo seguinte é fazer uma comparação entre ambos, de modo a que seja possível encontrar a melhor solução para fazer a implementação da peça de software adequada, com o intuito de transformar um modelo RPC num esqueleto correspondente para o sistema de ETL.

O formato PNML oferece uma API - a PNML*framework* - que proporciona ao programador a capacidade de abstração dos construtores XML, usando um conjunto de métodos para obter a informação contida no modelo RPC. Ao passo que usando o formato XML o programador precisa de construir um processador de XML para poder aceder à informação contida no ficheiro, com a utilização do formato XML é necessário ter uma abordagem diferente na exploração do ficheiro. Se com o formato PNML e a API da PNML *framework* existe uma documentação associada para consulta, no formato XML essa documentação simplesmente não existe.

Tomemos como exemplo a tarefa de obtenção de todos os identificadores associados aos lugares num modelo RPC. Para realizar esta ação na PNML *framework*, basta consultar a documentação, encontrar o(s) método(s) capaz(es) de efetuar a extração dos identificadores associados aos lugares. A mesma ação com o formato XML não possui documentação passível de consulta que mostre como extrair os identificadores associados aos lugares num modelo RPC. É então prioritário analisar o formato XML de um modelo RPC, encontrar os elementos que identificam um lugar, construir um processador e no final extrair os identificadores. A falta de documentação para o formato XML na extração de informações sobre os modelos RPC deve ser interpretada desta forma.

Embora ambos os formatos disponham de APIs capazes de ler e extrair as informações dos ficheiros, no caso do formato PNML a API está limitada ao paradigma de programação orientado aos objetos, a linguagem Java. Este aspeto pode ser encarado como uma vantagem ou uma desvantagem, sendo o seu valor subjetivo de programador para programador. Embora à partida o formato PNML seja mais vantajoso que o formato XML, sendo o mais indicado para importar e extrair a informação do modelo RPC exportado pelo CPN Tools, não será a solução usada para o problema em questão. Uma limitação relativamente à exportação de ficheiros no formato PNML a partir da ferramenta impede o uso deste formato. A limitação prende-se com o facto de que a exportação segue um formato pré-standard, isto é, a organização interna do ficheiro, a sua estrutura, bem como os seus construtores, não estão conforme o formato standard definido pela

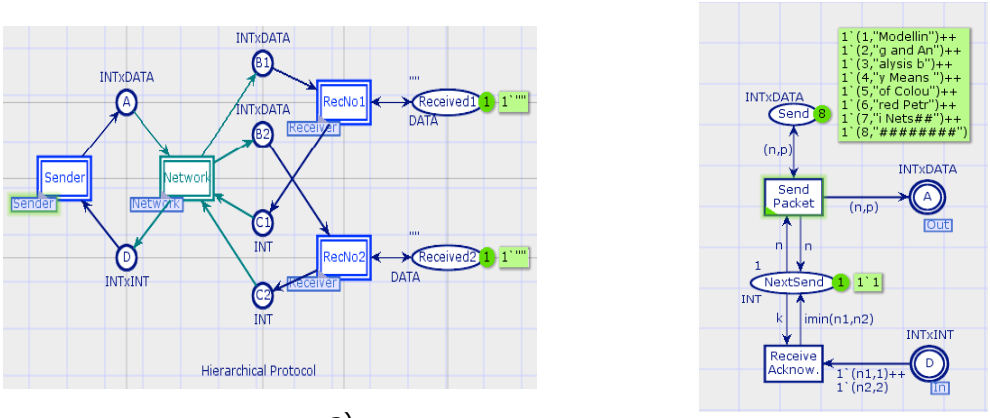
norma ISO/IEC 15909. Posto isto torna-se difícil o uso da PNML *framework* como API para a extração da informação do modelo RPC exportado. Com a presença desta limitação, será necessário a construção de um processador de XML, usando a linguagem de programação orientada a objetos Java para contornar o problema. Na Tabela 2–1 podemos encontrar um resumo acerca da comparação dos dois formatos, com referência aos diferentes aspetos previamente mencionados.

Tabela 2–1 Diferentes aspetos dos formatos PNML e XML.

	PNML	**XML**
API	PNML Framework para leitura e escrita de ficheiro PNML.	Nada concreto para leitura e escrita de ficheiro com meta-informação referente a modelos RPC. Apenas APIs primitivas para extração dos elementos.
Documentação	Presente na PNML Framework.	Sem documentação associada a extração de meta-informação de modelos RPC.
Linguagem	Limitado (Java).	Diferentes recursos (Perl, Java, PHP, Phyton).
Abstração	Existente devido à PNML Framework.	Não existente, sem API capaz de abstrair o programador dos construtores XML.

2.6 Análise do Ficheiro XML Exportado pelo CPN Tools

Estando já analisados e comparados os ficheiros exportados pela ferramenta CPN Tools e concluindo que, apesar de tudo, o formato XML é o mais adequado para a construção de uma peça de software capaz de traduzir o modelo RPC num esqueleto para posterior adoção numa ferramenta de construção de sistemas de ETL, vamos de seguida apresentar e discutir um possível processo para a dissecação do modelo exportado pelo CPN Tools. Já aqui foi referido, que os elementos chaves para a construção de modelos RPC são: os lugares, as transições, os arcos e as páginas. Contudo é necessário compreender também como a estrutura interna do ficheiro XML se encontra organizada, para que se possa extrair a informação necessária à construção do correspondente esqueleto. Assim, em seguida, serão analisados os diferentes elementos que estão integrados na meta-informação envolvida.



a) b)

Figura 2.4 –Exemplo de um protocolo hierárquico da ferramenta CPN Tools.

2.6.1 Lugares

O lugar, ou *place*, é um dos dois tipos de nodos presentes num modelo RPC. Para este tipo de nodo, a informação que foi necessária extrair para se formar o esqueleto correspondente foi a seguinte: o identificador do lugar, o nome do lugar, o seu posicionamento (coordenadas cartesianas) e o seu tipo (opcional). O identificador é um valor único associado ao lugar, que serve essencialmente para identificar de forma inequívoca o lugar. O nome do lugar tem como intuito determinar o elemento ETL que vai dar lugar, bem como definir o nome desse mesmo elemento. O posicionamento do lugar, tal como o próprio nome indica, serve para posicionar o elemento ETL no ambiente da ferramenta. Um lugar pode ter três tipos distintos: Entrada (*In*), Saída (*Out*) ou Entrada/Saída (*I/O*). No caso do modelo RPC, como existe a hierarquização do modelo, os *tokens* que estão presentes no lugar podem ter que descer ou subir de nível na hierarquia definida. Nem todos os lugares possuem associados à sua subárvore um tipo. Este apenas surge quando existe a necessidade para tal. De seguida apresenta-se um pequeno excerto demonstrativo daquilo que foi acima referido. O excerto extraído refere-se a um modelo RPC presente nos exemplos da ferramenta CPN Tools apresentados anteriormente na Figura 2.4 b).

```
<place id="id442">
  <posattr x="-57.000000" y="48.000000"/>
  (...)
  <text>D</text>
  (...)
  <port id="ID5563" type="In">
    (...)
  </port>
</place>
```

Neste pequeno excerto XML, podemos ver que o elemento XML *posattr* possui as coordenadas cartesianas do lugar, o elemento XML *text* contém o nome do lugar e o elemento XML *port* possui o atributo *type* que diz respeito ao tipo do lugar, neste caso é de entrada.

2.6.2 Transições

A transição é o outro tipo de nodo presente num modelo RPC. Neste tipo de nodo, a informação a extrair será semelhante à informação extraída do nodo lugar. Sendo assim, a informação relevante para a sua consequente transformação num esqueleto será: o identificador da transição, o nome da transição, o seu posicionamento (coordenadas cartesianas), informações relativa à subpágina, caso esta exista, e os identificadores dos lugares nos quais existam componentes que façam a transferência de *tokens* entre os diferentes níveis da hierarquia. O identificador, nome e posicionamento já foram explicados anteriormente e como tal não será feita aqui realizada uma nova explicação. A parte interessante numa transição é que ela pode ser um apontador para uma nova página, o que torna um modelo RPC muito mais modular e fácil de simular e validar. Tudo isto porque cada página pode ser simulada independentemente do modelo definido. Existe então a informação referente a esse apontador. Esta pode ser encontrada através do elemento XML *subst* no atributo *subpage*. Nesse atributo existe um identificador para a página correspondente, neste caso é a página com o identificador *id417*. Ainda no elemento XML *subst*, este tem um atributo denominado *portsock*. Nele podemos encontrar pares de identificadores. Estes identificadores dizem respeito a lugares nos quais os *tokens*, dependendo do tipo associado ao lugar, saem, entram, ou saem e entram. A primeira componente do par de identificadores refere-se ao lugar da página para a qual a transição aponta. Neste caso o identificador *id442* encontra-se noutra página. O segundo componente do par, refere-se ao lugar da própria página que irá receber e/ou enviar os *tokens* para os diferentes níveis da hierarquia. Também podemos encontrar outra informação relevante no atributo *name* do elemento XML *subpageinfo*. Este elemento irá servir para identificar

o padrão ETL. De seguida, apresentamos um pequeno excerto em XML do mesmo modelo RPC que apresentámos anteriormente na Figura 2.4 a).

```
<trans id="id291" explicit="false">
  <posattr x="-213.000000" y="219.000000"/>
  (...)
  <text>Sender</text>
  (...)
  <subst subpage="id417" portsock="(id451,id262)(id442,id256)">
    <subpageinfo id="ID40061" name="Sender">
      (...)
    </subpageinfo>
  </subst>
  (...)
</trans>
```

2.6.3 Arcos

Num modelo RPC, os arcos têm o intuito, essencialmente, de ligar e passar informação entre os nodos. Porém, apresentam uma pequena restrição: num modelo RPC, um arco nunca pode ligar dois nodos do mesmo tipo, isto é, as ligações devem ser de transições para lugares ou vice-versa. Um mesmo arco também não pode ligar dois lugares ou duas transições. Posto isto, a informação relevante a extrair dos arcos será, nomeadamente: o identificador, a orientação, a transição final e o lugar final. Relativamente ao identificador, este já foi discutido anteriormente e, como tal, não há a necessidade de uma nova explicação. Quanto à transição final e ao lugar final, estes estão representados pelos elementos XML, *transend* e *placeend*, respetivamente. Nestes elementos existe um atributo denominado *idref* que possui um identificador da transição ou do lugar, dependendo do elemento XML em questão. Para saber se um arco está a ligar uma transição a um lugar ou, então, um lugar a uma transição (ou em ambas as orientações) é necessário recorrer ao atributo *orientation* no elemento XML *arc* para que seja possível retirar qualquer conclusão. Neste atributo existem três tipos distintos de texto que podem surgir, de referir *PtoT*, *TtoP* ou *BOTHDIR*. Caso seja, *PtoT*, o arco está a ligar um lugar a uma transição. O P corresponde a *place* (lugar) e o T corresponde a *transition* (transição). Seguindo a mesma lógica, *TtoP* está a ligar uma transição a um lugar. Caso seja *BOTHDIR* quer dizer que o arco tem a dupla orientação. Este atributo é importante pois, aquando da transformação do modelo RPC num esqueleto é necessário saber que diferentes componentes estão ligados, bem como a sua orientação para que seja possível fazer a

sua recriação no processo de geração do esqueleto. Como exemplo, apresenta-se de seguida um pequeno excerto XML contendo a informação mais relevante associada a um arco (Figura 2.4 b)).

```
<arc id="id500" orientation="PtoT" order="0">
  (...)
  <transend idref="id431"/>
  <placeend idref="id442"/>
  (...)
</arc>
```

2.6.4 Páginas

Uma página num modelo RPC é, essencialmente, a sua entidade principal. Cada página tem associada a si os lugares, as transições e os arcos. Sendo assim, a informação mais relevante de uma página a ter em conta na sua transformação para integração num esqueleto será: o identificador da página, o nome da página e todos os elementos que a compõem, nomeadamente, os lugares, as transições e os arcos. De seguida, apresentamos um excerto demonstrativo da organização em XML relativa ao modelo RPC exportado (Figura 2.4 a)).

```
<page id="id4">
  <pageattr name="Network"/>
  <place id="id442">
    <posattr x="-57.000000" y="48.000000"/>
    (...)
    <text>D</text>
    (...)
    <port id="ID5563" type="In">
      (...)
    </port>
  </place>
  (...)
  <trans id="id291" explicit="false">
    <posattr x="-213.000000" y="219.000000"/>
    (...)
    <text>Sender</text>
    (...)
    <subst subpage="id417" portsock="(id451,id262)(id442,id256)">
      <subpageinfo id="ID40061" name="Sender">
        (...)
      </subpageinfo>
    </subst>
    (...)
  </trans>
  (...)
  <arc id="id500" orientation="PtoT" order="0">
    <posattr x="0.000000" y="0.000000"/>
```

```
    (...)
    <transend idref="id431"/>
    <placeend idref="id442"/>
    (...)
  </arc>
</page>
```

Após termos feito a "dissecação" do ficheiro XML exportado pelo CPN Tools de forma a podermos entender a sua organização e localizar os diversos lugares da informação relevante para cada um dos elementos chaves do modelo RPC, passámos à fase de construção do processador XML necessário e das estruturas necessárias para guardar toda a informação envolvida no processo. Para isso, utilizámos a linguagem de programação orientada a objetos – Java. A escolha recaiu nesta linguagem pelo simples facto de termos bastante experiência no desenvolvimento de peças de software em Java, por ser uma linguagem "universal" (multi-plataforma) de desenvolvimento de software e por dispor de várias bibliotecas para a leitura e interpretação de ficheiros XML.. Esta é, pois, uma etapa bastante importante, visto que não será possível utilizar a PNML Framework que fazia a abstração dos construtores XML. Pelo contrário. Tivemos que criar o processador XML para que fosse possível fazer a transformação do modelo em esqueleto para uma posterior inclusão numa ferramenta de construção de sistemas de ETL.

Capítulo 3

Especificação de Padrões ETL com RPC

3.1 Padrões ETL

Um sistema de ETL apresenta diferentes tipos de complexidade, dependendo obviamente da própria complexidade do DW. A complexidade de um sistema de ETL está associada ao número de tarefas ou meta-tarefas que este pode ter. Contudo, existe sempre um conjunto base de tarefas que, de uma forma ou de outra, encontramos sempre implementado num sistema de ETL. A essas tarefas mais comuns usualmente são denominadas por padrões ETL, sendo que um padrão é um elemento de trabalho que representa uma tarefa frequentemente utilizada num sistema de povoamento. Existem diversos padrões ETL que podemos ver implementados neste tipo de sistemas, como por exemplo: *surrogate key pipelining* (SKP), *slow changing dimension* (SCD), *change data capture* (CDC), *data quality enhancement* (DQE) ou *intesive data loading* (IDL). A identificação de padrões ETL permite que, quando modelados recorrendo às RPC, possam ser reutilizados em futuras modelações, já que uma das vantagens que as RPC oferecem é a modularização, o que permite reutilizar o padrão entre diferentes modelos. Além disso, como as RPC permitem a simulação e a validação dos diferentes módulos independentemente do modelo em questão, é seguro incorporar um módulo padrão que já foi previamente testado e validado respondendo-se assim àquilo que é requerido para esse padrão. Três desses padrões ETL – SKP,

SCD e CDC – foram previamente modelados e validados usando RPC. De seguida, neste capítulo, realizaremos uma análise detalhada relativamente a cada um desses, mostrando como estes foram modelados e qual a sua importância para o desenvolvimento e aceitação da sua modelação conceptual para sistemas de ETL.

3.2 Caso de Estudo

O *Data Mart* (DM) representado na Figura 3.1 foi aquele que decidimos utilizar como o nosso caso prático para desenvolvermos a modelação do sistema de ETL.

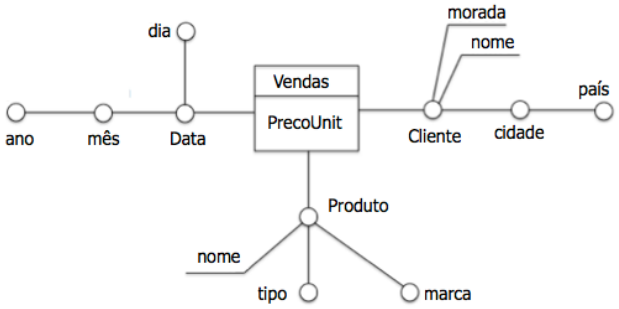


Figura 3.1 - Esquema do Data Mart de Vendas.

Como podemo ver, o DM escolhido possui três dimensões e uma tabela de factos. As dimensões apresentadas são, respetivamente: "Data", "Cliente" e "Produto". A dimensão "Data" é composta por três atributos dimensionais: "dia", "mês" e "ano". Nesta dimensão existem duas hierarquias que são formadas pelos seguintes atributos dimensionais: Data -> dia e Data -> mês -> ano. A dimensão Cliente possui dois atributos dimensionais: "cidade" e "país"; e dois atributos descritivos: "nome" e "morada". Relativamente aos atributos dimensionais, estes formam a hierarquia: Cliente -> cidade -> país. Por último, a dimensão "Produto" é formada por dois atributos dimensionais: "tipo" e "marca", que integram duas hierarquias, respetivamente: Produto -> tipo e Produto -> marca. Existe também um atributo descritivo: "nome", de forma a dotar a dimensão de alguma informação adicional. A tabela de factos "Vendas" agrega a informação do produto vendido a um determinado cliente numa certa data, integrando apenas o atributo "PrecoUnit" como medida.

Estando apresentado o DM utilizado como base de trabalho no processo de modelação RPC dos padrões ETL é altura adequada para revelar como tais padrões foram modelados.

3.3 Surrogate Key Pipelining

SKP é o processo responsável pela conversão de chaves naturais pelas correspondentes chaves de substituição (*surrogate keys*) nas dimensões referenciadas pelos factos antes que estes sejam carregados na respetiva tabela de factos. A existência de um processo SKP para cada tabela de factos presente num DW é algo muito comum, o que faz com que este processo possa ser considerado um padrão ETL. Existem diferentes maneiras de implementar este processo, como por exemplo, recorrendo a tabelas de mapeamento para gerar e gerir estes atributos. Contudo, neste trabalho será utilizado outro tipo de abordagem para que seja possível obter um melhor desempenho do sistema de ETL: a utilização de tabelas de *lookup*. Estas tabelas são fundamentais em qualquer processo SKP, uma vez que têm um tamanho reduzido quando comparado com o das tabelas de dimensão, sendo assim passíveis de serem carregadas para memória, evitando leituras de ficheiros em disco que podem prejudicar o desempenho do próprio sistema. Por norma, cada registo destas tabelas é composto pela chave de substituição gerada durante o processo de ETL para a dimensão em questão, e uma ou mais chaves naturais presentes nos dados provenientes das fontes organizacionais. No final do processo as chaves naturais são convertidas em chaves de substituição. A importância do uso destas chaves ao invés das chaves naturais, reside no facto de que, por norma, as chaves naturais nas fontes organizacionais são algo mais que um identificador inequívoco para um registo numa tabela, possuem por vezes, também, algum significado. Porém, é recomendável que todas as chaves num DW sejam desprovidas de qualquer "inteligência". As chaves de substituição têm esse propósito: converter as chaves naturais numa simples sequência de números. Além disso, num contexto empresarial, sabemos que as chaves naturais podem mudar ao longo do tempo, enquanto que num DW isso não é realmente uma boa prática. Daí a importância de tal processo num sistema de ETL.

Para o caso de estudo em questão, o padrão SKP opera sobre três dimensões – "data", "cliente" e "produto"-, sendo os registos provenientes da mesma fonte operacional, tendo apenas uma chave natural. Assim, serão usadas três tabelas de *lookup*, cada uma delas correspondendo a uma das três dimensões já referidas.

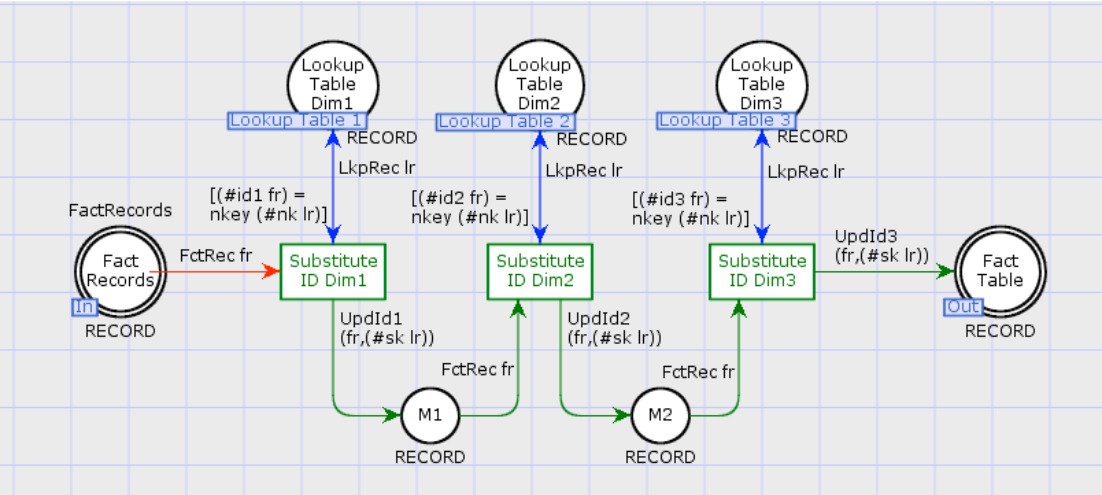


Figura 3.2 – Modelo RPC para o padrão ETL SKP.

A Figura 3.2 apresenta o modelo RPC do padrão ETL SKP. Este modelo é composto por sete lugares e três transições. O lugar *Fact Records* possui os registos provenientes da fonte operacional, ainda com as chaves operacionais, enquanto que o lugar *Fact Table* contém os registos após o processo de substituição das chaves naturais pelas respectivas chaves de substituição. Os lugares *Lookup Table Dim1*, *Lookup Table Dim2* e *Lookup Table Dim3* correspondem às tabelas de *lookup*, para cada uma das dimensões presentes no DM (Figura 3.1). Os lugares *M1* e *M2* tem o intuito de simular os registos em posições de memória, isto é, durante o processo o registo vai ocupando os lugar *M1* e *M2* após a substituição da chave natural pela chave de substituição. As transições *Substitute ID Dim1*, *Substitute ID Dim2* e *Substitute ID Dim3* representam a substituição da chave natural pela chave de substituição. O modelo RPC do padrão ETL SKP fica então assim composto, de uma forma simples e clara, mostrando que é possível modelar padrões ETL recorrendo a RPC. É, também, uma prova clara que a posterior transformação do modelo num esqueleto é algo bastante concretizável, o que permite desenvolver um processo de construção de sistemas de ETL mais rico e menos susceptível a falhas.

3.4 Slowly Changing Dimension

O processo de tomada de decisão deve ser feito recorrendo a dados que possam traduzir, de uma forma exata e correta, aquilo que é a realidade empresarial. Uma das características presente num

DW é a evolução dos seus dados ao longo do tempo e a não a sua volatilidade, isto é, as alterações que ocorrem nas fontes de dados operacionais devem ser acompanhados pelo DW e, como tal, nenhuma informação deve ser descartada do mesmo. Por este motivo é necessário ter especial atenção às dimensões com atributos de variação temporal. Este tipo de dimensões são usualmente designadas por dimensões de variação lenta.

Para implementar este tipo de padrão ETL existem diferentes estratégias. Porém, todas elas dependem da forma como queremos (ou não) manter os dados no DW. Tomemos como exemplo o DM da Figura 3.1. Olhando para a dimensão "Cliente" é possível que, no espaço temporal, um determinado cliente possa mudar a sua morada ou mudar-se para outra cidade. Para um determinado agente de decisão a mudança de morada pode ser importante ou não para o processo de tomada de decisão. Caso o seja, o DW deve estar preparado para acolher estas mudanças na dimensão, mantendo, ainda assim, o histórico dos valores anteriores, para que a realidade não seja deturpada e o agente de decisão possa ver o processo de tomada de decisão melhorado. Este tipo de manutenção de dados históricos faz com que a variação de dimensão lenta passe a ser uma dimensão de variação lenta com manutenção da história (SCD-H).

O processo SCD-H mantém o histórico dos dados através duma tabela adicional com cardinalidade N:1 em relação à dimensão de variação lenta. Essa tabela irá ser posteriormente alimentada durante o processo ETL sempre que um valor seja atualizado nos sistemas operacionais. Na tabela histórica é criado um registo contendo a chave de substituição referente ao registo atualizado, o valor do registo antigo e a data em que a alteração foi realizada. Na dimensão de variação lenta é atualizado o registo após o mesmo ter sido movido para a tabela com os dados históricos.

3.4.1 Módulo Geral

A Figura 3.3 apresenta o modelo RPC para o padrão ETL SCD-H. O modelo encontra-se integra três lugares e quatro transições. Devido à capacidade hierárquica presente nas RPC é possível desenvolver este processo através de uma modulação por módulos, sendo que cada transição neste modelo representa outra página com um nível de detalhe superior. Isto torna não só o modelo mais legível e expressivo, como favorece a simulação e validação independentemente dos outros módulos.

Relativamente às estratégias de modelação usadas para este padrão, estas passaram por dividir o processo ETL nas três operações principais - inserção, remoção e atualização - e adicionar uma componente para a verificação da qualidade dos dados de auditoria. A transição *Audit Data Verification* está, então, encarregue dessa verificação e as transições *Insert Record*, *Delete Record* e *Update Record* dizem respeito às operações de inserção, remoção e atualização, respetivamente. O lugar *Audit Records* possui os registos que irão alimentar as atividades presentes no modelo e o lugar *Verified Audit Records* serve como ponte para acolher os registos que passaram pelo processo de verificação. Este último lugar também encaminhará os registos verificados para as diferentes operações. Por fim, o lugar *Slowly Changing Dim* terá todas as modificações presentes no sistema operacional traduzidas em registos na dimensão.

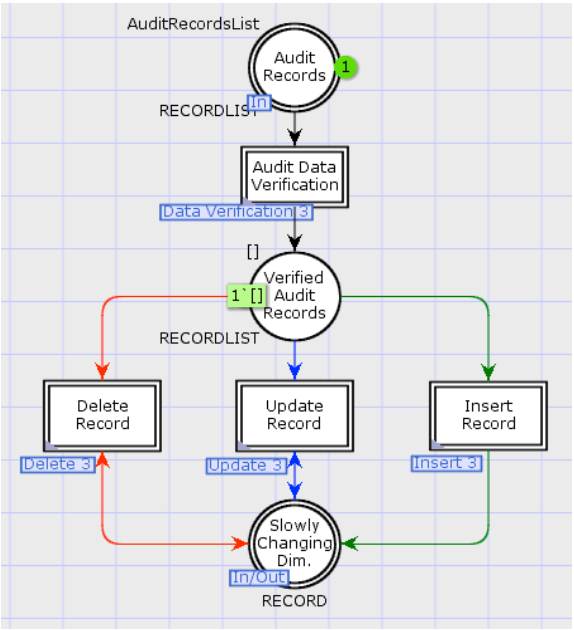


Figura 3.3 - Módulo geral do padrão SCD-H em RPC.

3.4.2 Módulo de Verificação dos Dados

O módulo de verificação dos dados (Figura 3.4) é o modelo RPC detalhado da transição *Audit Data Verification* (Figura 3.3). Este módulo é responsável por assegurar que os dados que irão alimentar

o DW estão de acordo com as regras de negócio estabelecidas. O modelo é composto por seis lugares e uma transição. O lugar *Audit Records* contém os registos das tabelas de auditoria e é o responsável por alimentar o módulo. Os lugares *Error Log*, *Quarantine Table* e *ETL Log* registam o resultado da verificação dos dados. O lugar *Error Log* é atualizado sempre que um registo é colocado em quarentena, com a marca temporal, a descrição do erro e o registo em quarentena. O lugar *Quarantine Table* guarda os registos que foram movidos para quarentena que não respeitaram determinadas condições de aceitação para que possam, posteriormente, ser tratados pelo administrador do DW e nele integrados. O lugar *ETL Log* tem a mesma função que o *Error Log* com a pequena diferença de que o *ETL Log* vai registando todos os acontecimentos que durante o processo de ETL possam ocorrer, quer seja por um registo ser movido para quarentena como uma inserção, remoção ou atualização registos. É importante registar este tipo de operações pois, caso o processo de ETL pare de executar por algum motivo, existe um *log* onde é possível consultar aquilo que já foi executado e retomar o processo de ETL a partir desse ponto.

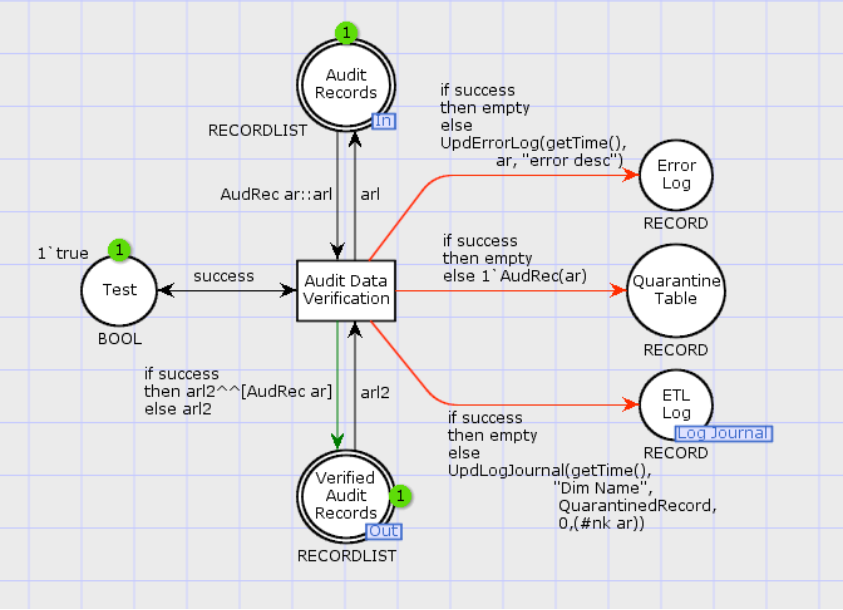


Figura 3.4 - Módulo de verificação dos dados do padrão SCD-H em RPC.

O processo de verificação é realizado recorrendo apenas a um lugar e a uma transição, *Test* e *Audit Data Verification*, respetivamente. Este processo é simulado através da atribuição aleatória

do valor verdadeiro ou falso. O processo de verificação serve apenas para demonstrar que este processo é passível de ser modelado através de RPC. Por último, o lugar *Verified Audit Records* contém os registos que passaram com sucesso no processo de verificação e serão reencaminhados para os próximos módulos.

3.4.3 Módulo de Inserção dos Registos

O modelo representado na Figura 3.5 é aquele que é responsável por inserir os registos na dimensão, bem como gerar as respetivas chaves de substituição.

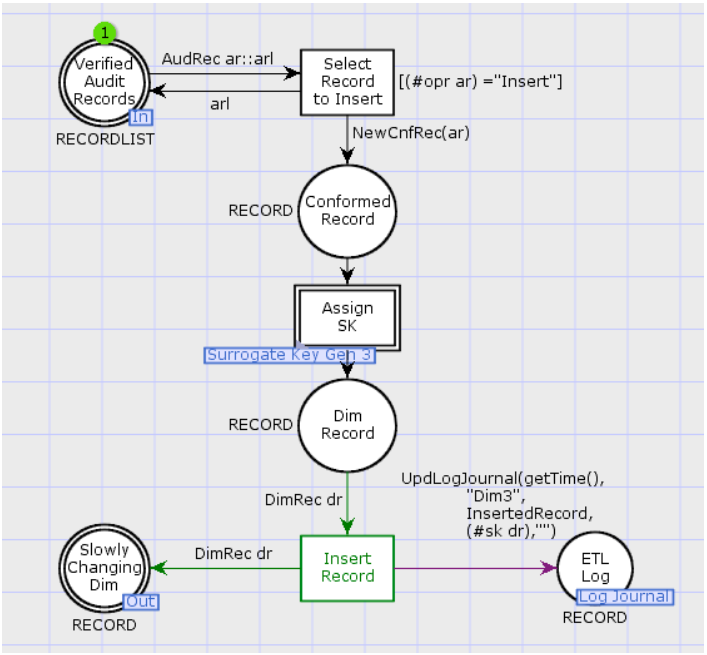


Figura 3.5 - Módulo de inserção dos dados do padrão SCD-H em RPC.

Este módulo é composto por cinco lugares e três transições, sendo uma delas, *Assign SK*, um apontador para outra página. Neste caso, o lugar *Verified Audit Records* é o responsável pela alimentação do módulo com os registos provenientes do módulo anterior (Figura 3.4). Seguindo a ordem de operações deste módulo, os registos contidos no lugar *Verified Audit Records* são submetidos a um processo de seleção. Aqueles cujo o tipo de operação realizada no sistema

operacional seja de inserção, avançam para a próxima etapa. A transição *Select Record to Insert* opera essa mesma seleção. Assim, o próximo passo é fazer a geração da chave de substituição para o registo e atualizar a tabela de *lookup* da dimensão em questão. Este processo tem necessariamente que ocorrer, visto que os registos são novos e, como tal, ainda não possuem uma chave capaz de os identificar no DW. A geração das chaves de substituição é realizada recorrendo ao módulo *Assign SK*, que pode ser consultado em maior detalhe na Figura 3.6. Este módulo faz a recolha do registo e atribuii-lhe uma chave de substituição, que, neste caso, é um número inteiro incrementado por uma unidade, atualiza o lugar *LookupTable* com o registo já com um chave de substituição atribuída e devolve o mesmo para ser devidamente inserido na dimensão. Descortinado o processo de geração de chaves de substituição, o registo é então inserido na dimensão correspondente, representada pelo lugar *Slowly Changing Dim*, e atualizar o lugar *ETL Log* que, tal como foi explicado anteriormente, é o responsável por anotar todas as operações que ocorrem durante o processo ETL de forma a que haja um ponto de retorno caso o processo deixe de executar corretamente, neste caso irá anotar uma operação de inserção de uma registo numa dimensão.

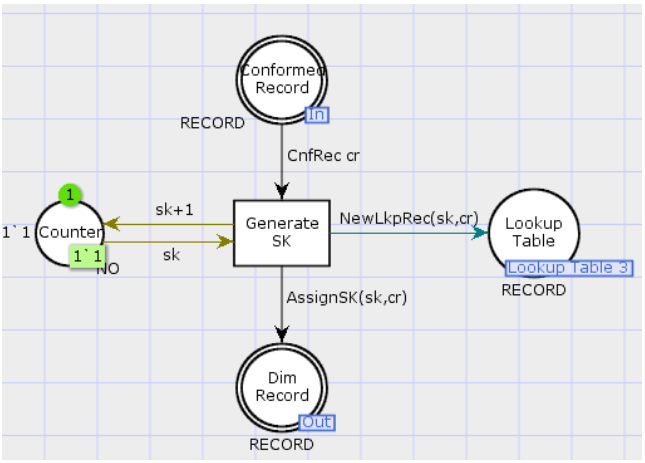


Figura 3.6 - Módulo de geração das chaves de substituição do padrão SCD-H em RPC.

3.4.4 Módulo de Remoção dos Registos

Usualmente, num DW os registos nunca são eliminados, mas sim atualizados de forma a puderem descrever com correção o que ocorreu nos sistemas operacionais. Como tal, o módulo de remoção de registos não elimina fisicamente o registo da dimensão, mas simplesmente atualiza o seu estado, mudando-o de ativo para inativo, podendo ser acedido em eventuais interrogações que um agente de decisão possa fazer ao DW no futuro. A Figura 3.7 reflete o resultado da modelação para este tipo de atividade. Este módulo é composto por cinco lugares e duas transições. Os registos que alimentam a atividade são provenientes do lugar *Verified Audit Records*, tal como aconteceu anteriormente no módulo apresentado na Figura 3.5. Neste caso, ao invés de se selecionar os registos para inserir, seleciona-se os registos com o tipo de operação *Delete*. Os registos são então selecionados e, de seguida, três operações são realizadas, nomeadamente: a procura do registo, a atualização do registo e a atualização do *log* do ETL. A primeira destas operações passa por encontrar o registo proveniente das fontes de dados operacionais na tabela de dimensão, para que o seu estado possa ser alterado. Para isso, é necessário recorrer às tabelas de *lookup*, que guardam o par chave natural, chave de substituição.

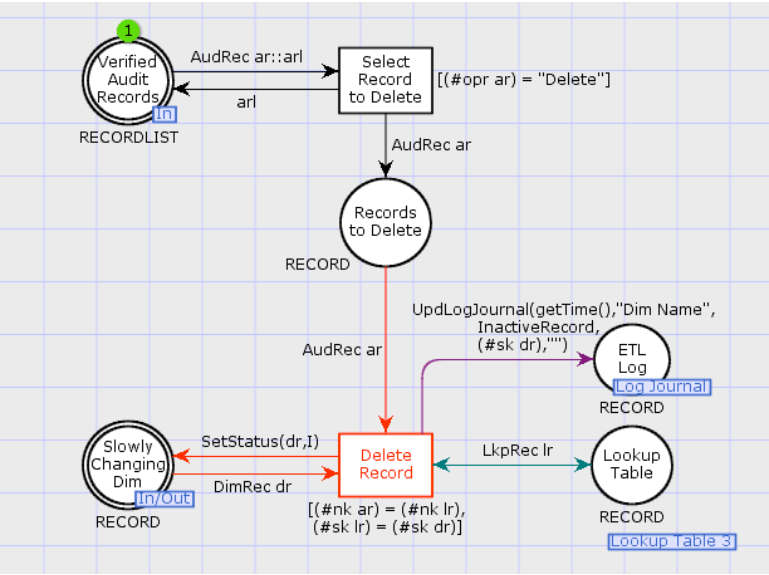


Figura 3.7 - Módulo de remoção dos registos do padrão SCD-H em RPC.

Encontrada a correspondência das chaves envolvidas no processo, procede-se à seleção do registo na dimensão, à atualização do estado do registo selecionado para inativo e, por fim, à atualização do *log*, com mais uma operação realizada no processo de ETL, neste caso de remoção do registo.

3.4.5 Módulo de Atualização dos Registos

O módulo para a atualização dos registos é muito importante no processo de manutenção dos dados históricos no DW. Na Figura 3.8 está apresentado o modelo RPC para este módulo. Como já se explicou anteriormente, também aqui haverá uma tabela adicional na qual serão guardados todos os dados históricos. Assim, o processo acaba por ser semelhante ao apresentado na Figura 3.7, com a diferença de que os registos antigos passarão para a tabela na qual é mantido o histórico dos dados. Este módulo é composto por seis lugares e duas transições.

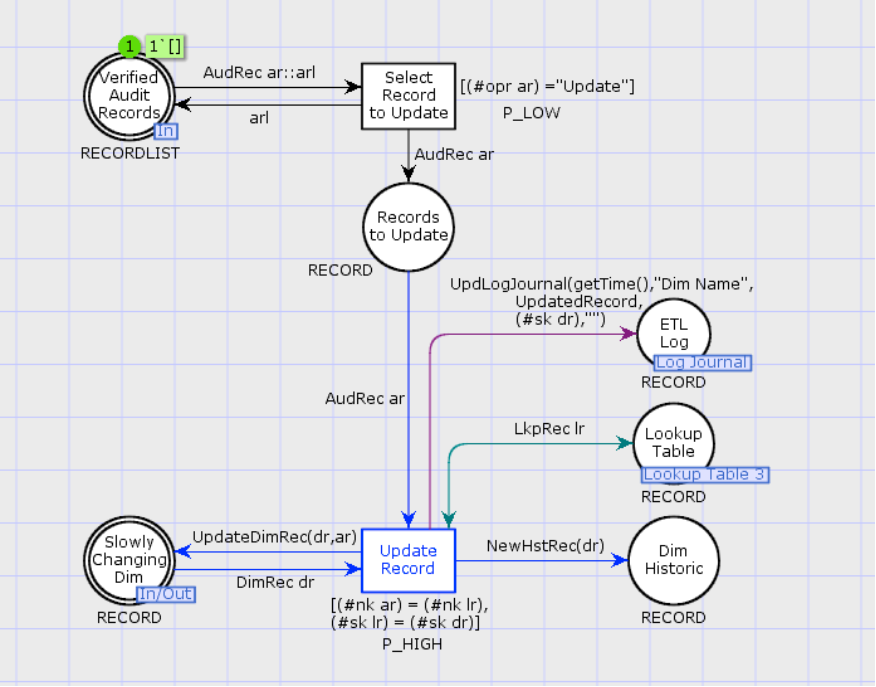


Figura 3.8 - Módulo de atualização dos registos do padrão SCD-H em RPC.

Seguindo a ordem das operações, aparece agora o lugar *Verified Audit Records* que contém os registos prontos a serem selecionados para atualizar a respetiva dimensão. Essa seleção é

realizada pela transição *Select Record to Update*. Tendo o registo selecionado é necessário encontrar a sua correspondência na tabela de dimensão. Essa seleção, tal como no caso do módulo de remoção dos registos (secção 3.4.4) é realizada através da tabela de *lookup*. Encontrado o registo a atualizar, faz-se a sua inserção no lugar *Dim Historic*, para que o valor antigo seja mantido como histórico e o registo seja atualizado na dimensão com o novo valor proveniente da fonte operacional. Uma vez mais, o *log* do sistema de ETL é atualizado com a operação realizada.

3.5 Change Data Capture

O povoamento de um DW é um dos procedimentos mais críticos de um sistema de data warehousing, contribuindo de forma muito significativa para o bom funcionamento do processo de tomada de decisão. Existem duas fases distintas no povoamento de um DW. Uma fase inicial, na qual o DW ainda não possui qualquer tipo de informação, sendo necessário recolher os dados considerados interessantes nas fontes operacionais definidas de forma a poder alimentá-lo, e uma fase regular, que acompanha as alterações dos dados nas fontes operacionais. Todavia, para que nesta fase regular se capte apenas as alterações que foram realizadas após a fase inicial de povoamento é necessário recorrer a processos ETL específicos que são capazes de colecionar essas mesmas mudanças, sejam elas devido a inserções, remoções ou atualizações de dados que ocorreram nas fontes operacionais. Este tipo de processos é usualmente denominado por CDC (*Change Data Capture*) ou captura de dados modificados. Este processo pode ser também considerado um padrão ETL, porque, de uma forma ou de outra, a sua implementação é algo que sempre acontece dado ser processo responsável pela angariação dos dados modificados e, como tal, o responsável por povoar o DW durante a sua vida.

Para implementar um processo CDC existem diferentes estratégias, sendo a sua utilização usualmente definida pela própria natureza do problema. As implementações deste tipo de processo podem ser categorizadas de duas formas: intrusiva e não intrusiva. Uma possível solução passa, por exemplo, pela criação de atributos de controlo nas fontes operacionais. Esta solução implica a adição de atributos de controlo, como o tipo de operação realizada, a etiqueta temporal para preservar a integridade referencial ou a ordem como as operações foram executadas. Apesar de ser uma solução simples, esta solução implica a alteração da estrutura de tabelas que, caso não tenha sido idealizada durante a fase de concepção do sistema operacional, tornaria esta solução

intrusiva. Outra possibilidade para implementar um processo CDC poderia ser o uso de mecanismos automáticos como os *triggers*, que seriam acionados sempre que alguma alteração ocorresse no sistema operacional, quer esta fosse uma inserção, uma remoção ou uma atualização. O registo modificado seria adicionado pelo *trigger* a uma tabela de auditoria, que posteriormente seria coletada pelos mecanismos de CDC integrados no sistema de ETL. É, também, considerada uma solução de carácter intrusivo, apesar de não implicar a reestruturação de qualquer tabela nas fontes operacionais. Porém, e tal como a medida anterior, implicaria que o administrador do SDW tivesse permissões para criar tais mecanismos nas fontes operacionais. Quanto às soluções ditas não intrusivas, estas detetar as mudanças ocorridas nos dados nas fontes operacionais utilizam frequentemente o cálculo da "diferença" entre os registos localizados no sistema operacional e os localizados no DW. Para casos que impliquem a manipulaão de poucos registos a ideia pode ser interessante. Porém, quando o DW e o sistema operacional revelam casos com grandes volumes de dados, esta opção impõe um processo de cálculo bastante moroso, exigente e consumidor da maior parte do tempo de processamento ETL. Mas existem outras alternativas. Uma delas, uma solução não intrusiva, com pouco impacto no sistema operacional e menos exigente em termos de processamento que o cálculo da diferença, passa pela leitura do ficheiro de *log* das transações ocorridas e registadas em estruturas específicas dos sistemas operacionais. O ficheiro de *log* das transações guarda todas as transações utilizadas na modificação do estado do sistemas operacional. No modelo RPC relativo ao padrão CDC foi usado o *log* de transações do SQL Server 2008 - para ajudar na compreensão do modelo RPC, a estrutura do ficheiro *log* será brevemente analisada e explicada de seguida.

Tabela 3–1 Parte da informação presente no log de transações - extraída de (Silva et al., 2013).

LSN	T. ID	Operation	T. Name	End Time	AllocUnitName	RowLog Contents
1	1	LOP_BEGIN_XACT	INSERT	NULL	NULL	NULL
2	1	LOP_INSERT_XACT	NULL	NULL	dbo.TestTable	0x10006C0
3	1	LOP_COMMIT_XACT	NULL	2012/08/03 22:57:05	NULL	NULL
4	2	LOP_BEGIN_XACT	UPDATE	NULL	NULL	NULL
5	2	LOP_MODIFY_XACT	NULL	NULL	dbo.TestTable	0x63
6	2	LOP_MODIFY_XACT	NULL	NULL	dbo.TestTable	0x63
7	2	LOP_COMMIT_XACT	NULL	2012/08/03 22:58:29	NULL	NULL
8	3	LOP_BEGIN_XACT	DELETE	NULL	NULL	NULL

O ficheiro de log das transações do SQL Server possui várias colunas, que na sua maioria contêm valores NULL. Para modelar o padrão CDC foram usadas apenas sete dessas colunas, a saber: *LSN*, *T.ID*, *Operation*, *T.Name*, *End Time*, *AllocUnitName* e *RowLog Contents*. A coluna *LSN* guarda o identificador único para cada registo da transação, a coluna *T.ID* guarda o identificador da transação. A coluna *Operation* para cada transação começa com o valor LOP_BEGIN_XACT e termina com o valor LOP_COMMIT_XACT. Por sua vez, *T.Name* representa o nome da transação e pode ser encontrado no primeiro registo de cada nova transação. Para assinalar o final de cada transação é guardada uma marca temporal na coluna *End Time*. A coluna *AllocUnitName* acolhe o nome do esquema e da tabela na qual a modificação ocorreu, enquanto que a coluna *RowLog Contents* possui os valores modificados em formato hexadecimal (Silva et al., 2013). A Tabela 3–1 apresenta a informação que podemos encontrar no log para a qual o modelo foi desenhado.

3.5.1 Módulo Geral

O modelo RPC para o padrão ETL CDC (Figura 3.9) integra na sua composição três transições e seis lugares. De forma análoga ao padrão ETL SCD-H (secção 3.4.1), as três transições são apontadores para outras páginas, o que permite que o modelo fique mais legível e claro.

A estratégia para a modelação deste padrão assenta em distribuir as diferentes tarefas do padrão pelos diferentes módulos. Temos, então, a tarefa de leitura do ficheiro de *log* de transações, fazendo-se a separação dos diferentes registos de forma a que se possam detetar as várias mudanças ocorridas na fonte. A transição *Read Transaction Log* tem a seu cargo a realização de tal tarefa. Depois, segue-se um processo de descodificação do conteúdo presente no *log,* sendo a transição *Decode RowLog Contents* agora a responsável pela execução do processo. Por último temos a inserção dos registos nas tabelas de auditoria, com o objetivo de serem processados, transformados adequadamente e carregados no DW. Esta tarefa é da responsabilidade da transição *Update Audit Tables*.

Vejamos agora os diversos aspetos do modelo naquilo que diz respeito aos lugares do modelo. O lugar *Transact Log* possui os registos provenientes do *log* de transações. Este lugar irá alimentar todo o processo CDC que irá culminar no lugar *Audit table*, lugar este que guardará todos os registos já descodificados. Os lugares *Opr*, *TLog Record*, *End Time* e *Row* servem como ponte para receber e encaminhar os diferentes dados presentes no *log*.

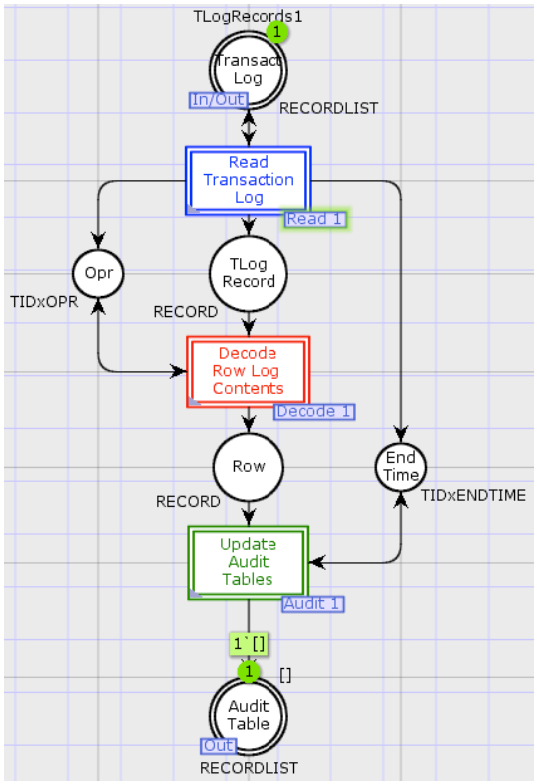


Figura 3.9 - Módulo geral do padrão ETL CDC em RPC.-

3.5.2 Módulo de Leitura

A Figura 3.10 apresenta o módulo de leitura do padrão ETL CDC. Este módulo atua sobre os registos presentes no ficheiro *log* de transações analisado previamente (Tabela 3–1). O módulo é composto por três transições e cinco lugares. Como observado anteriormente, todas as transações começam com a operação do tipo LOP_BEGIN_XACT e terminam com a operação LOP_COMMIT_XACT. O registo com o tipo de operação LOP_BEGIN_XACT possui também o nome da operação a realizar (e.g. INSERT, UPDATE ou DELETE), que no caso de ser LOP_COMMIT_XACT possui uma marca temporal. A separação desses registos é feita recorrendo às transições *Extract Begins* e *Extract Commits*, respetivamente. De forma a manter a integridade de todas as transações, sempre que um registo é do tipo LOP_BEGIN_XACT no lugar *Opr* guarda-se um par composto pelo identificador da transação (*T. ID*) e pelo nome da operação (*T. Name*).

Caso o registo seja do tipo LOP_COMMIT_XACT é guardado no lugar *End Time* um par composto pelo identificador da transação (*T. ID*) e pela marca temporal (*End Time*). A transição *Extract I,U,D* é responsável por selecionar os registos cujo tipo de operação seja LOP_INSERT_ROWS, LOP_UPDATE_ROWS ou LOP_DELETE_ROWS. Os registos que obedecerem à restrição envolvida serão encaminhados para o lugar *TLog Rec* para serem descodificados na próxima tarefa. Como o processo ETL pode, por algum motivo (e.g. falha na energia, problemas de hardware, ou outros), terminar inesperadamente, o lugar *TLog Prog* guarda o valor do atributo *LSN* do *log* de transações, para que o processo possa ser retomado a partir do último elemento processado numa eventual situação de falha.

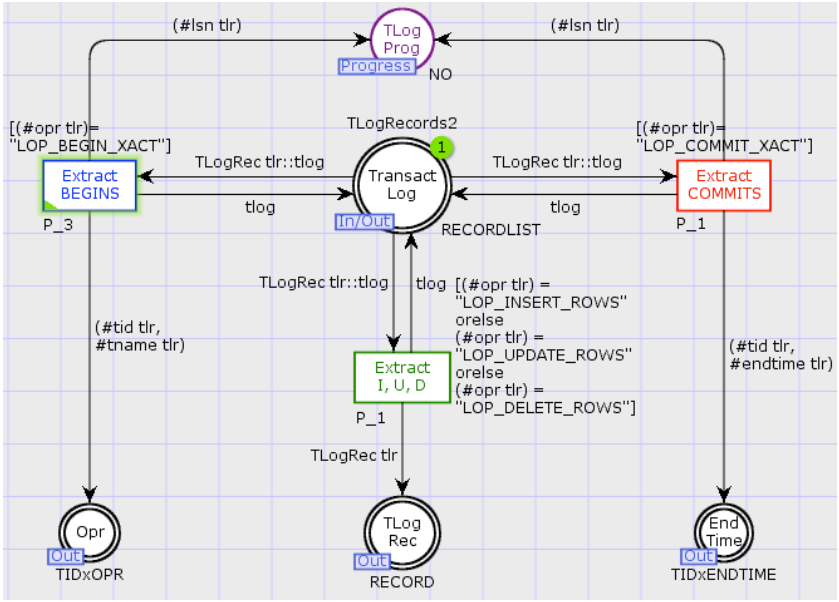


Figura 3.10 - Módulo de leitura do padrão CDC em RPC.

3.5.3 Módulo de Descodificação

O módulo de descodificação tem como objetivo descodificar os valores hexadecimais presentes no log de transações do SQL Server. Aquando da modelação deste módulo, o objetivo primordial foi o de especificar padrões de ETL recorrendo à linguagem de modelação RPC. Como tal, o processo de

descodificação dos valores hexadecimais é uma simulação dessa descodificação. Este módulo integra uma transição e quatro lugares (Figura 3.11).

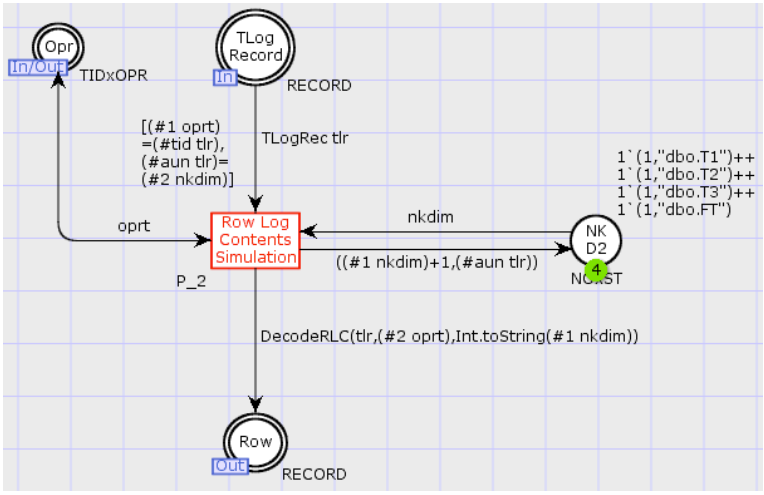


Figura 3.11 - Módulo de descodificação do padrão CDC em RPC.

O lugar *TLog Record* possui os registos do log de transações, que são do tipo LOP_INSERT_XACT, LOP_UPDATE_XACT e LOP_DELETE_XACT. São estes registos que contêm o valor hexadecimal a ser descodificado. O lugar *Opr*, como analisado anteriormente, guarda em si todos os pares identificador da transação, bem como o nome da operação. O lugar *Row* recebe os registos já com o valor hexadecimal descodificado. O lugar, *NK D2* juntamente com a transição *Row Log Contents Simulation,* simula a descodificação da coluna *RowLog Contents* do *log* de transações. Em termos simples a simulação consiste em criar um registo com a informação proveniente do *log* (*LSN*, *T. ID*, *T. Name* e *AllocUnitName*) e com a informação gerada pelo modelo. O modelo é responsável por gerar uma chave natural para o registo, o nome da fonte de dados e o atributo modificado, que neste caso vai tomar o valor da chave natural. De referir que, o propósito deste módulo não é o de descodificar realmente o valor hexadecimal, mas sim mostrar que é possível simular o processo de CDC sem que a atual descodificação do valor interfira no objetivo do módulo.

3.5.4 Módulo de Atualização

O módulo de atualização tem como função acrescentar a marca temporal da transação e adicionar os registos à tabela de auditoria. Na Figura 3.12 está apresentado o módulo de atualização. Este é composto por uma transição e quatro lugares. O lugar *Row* contém os registos que sofreram o processo de descodificação e que agora estão prontos para serem inseridos na tabela de auditoria. O lugar *End Time* possui os pares identificador da transação e marca temporal, os registos deste lugar são provenientes do módulo de leitura (Figura 3.10). O lugar *TLog Prog* guarda o *LSN* do último registo processado. Tal como aconteceu no módulo de leitura, a atualização deste lugar permite, em caso de falha, retomar o processo a partir de um estado anterior. Por fim, o lugar Audit Table é a última paragem dos registos neste processo CDC. Neste lugar os registos já possuem a marca temporal da transação passada através da transição *Update Audit Table*.

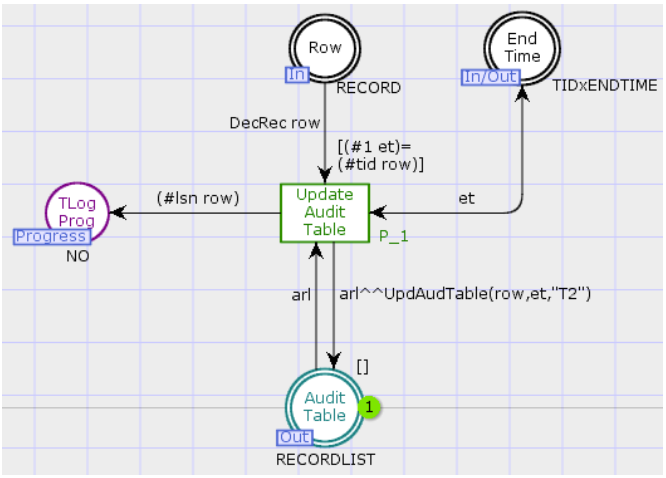


Figura 3.12 - Módulo de atualização do padrão CDC em RPC.

3.6 O Sistema ETL Modelado

Recordando o caso de estudo apresentado na Figura 3.1 e conhecidos os três modelos dos padrões ETL é possível criar um modelo completo para simular e validar o processo de ETL para o

DM em questão. A Figura 3.13 representa esse mesmo modelo, no qual é possível ver as diferentes dimensões que o DM possui, "Data" (*Time*), "Produto" (*Product*) e "Cliente" (*Customer*), e a tabela de factos "Vendas" (*Sales Fact Table*).

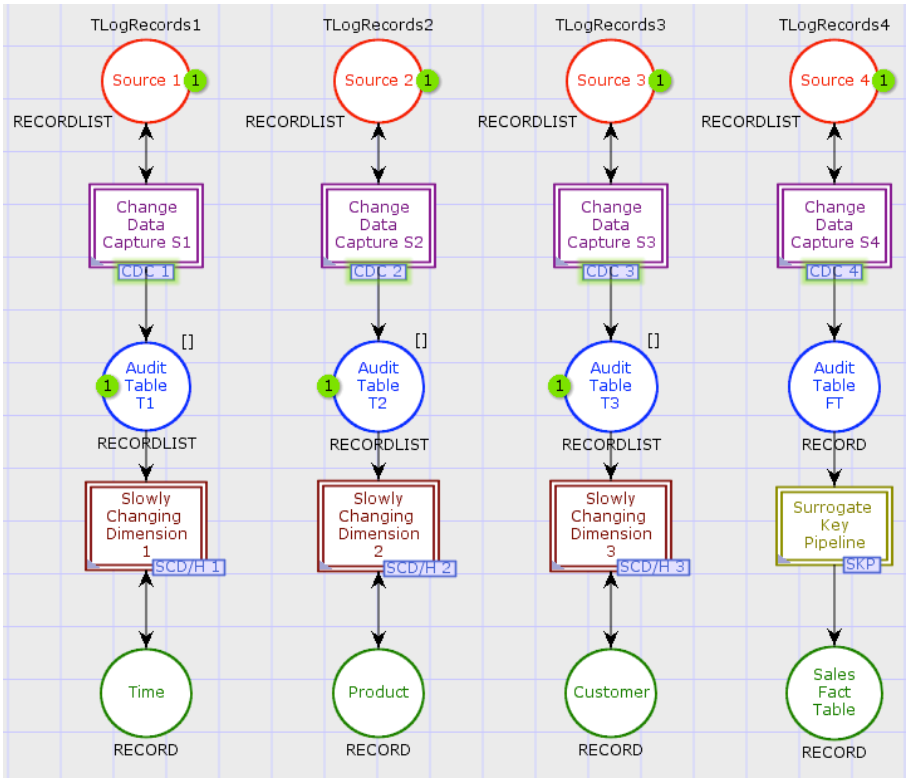


Figura 3.13 - Modelo RPC para o Data Mart de Vendas

Para cada uma das dimensões está associado um módulo CDC e um módulo SCD-H. Apesar de nem todas as dimensões serem de variação lenta, a componente SCD-H não serve só para atualizar os registos, mas também para os inserir e eliminar. O padrão ETL SKP obrigatório para fazer a substituição das chaves naturais.

Apenas utilizando três padrões ETL, modelados e validados usando as RPC, é possível construir um sistema de ETL, totalmente operacional e capaz de simular todo o processo desde a recolha dos dados nas fontes até ao carregamento dos mesmos para as respetivas dimensões e tabela de factos. Isto demonstra que, as RPC são uma ferramenta muito poderosa, diversificada e

com uma excelente capacidade adaptativa, sendo mesmo capazes de modelar sistemas com um certo grau de complexidade, como é o caso de um sistema de ETL. Todavia, ainda existe algum caminho a percorrer, uma vez que é preciso modelar mais padrões para abranger as diversas situações que um sistema de ETL utiliza usualmente.

Capítulo 4

Geração de Esqueletos ETL

4.1 Esqueletos ETL

Através dos três padrões ETL que foram modelados anteriormente com RPC, pudemos comprovar que a modelação é, pois, o primeiro passo que se dá na construção de sistemas ETL capazes e robustos. Modelar conceptualmente os sistemas de ETL torna o processo de construção mais criterioso e sólido, fazendo com que estes fiquem menos susceptíveis a erros, já que, desta forma, é possível simular e validar cada componente modelado, evitando assim alguns dos erros mais vulgares que acontecem durante a execução do sistema. Os padrões aqui descritos são um pequeno passo para a aceitação das RPC como linguagem para a modelação de sistemas de ETL. Como tal, no seguimento de todo o processo de modelação, a automatização do ciclo de desenvolvimento de um sistema de ETL pode trazer, de certa forma, alguns argumentos mais para uma maior aceitação por parte da comunidade de sistemas de ETL. Aliar a modelação conceptual à modelação física pode ser um aspeto muito aliciante para justificar o emprego das RPC como linguagem de modelação de padrões ETL. Um dos problemas que a modelação pode acarretar é o facto de que, em muitos casos, ficamos apenas com um modelo que define, de certa forma, as regras para a construção do sistema. Se existir a hipótese de reutilizar parte do trabalho despendido durante a modelação conceptual, o modelo RPC passaria a ter uma mais-valia em

relação a outras linguagens de modelação. A geração de um esqueleto ETL a partir de um modelo RPC traduz essa mais-valia. Um esqueleto ETL não possui o estatuto de modelo físico executável, para isso, será preciso adicionar certos componentes ou informações extra de configuração que pode não estar necessariamente modelada. Um esqueleto ETL pode ser compreendido como uma base para a construção do sistema, de forma análoga ao esqueleto humano que é constituído por ossos que servem como base para os músculos e os órgãos. É neste princípio que a geração de esqueletos ETL assenta, uma base para a construção de sistemas de ETL.

4.2 Ferramenta de Acolhimento para os Esqueletos ETL

Hoje em dia existe uma enorme variedade de ferramentas especificamente orientadas para a implementação de sistemas de ETL. De mencionar, por exemplo, o *Oracle Warehouse Builder*, o *Microsoft SQL Server Integration Service*, o *DB2 Infosphere Warehouse Edition*, o *SAS Data Integration Studio*, o *PowerCenter Informatica*, o *Talend Studio for Data Integration* ou o *Pentaho Data Integration*, entre outros. Cada uma destas ferramentas apresenta diferentes vantagens e desvantagens na sua utilização em aplicações prática reais. Para acolher os esqueletos gerados a partir das RPC foi escolhida a ferramenta *Pentaho Data Integration* (PDI). O PDI, também conhecido por *Kettle*, é uma ferramenta *open-source* que não requer a aquisição de nenhuma licença para a sua utilização. Além do mais, pode ser executado em diferentes plataformas, tal como Windows, Linux e Mac OSX. Através da sua utilização os ficheiros podem ser importados no formato XML ou, alternativamente, no formato proprietário KJB e KTR. O formato KJB corresponde a um trabalho (*job*), que pode conter várias transformações, assentando basicamente num conjunto de estruturas de controlo de fluxo de elevado nível, enquanto que, o formato KTR corresponde a uma transformação (*transformation*) que é responsável por mover e transformar os diferentes registos, desde a fonte de dados operacionais até ao destino final, o DW. Apesar de existirem este três tipos de formatos, a sua configuração interna é a mesma, isso faz com que, o esqueleto ETL gerado tenha que respeitar as regras de configuração desses formatos para poder ser importado para o ambiente do PDI.

4.3 Processo de Geração de Esqueletos ETL

A fase de desenho conceptual de um esqueleto ETL inicia o ciclo de vida do processo de desenvolvimento de esqueletos ETL (Figura 4.1). Após o desenho e implementação do modelo, no qual as RPC atuam como linguagem de modelação, segue-se a exportação do modelo para o formato XML. O ficheiro XML exportado com o modelo RPC do esqueleto é então submetido à ferramenta para a geração do esqueleto ETL. Concluído o processo de geração do esqueleto ETL, este pode ser visualizado numa ferramenta de ETL, que neste caso corresponde à ferramenta PDI. Com a visualização do esqueleto termina o ciclo de vida do desenvolvimento do esqueleto ETL.

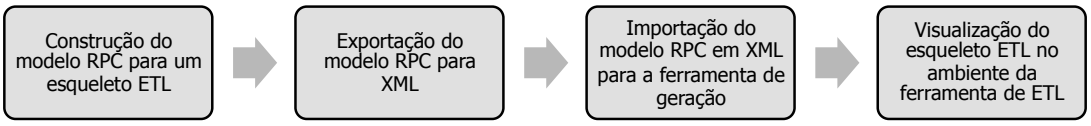


Figura 4.1 - Ciclo de vida do processo de desenvolvimento de esqueletos ETL.

A terceira fase do ciclo de desenvolvimento diz respeito a todo o processo de geração do esqueleto ETL a partir do XML exportado para o modelo RPC. Esta etapa consiste em transformar um modelo RPC num esqueleto ETL, sendo nuclear para o bom funcionamento da ferramenta (Figura 4.2). Para isso se suceder existe uma série de passos a seguir para a geração do esqueleto. De uma forma simples a ordem de execução da ferramenta para a geração de esqueletos ETL é:

1. Processar o ficheiro XML exportado pelo CPN Tools;
2. Detetar os padrões presentes no ficheiro XML;
3. Para cada um dos padrões detetados:
    A. Converter os elementos RPC para componentes ETL;
    B. Ligar os componentes convertidos segundo os arcos presente no modelo RPC;
    C. Exportar para XML o resultado da transformação.

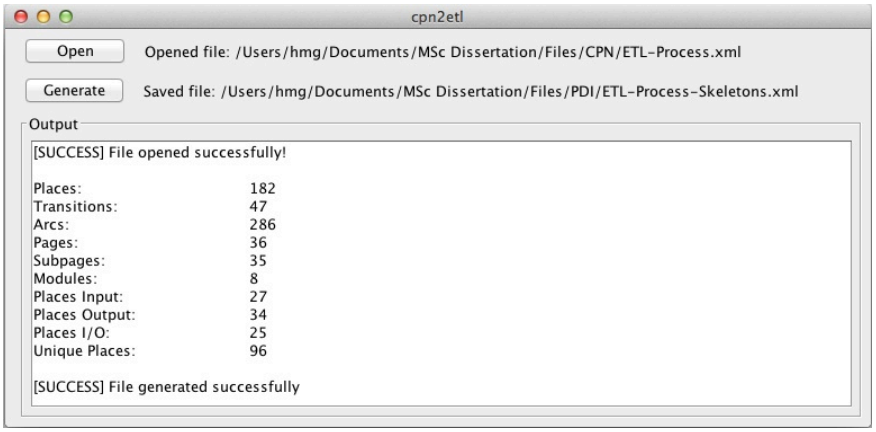


Figura 4.2 - Interface gráfico da ferramenta da geração de esqueletos ETL.

O primeiro passo enunciado já aqui foi explicado. Trata-se da construção do processador de XML para a leitura e interpretação do ficheiro exportado. Com toda a informação relevante para a construção do esqueleto interpretada e devidamente armazenada em estrutura próprias, o próximo passo será o de detetar os padrões ETL que estão presentes no modelo. Essa deteção é feita através do nome da página de substituição. Recordemos a Figura 3.13, e tomemos como exemplo a transição com o texto *Change Data Capture S1*, o nome da página de substituição encontra-se no rectângulo azul por baixo da transição, podendo-se nele ler o texto *CDC 1*. A partir desta informação a ferramenta ETL sabe que irá encontrar naquela transição um padrão ETL CDC, sendo então feita a conversão dos elementos RPC com base no padrão que detetou. A conversão dos elementos RPC para componentes ETL é feita através do tipo do elemento e do nome associado ao elemento. Convertidos os elementos, o próximo passo será transformar os arcos do modelo RPC em ligação dos componentes ETL. Para tal, através da informação acerca dos arcos, contida no ficheiro XML, será construído um grafo orientado replicando-se as ligações. Criado o grafo, este é percorrido em toda a sua extensão para saber quais os componentes que estão interligados, recorrendo-se a um algoritmo capaz de encontrar o caminho mais curto entre dois pontos. Nesta fase inicial da geração de esqueletos ETL o peso da escolha do algoritmo acaba por não ter grande influência já que, nesta altura o mais importante é conseguir gerar os esqueletos ETL a partir do modelo RPC. Numa fase posterior poderá existir a necessidade de fazer outras abordagens que levem a um processo de geração de esqueletos mais eficientemente. Convertidos e interligados os componentes ETL, é altura de percorrer as estruturas que foram guardando a informação e recriar

o XML necessário para que a ferramenta PDI possa importar sem qualquer tipo de problema. Entre esse XML estão uma série de elementos de *log*, as ligações e os componentes ETL. As ligações são representadas pelo elemento XML *order* e os componentes ETL pelo elemento XML *step*. Cada um destes elementos, o *order* e o *step*, tem informações associada à sua subárvore que serão exploradas mais à frente nesta dissertação. O ciclo de vida de um esqueleto ETL fica então completo chegando à quarta fase. Nesta fase o esqueleto ETL, em formato XML, é importado para o ambiente da ferramenta PDI, onde pode ser visualizado e feitas as devidas adições e configurações, de forma a que, seja possível a sua execução.

Nesta altura, é importante reter que se trata ainda de uma primeira fase de geração de esqueletos ETL, não existindo, até à data, e de acordo com o conhecimento que possuímos, nenhuma ferramenta capaz de traduzir um modelo RPC num esqueleto ETL pronto a ser importado e visualizado numa ferramenta de construção de sistemas de ETL. É claro que, uma vez mais, trata-se de um esqueleto, ou seja uma base para a construção do sistema de ETL, um princípio da sua implementação. Nas secções que se seguem serão apresentadas as fases três e quatro do ciclo de vida do processo de desenvolvimento de um esqueleto ETL para os três padrões modelados anteriormente, nomeadamente: SKP, SCH-H e CDC.

4.4 Surrogate Key Pipelining

O resultado final do processo de geração de esqueletos ETL para o padrão ETL SKP (Figura 4.3) consiste numa tradução direta dos componentes do modelo RPC apresentado anteriormente (Figura 3.2).

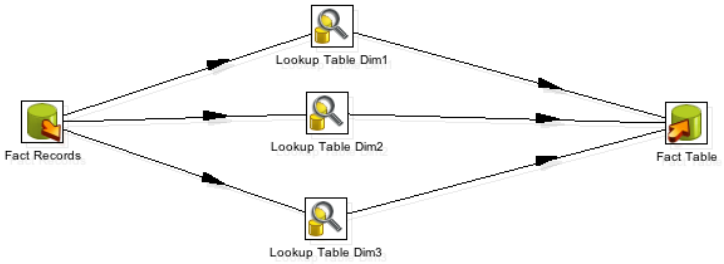


Figura 4.3 - Esquema físico geral do padrão ETL SKP em PDI.

Cada um dos lugares do modelo RPC, com a exceção dos lugares M1 e M2, foram transformados em componentes ETL. Obtivemos, então, uma tabela de entrada de dados (que corresponde ao lugar *Fact Records*), uma tabela de saída de dados (que corresponde ao lugar *Fact Table*) e três tabelas de *lookup* (que correspondem aos lugares *Lookup Table Dim 1-3*). Cada um destes componentes tem uma representação XML.

Como referido anteriormente, cada componente ETL é composto pelo elemento XML *step*, e cada um deles possui um conjunto de elementos XML associados à sua subárvore. Para sabermos o tipo do *step*, este possui um elemento XML denominado por *type*. Cada *type* toma valores diferentes dependendo do componente ETL. Para o caso de uma tabela de entrada de dados, o seu *type* é *TableInput*, para o caso de uma tabela de saída de dados o seu *type* é *TableOutput* e para o caso de ser uma tabela de *lookup* o seu *type* é *DBLookup*. Para além do elemento XML *type* existe o elemento XML *name* que corresponde ao nome do componente ETL. Cada um dos diferentes componentes ETL tem associado diferentes configurações internas, refletindo-se como tal na representação XML desses mesmos componentes. É importante que estas configurações estejam representadas no XML pois, caso isso não aconteça, após o processo de importação do esqueleto para a ferramenta PDI, será impossível aceder às configurações de um componente ETL.

O excerto que se segue foi retirado do ficheiro XML gerado pela ferramenta de transformação de modelos RPC em esqueletos ETL que desenvolvemos e mostra um componente ETL, neste caso uma tabela de entrada de dados denominada por *Fact Records*.

```
<transformation>
  (...)
  <step>
    <name>Fact Records</name>
    <type>TableInput</type>
    (...)
  </step>
  (...)
</transformation>
```

As ligações entre os componentes ETL são representadas pelo elemento XML *order*. Este elemento possui todas as ligações presente no esqueleto ETL. Cada ligação entre dois componentes é representada pelo elemento XML *hop*. Por sua vez, o *hop* possui na sua subárvore os elementos XML *from* e *to*. A identificação do *step* é feita através do seu nome. Assim, uma das restrições para

a construção dos esqueletos ETL é verificar que nenhum *step* possui o mesmo nome, já que a ferramenta não recorre ao uso de identificadores únicos. Segue-se um excerto retirado do mesmo ficheiro XML.

```
<transformation>
  <order>
    <hop>
      <from>Fact Records</from>
      <to>Lookup Table Dim1</to>
    </hop>
    (...)
  </order>
  (...)
</transformation>
```

4.5 Slowly Changing Dimension

O padrão ETL SCD modelado recorrendo as RPC (Figura 3.3) encontra-se dividido por módulos, tal como previamente exposto, e como tal, a transformação do modelo RPC em esqueleto ETL seguiu o mesmo princípio. Os quatro módulos principais – verificação dos dados, inserção de registos, remoção de registos e atualização dos registos – deram origem a quatro componentes ETL responsáveis por executar outras transformações, de forma a que houvesse uma continuidade da estrutura modular apresentada no modelo RPC.

4.5.1 O Módulo Geral

O módulo geral do esqueleto ETL (Figura 4.4) é o resultado do processo de transformação do modelo RPC em esqueleto ETL. Cada uma das quatro transições presentes no modelo RPC deram origem a um componente ETL denominado por *Transformation Executor*, sendo representado ao nível do XML pelo elemento *type* com o texto *TransExecutor*. Este componente ETL é responsável por executar uma transformação ETL. Assim sendo, as transições *Audit Data Verification*, *Delete Record*, *Update Record* e *Insert Record* deram origem a quatros *Transformation Executor* com os respetivos nomes das transições.

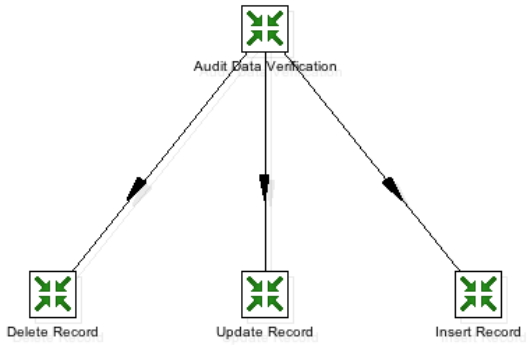


Figura 4.4 - Esquema físico geral do padrão ETL SCD-H em PDI.

O excerto que se segue foi retirado do ficheiro XML gerado durante o processo de transformação do esqueleto ETL e mostra o componente ETL *Transformation Executor*. Além dos elementos XML *type* e *name* outra particularidade deste componente é o elemento XML *filename*, que contém o caminho absoluto para a localização da transformação a executar.

```
<transformation>
  (...)
  <step>
    <name>Update Record</name>
    <type>TransExecutor</type>
    (...)
    <filename>/Users/hmg/Desktop/kettle/Update_3.ktr</filename>
    (...)
   </step>
   (...)
</transformation>
```

4.5.2 O Módulo de Verificação dos Dados

O esqueleto ETL representado na Figura 4.5 diz respeito ao módulo de verificação dos dados previamente modelado usando RPC (Figura 3.4). Este módulo é composto por cinco tabelas de dados, uma delas de entrada – *Audit Records* – e as restantes de saída. Existe também um componente ETL responsável pela verificação dos dados. Cada um dos lugares, com a exceção do lugar *Test*, no módulo RPC deram origem a uma tabela de dados. A única transição presente no modelo originou o componente ETL de verificação dos dados. A representação XML para as tabelas

de dados, quer de entrada quer de saída, foi exposta anteriormente e como tal, não é necessário uma nova explicação. O componente ETL para a verificação dos dados é denominado por *Data Validator* e tem o texto *Validator* associado ao *type* na representação XML.

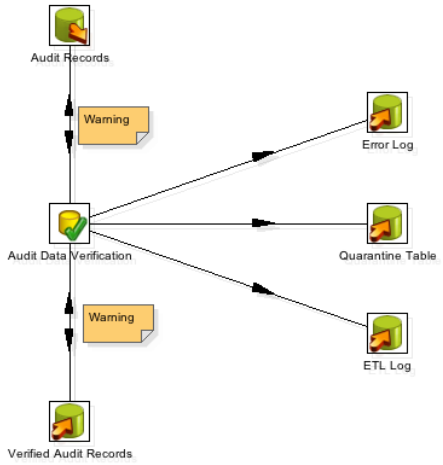


Figura 4.5 - Esquema físico da atividade verificação dos dados em PDI.

Neste esqueleto é possível observar a adição de etiquetas perto das ligações onde exista uma dupla conexão. Se tivermos em conta a Figura 3.4 é possível observar que entre o lugar *Audit Records* e a transição *Audit Data Verification* existe uma dupla ligação e, quando o modelo RPC for convertido num esqueleto ETL irá resultar na dupla ligação dos componentes ETL. De modo a notificar o utilizador que algo não está correto, é colocada perto da ligação um etiqueta como forma de sinalética para o erro. Em grande parte, a colocação desta etiqueta deve-se ao facto do processo de geração de esqueletos ETL ser algo primitivo e bastante rudimentar. Para representar as etiquetas usadas ao nível do XML, existe um elemento denominado por *notepads* que possui na sua subárvore todas as etiquetas presentes no ficheiro XML, e, cada etiqueta é representada pelo elemento XML *notepad* que, na sua subárvore possui os elementos necessários para configurar uma etiqueta. Dentro desses elementos estão, o elemento XML *note* e os elementos XML *xloc* e *yloc*. O elemento XML *note* possui o texto relativo à mensagem a dar, enquanto que os elementos *xloc* e *yloc* tratam do posicionamento da etiqueta. No excerto XML seguinte é possível observar a representação XML para as etiquetas.

```
<transformation>
  (...)
  <notepads>
    <notepad>
      <note>Warning</note>
      <xloc>446</xloc>
      <yloc>329</yloc>
      (...)
    </notepad>
  </notepads>
  (...)
</transformation>
```

Quanto aos aos componente ETL *Data Validator*, podemos de seguida ver um seu exemplo através de um pequeno excerto XML retirado do ficheiro XML resultante do processo de transformação do modelo RPC em esqueleto ETL.

```
<transformation>
  (...)
  <step>
    <name>Audit Data Verification</name>
    <type>Validator</type>
    (...)
  </step>
  (...)
</transformation>
```

4.5.3 O Módulo de Inserção dos Registos

O esqueleto ETL para o módulo de inserção dos registos encontra-se apresentado na Figura 4.6. A geração do esqueleto a partir do modelo RPC (Figura 3.5) resultou em sete componentes ETL. Entre esses componentes encontram-se quatro tabelas de dados, uma delas é a de entrada as restantes as de saída. Existe também um componente ETL de seleção, um componente de sequenciação e um componente de modificação de valores. Apesar do modelo RPC possuir outro módulo, a solução escolhida foi a de incorporar o conteúdo presente no módulo *Assign SK* (Figura 3.6) no esqueleto ETL gerado, o que fez surgir os componentes de sequenciação e de modificações de valores, que são a tradução do lugar *Counter* e da transição *Generate SK*, respetivamente. Em relação às tabelas de dados a sua representação XML é conhecida e, como tal, não será necessário uma nova explicação. O componente de seleção é denominado por *Switch / Case* e para identificá-lo ao nível do XML utilizámos o texto *SwitchCase* ao elemento *type*. O componente de sequenciação numérica é denominado por *Add sequence* e para representá-lo

utiliza-se o texto *Sequence* associado ao elemento XML type. Por último surge o componente de modificação de valores denominado por *Set field value* que, para representá-lo é necessário associar ao elemento XML *type* o texto *SetValueField*. Tal como no esqueleto anterior (Figura 4.5), neste esqueleto também existe a adição de etiquetas junto das duplas ligações entre componentes ETL, para sinalizar que existiu uma falha durante o processo de geração de esqueletos ETL.

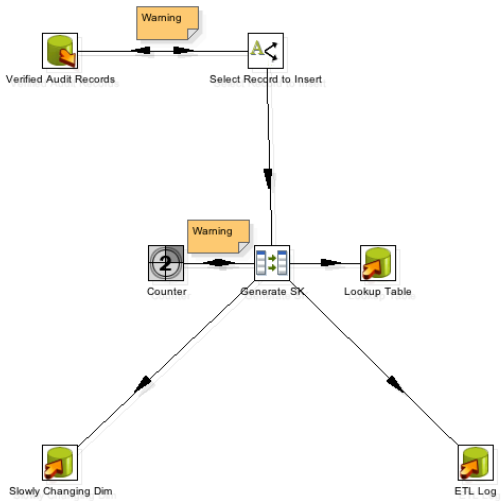


Figura 4.6 - Esquema físico da atividade inserção dos registos me PDI

O excerto que se apresenta de seguida revela os componentes de sequenciação e de modificação de valores, retirados do ficheiro XML gerado durante o processo de construção dos esqueletos ETL.

```
<transformation>
  (...)
  <step>
    <name>Counter</name>
    <type>Sequence</type>
    (...)
  </step>
  (...)
  <step>
    <name>Generate SK</name>
    <type>SetValueField</type>
    (...)
  </step>
  (...)
</transformation>
```

4.5.4 O Módulo de Remoção dos Registos

A Figura 4.7 representa o resultado final do processo de geração de esqueletos ETL para o módulo de remoção dos registos. O modelo RPC deste módulo depois de deviamente transformado deu origem a cinco componentes ETL: três tabelas de dados, um componente de seleção e uma tabela de *lookup*. As tabelas de dados correspondem aos lugares *Verified Audit Records*, *ETL Log* e *Slowly Changing Dim*. O componente de seleção resultou da tradução direta da transição *Select Record to Delete* e o lugar *Lookup Table* originou a tabela de *lookup*. A representação XML para cada um dos componentes ETL gerados é conhecida e, como tal, não existe a necessidade de os explicar novamente. No esqueleto gerado para este módulo ocorreu também a adição de uma etiqueta junto da dupla ligação entre componentes ETL para sinalizar que existiu uma falha durante o processo de geração de esqueletos ETL.

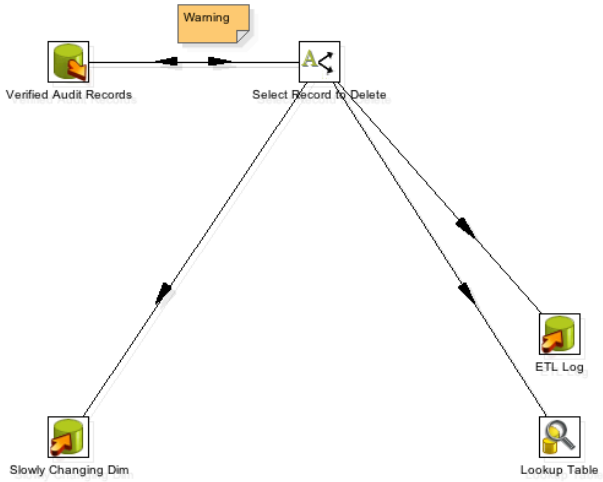


Figura 4.7 - Esquema físico da atividade remoção dos registos em PDI.

A representação XML da tabela de *lookup*, retirada do ficheiro XML gerado durante o processo de construção dos esqueletos ETL, pode ser consultada no excerto de código XML apresentado de seguida.

```
<transformation>
  (...)
  <step>
    <name>Lookup Table</name>
    <type>DBLookup</type>
    (...)
  </step>
  (...)
</transformation>
```

4.5.5 O Módulo de Atualização dos Registos

O módulo de atualização dos registos representado na Figura 4.8 diz respeito ao esqueleto ETL gerado a partir do modelo RPC (Figura 3.8). O módulo é representado por seis componentes ETL, sendo que três delas são tabelas de dados, uma componente de seleção e uma tabela de *lookup*. Todos os lugares do modelo RPC, com a exceção do lugar *Records to Update,* foram traduzidos em

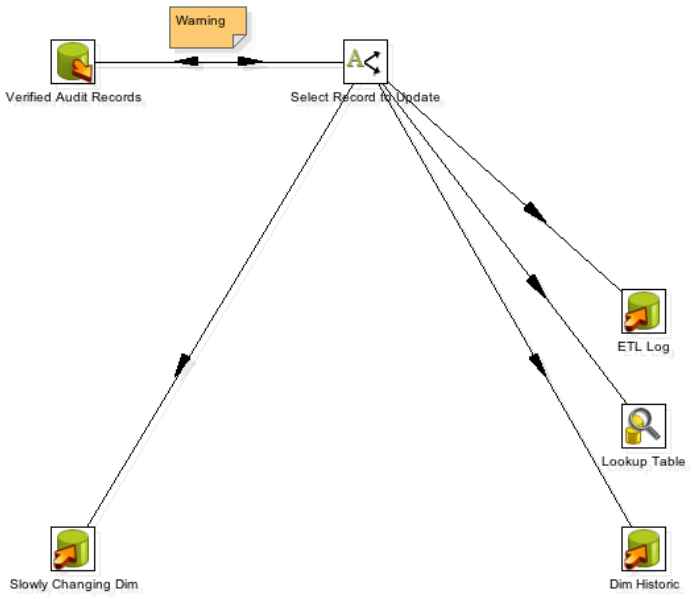


Figura 4.8 - Esquema físico da atividade atualização dos registos em PDI.

tabelas de dados e uma tabela de *lookup* (lugar *Lookup Table*). A transição *Select Records to Update* originou o componente ETL de seleção. Neste módulo é também notória a presença da etiqueta como sinalética para a falha ocorrida durante o processo de geração de esqueletos. A representação XML dos componentes de ETL presentes no esquema foi previamente demonstrada e, como tal, não será necessária uma nova explicação sobre os seus componentes. Também, de seguida, apenas será apresentado um pequeno excerto relativo ao componente de seleção retirado do ficheiro XML gerado pela ferramenta.

```
<transformation>
  (...)
  <step>
    <name>Select Records to Update</name>
    <type>SwitchCase</type>
    (...)
  </step>
  (...)
</transformation>
```

4.6 Change Data Capture

O padrão ETL CDC está dividido em vários módulos, e de uma forma análoga ao padrão ETL SCD-H, a transformação do modelo RPC num esqueleto ETL acompanhou o mesmo princípio que no padrão SCD-H. Cada um dos três módulos presentes no modelo RPC do padrão ETL CDC – *Read Transaction Log*, *Decode Row Log Contents* e *Update Audit Tables* – originaram três componentes ETL de execução de transformações.

4.6.1 Módulo Geral

A Figura 4.9 apresenta o resultado final da transformação do modelo RPC em esqueleto ETL para o padrão ETL CDC. O esqueleto é composto por três componentes ETL de execução de transformações. A representação XML para o componente ETL em questão, o *Transformation Executor*, que foi previamente apresentada, tal como a apresentação de um excerto referente à representação XML do componente, visto que apenas o texto relativo aos elementos XML *name* e *filename* iriam sofrer alterações.

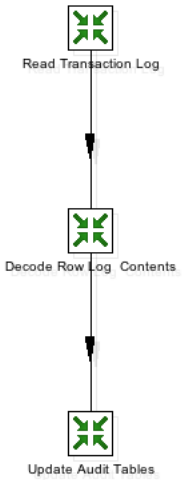


Figura 4.9 - Esquema físico geral do padrão ETL CDC em PDI.

4.6.2 Módulo de Leitura

O esqueleto ETL para o módulo de leitura presente na Figura 4.10 representa o resultado final do processo de geração de esqueletos a partir das RPC (Figura 3.10). Todos os elementos presentes no modelo RPC, quer sejam transições ou lugares, deram origem a componentes ETL. O esqueleto ETL é, então, composto por oito componentes ETL. Os lugares no modelo RPC resultaram em tabelas de dados, ao passo que as transições resultaram em componentes de filtragem. A representação XML das tabelas de dados encontra-se exposta e como tal, não será necessária uma nova explicação. Para o caso do componente ETL de filtragem a sua representação XML não é conhecida. Este componente é denominado por *Filter Rows*. Ao nível do XML para identificar este componente é necessário associar o texto *FilterRows* ao elemento XML *type*. Uma vez mais surgiram etiquetas, assinalando uma falha durante o processo de geração de esqueletos ETL.

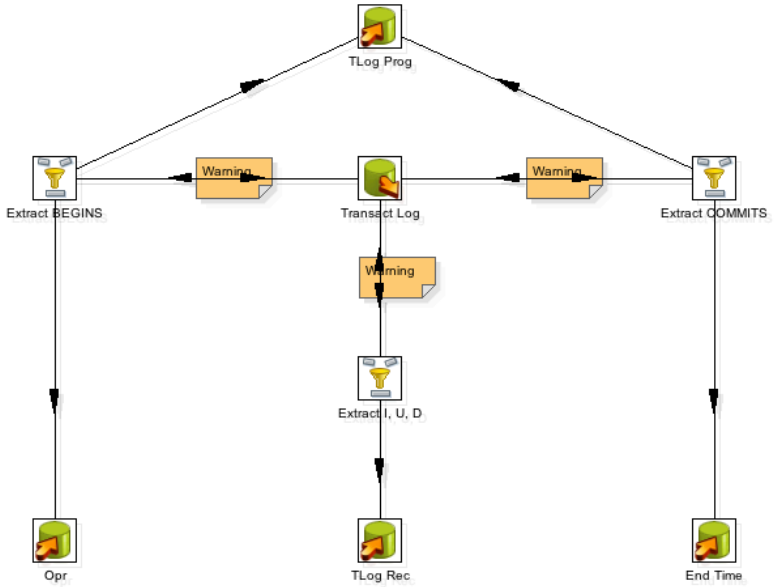


Figura 4.10 - Esquema físico da atividade de leitura em PDI.

A representação XML do componente de filtragem pode ser consultado no excerto que se segue, retirado do ficheiro XML gerado durante o processo de transformação do modelo RPC em esqueleto ETL.

```
<transformation>
  (...)
  <step>
    <name>Extract I, U, D</name>
    <type>FilterRows</type>
    (...)
  </step>
  (...)
</transformation>
```

4.6.3 Módulo de Descodificação

O esquema final resultante do processo de geração de esqueletos ETL pode ser consultado na Figura 4.11. Relativamente a este módulo do modelo RPC para o padrão ETL CDC (Figura 3.11) todos os elementos, com a exceção do lugar *NK D2* foram transformados em componentes ETL. Os

três lugares *Opr*, *TLog Record* e *Row*, deram origem a tabelas de dados. A transição *Row Log Contents Simulation* resultou num componente ETL de execução de *scripts* SQL. Como o objetivo deste módulo era o de simular o processo de descodificação (secção 3.5.3), o componente ETL mais adequado a gerar para a transição *Row Log Contents Simulation* acaba por ser a componente capaz de executar *scripts* SQL. A representação XML para as tabelas de dados foi previamente apresentada, portanto não será necessário uma nova explicação. Relativamente ao componente de execução de *scripts* SQL, para que seja possível identificá-lo ao nível do XML é necessário associar ao elemento XML *type* o texto *ExecSQL*.

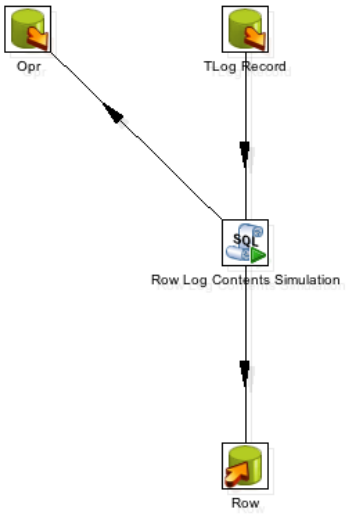


Figura 4.11 – Esquema físico da atividade de descodificação em PDI.

O excerto que se segue foi retirado do ficheiro XML gerado durante o processo de transformação do modelo RPC em esqueleto ETL e diz respeito ao componente ETL de execução de *scripts* SQL.

```
<transformation>
  (...)
  <step>
    <name>Row Log Contents Simulation</name>
    <type>ExecSQL</type>
    (...)
  </step>
  (...)
</transformation>
```

4.6.4 O Módulo de Atualização

O módulo de atualização do modelo RPC para o padrão ETL CDC teve como resultado final o esqueleto ETL representado na Figura 4.12. Relativamente a este módulo, todos os elemento RPC presentes no modelo tiveram uma tradução para componentes ETL. Os lugares *Row*, *End Time*, *TLog Prog* e *Audit Table*, deram origem a tabelas de dados e à transição *Update Audit Tables*, que originou um componente ETL de execução de *scripts* SQL. Os componentes ETL usados na construção deste esqueleto têm a sua representação XML conhecida e como tal, não será necessário voltar a explicar os mesmos.

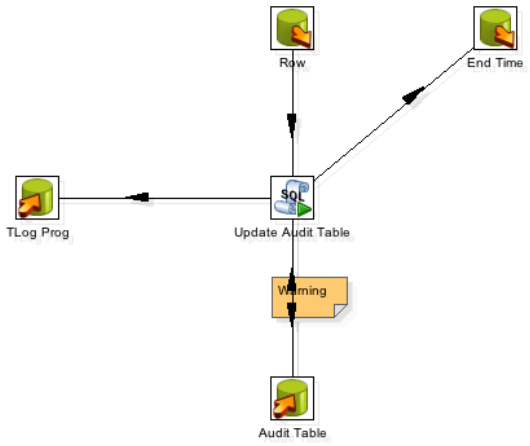


Figura 4.12 – Esquema físico da atividade de atualização em PDI.

O excerto de código XML que se segue diz respeito a um componente ETL de dados, neste caso, uma tabela de dados de saída.

```
<transformation>
  (...)
  <step>
    <name>Audit Table</name>
    <type>TableOutput</type>
    (...)
  </step>
  (...)
</transformation>
```

4.7 Análise dos Resultados

Apesar de parecer que todo trabalho realizado apenas se resume a uma série de cliques para gerar um esqueleto ETL, mais uma operação de exportação de um modelo e uma outra de importação, isso é claramente uma má interpretação, porque não toma em consideração o processo de desenvolvimento de software que foi realizado. Desde o processamento do ficheiro no formato XML exportado pelo CPN Tools até à construção do ficheiro para o esqueleto gerado, existe uma sequência de métodos e de pequenas transformações que foram sendo utilizados. Além disso, foi aplicado um conjunto muito vasto de verificações e de validações necessárias para garantir que o esqueleto ETL a gerar (ou gerado) correspondesse minimamente ao modelo RPC definido para um determinado padrão ETL. Por exemplo, para que fosse possível traduzir as coordenadas cartesianas da ferramenta CPN Tools nas da ferramenta PDI, de modo a que os componentes ETL estivessem posicionados no espaço de igual forma que os elementos do modelo RPC, foi necessário implementar uma série de transformações. Isto são apenas pormenores escondidos numa ferramenta de geração de esqueletos ETL.

Relativamente aos esqueletos ETL gerados, este são, ainda, o resultado de um processo inicial de tradução direta dos elementos dos modelos em RPC em componentes ETL. Contudo, não é de menosprezar este tipo de tradução rudimentar e primitiva, já que todo este processo de transformação acaba por ser uma mais valia no reaproveitamento de uma fase de conceptualização do sistema de ETL, que tantas vezes, por si só, é colocada de lado em detrimento de uma implementação *ad-hoc*. Mesmo assim, apesar de ser um processo inicial, acaba por ser também um passo importante para o desenvolvimento de um tradutor capaz de dar outro significado à modelação conceptual de sistemas de povoamento de *data warehouses*.

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Conclusões

O processo de construção de sistemas de ETL deve contemplar uma fase de conceptualização do sistema. Apesar de, na maior parte das vezes, a implementação dos sistemas de ETL ser feita de uma forma *ad-hoc*, a fase conceptual acaba por trazer vantagens em relação a uma implementação desse tipo. É durante a fase conceptual que são realizados os primeiros esboços do sistema, contendo os requisitos operacionais e funcionais mais essenciais que o sistema de ETL precisa de satisfazer. É também uma primeira abordagem aquilo que será o sistema a implementar.

Diferentes linguagens com resultados práticos na modelação de padrões ETL começam a surgir e a ganhar alguma aceitação na comunidade ETL. As RPC surgem como uma adaptação à necessidade de modelar conceptualmente os sistemas de ETL. A capacidade de simular e validar o modelo RPC é um enorme atrativo para o uso das RPC como linguagem para a modelação conceptual de sistemas de ETL, sendo prova disso os padrões ETL aqui apresentados. O uso de RPC permitiu modelar sistemas de ETL com um nível de detalhe superior. É esse nível de detalhe que torna possível traduzir diretamente o modelo conceptual num modelo físico, com algumas limitações, mas contudo passível de ser executado. O facto dessa tradução ser possível, é uma

prova que é possível gerar esqueletos ETL a partir das especificações presentes no modelo RPC. Trata-se de uma fase pioneira de geração de esqueletos ETL, e como tal, o processo de geração ainda produz estruturas primitivas e rudimentares. É um passo importante para a automatização do processo de desenvolvimento de um sistema de ETL, mas ainda existe a necessidade de adicionar e configurar componentes e tarefas, que neste caso serão da responsabilidade do utilizador do sistema de ETL.

O facto de que o modelo RPC desenhado, modelado e validado durante a fase de conceptualização do sistema ETL possa, ser reutilizado para gerar um modelo físico pronto a ser acolhido por um ferramenta de construção de sistemas de ETL é, uma mais valia para a agilização e automatização do processo de construção de sistemas ETL, sendo uma clara alternativa às implementações de sistemas ETL de uma forma *ad-hoc*, que são menos robustas e mais susceptíveis a erros, levando a que por vezes, o tempo despendido a resolver falhas no processo ETL seja francamente maior do que aquele despendido numa etapa conceptual.

5.2 Trabalho Futuro

O resultado final produzido, levou à criação de esqueletos ETL a partir de modelos em RPC para serem acolhidos pela ferramenta de ETL. Apesar de tudo, ainda existe um longo caminho a percorrer para que o nível de detalhe presente no modelo físico seja um pouco mais relevante do que um simples esqueleto, o que tornaria que o esqueleto deixasse de ser rudimentar e primitivo, bem como a necessidade do utilizador do sistema de ETL adicionar e configurar componentes fosse cada vez menor. Para além disso, é necessário alargar o número padrões ETL modelados com as RPC, para que se possa abranger as especificações dos mais variados sistemas de povoamento de DW. Apesar de existir o conceito de padrão ETL, não existe uma maneira única para a implementação do mesmo, como tal, é necessário também aumentar o número de instanciações que um determinado padrão ETL pode ter. Contudo, o objetivo final passa pela construção de uma ferramenta capaz de suportar todo o ciclo de vida de um processo de desenvolvimento de um sistemas de ETL baseado em padrões.

Bibliografia

Vassiliadis, P., Simitsis, A. & Skiadopoulos, S., 2002. "Conceptual modeling for ETL processes". In Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP. New York, NY, USA, 2002. ACM.

Abelló, A., Samos, J. & Saltor, F., 2006. YAM2: a multidimensional conceptual model extending UML. *Information Systems*, vol. 31(6), pp.541-67.

Akkaoui, Z.E. & Zimanyi, E., 2009. "Defining ETL Workflows Using BPMN and BPEL". In Proceedings of the ACM Twelfth International Workshop on Data Warehousing and OLAP. New York, NY, USA, 2009. ACM.

Akkaoui, Z.E., Zimànyi, E., Mazón, J.-N. & Trujillo, J., n.d. "A model-driven framework for ETL process development". In Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP. New York, NY, USA ACM.

Billington, J. et al., 2003. "The Petri net markup language: concepts, technology, and tools". In Proceedings of the 24th international conference on Applications and theory of Petri nets. Eindhoven, The Netherlands, 2003. Springer-Verlag Berlin, Heidelberg.

Bray, T. et al., 2006. *Extensible Markup Language (XML) 1.1 (Second Edition)*. Technical. W3C - World Wide Web Consortium.

English, L.P., 1999. *Improving Data Warehouse and Business Information Quality: Methods for Reducing Costs and Increasing Profits*. New York: John Wiley & Sons, Inc.

Goldfarb, C., 1991. *The SGML handbook*. New York, NY, USA: Oxford University Press, Inc.

Hillah, L.M., Kordon, F., Petrucci, L. & Trèves, N., 2010. "PNML framework: an extendable reference implementation of the petri net markup language". In Proceedings of the 31st international conference on Applications and Theory of Petri Nets. Braga, Portugal, 2010. Springer-Verlag Berlin, Heidelberg.

Jensen, K., Kristensen, L.M. & Wells, L., 2007. Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer (STTT)*, Volume 9(3), pp.213-54.

Billington, J. & Han, B., 2007. Modelling and analysing the functional behaviour of TCP's connection management procedures. *International Journal on Software Tools for Technology Transfer (STTT)*, Volume 9(3), pp.269-304.

Kimball, R. & Caserta, J., 2004. *The Data Warehouse ETL Toolkit: Pratical Techniques for Extracting, Cleaning, Conforming and Delivering Data*. Indianapolis: Wiley Publishing, Inc.

Mitchell, B., Kristensen, L.M. & Zhang, L., 2007. Formal Specification and State Space Analysis of an Operational Planning Process. *International Journal on Software Tools for Technology Transfer (STTT)*, Volume 9(3), pp.255-67.

Moncelet, G. et al., 1998. Analysing a mechatronic system with coloured Petri nets. *International Journal on Software Tools for Technology Transfer (STTT)*, Volume 2(2), pp.160-67.

Oliveira, B. & Belo, O., 2012. "BPMN Patterns for ETL Conceptual Modelling and Validation". In Proceedings of the 20th International Symposium, ISMIS. Macau, China, 2012. Springer Berlin Heidelberg.

Oliveira, B. & Belo, O., 2013. "Using REO on ETL conceptual modelling: a first approach". In Proceedings of the sixteenth international workshop on Data warehousing and OLAP. New York, NY, USA, 2013. ACM.

Oliveira, B. & Belo, O., 2013. "ETL Standard Processes Modelling - A Novel BPMN Approach". In Proceedings of 5th International Conference on Enterprise Information Systems (ICEIS). Angers, France, 2013.

Oliveira, B. & Belo, O., 2014. "ETL Patterns on YAWL - Towards to the specification of platform-independent data warehousing populating processes". In Proceedings of the 16th International Conference on Enterprise Information Systems (ICEIS). Lisboa, Portugal, 2014.

Silva, D., Belo, O. & Fernandes, J.M., 2012. "Colored Petri Nets in the Simulation of ETL Standard Tasks". In Proceedings of 26th European Simulation and Modelling Conference. FOM, Essen, Germany, 2012.

Silva, D., Belo, O. & Fernandes, J.M., 2013. "Assisting Data Warehousing Populating Processes Design Through Simulation Using Coloured Petri Nets." In Proceedings of the 3rd Industrial Conference on Simulation and Modeling Methodologies, Technologies and Applications. Reykjavik, Iceland, 2013.

Simitsis, A. & Vassiliadis, P., 2003. "A Methodology for the Conceptual Modeling of ETL Processes". In CAiSE Workshops. Klagenfurt/Velden, Austria, 2003.

Simitsis, A., 2003. "Modeling and managing ETL Processes". In VLBD PhD Workshop. Berlin, Germany, 2003.

Trujillo, J. & Luján-Mora, S., 2003. A UML Based Approach for Modeling ETL Processes in Data Warehouses. In *Conceptual Modeling - ER 2003*. Chicago, IL, USA: Springer Berlin Heidelberg., vol. 2813, pp.307-20.

Referências WWW

Industrial Use, 2013. Department of Computer Science, Aarhus University. [Online] Available at: <http://cs.au.dk/cpnets/industrial-use/> [Accessed: outubro 2013]

BPMN, 2012. Object Management Group Business Process Model and Notation. [Online] Available at: <http://www.bpmn.org/> [Accessed: abril 2014].

Reo, 2008. Reo Coordination Language. [Online] Available at: <http://reo.project.cwi.nl/reo/> [Accessed: abril 2014].

YAWL, 2014. YAWL. [Online] Available at: <http://www.yawlfoundation.org/> [Accessed: abril 2014]

CPN Tools, 2014. CPN Tools Homepage. [Online] Available at: <http://cpntools.org/> [Accessed: outubro 2013].

DB2 Infosphere Warehouse Edition, 2014. InfoSphere Warehouse family. [Online] Available at: <http://www-01.ibm.com/software/data/db2/warehouse-editions/> [Accessed: abril 2014].

Microsoft SQL Server Integration Services, 2014. Explore SQL Server 2012-2014 | Microsoft. [Online] Available at: <http://www.microsoft.com/en-us/server-cloud/products/sql-server/> [Accessed: abril 2014].

Oracle Warehouse Builder, 2014. Warehouse Builder 11gR2: Home Page on OTN. [Online] Available at: <http://www.oracle.com/technetwork/developer-tools/warehouse/overview/introduction/index.html> [Accessed: abril 2014].

Pentaho Data Integration, 2014. [Online] Available at: <http://www.pentaho.com/product/data-integration> [Accessed: abril 2014].

PowerCenter Informatica, 2014. [Online] Available at: <http://www.informatica.com/us/products/data-integration/enterprise/powercenter/> [Accessed: abril 2014].

SAS Data Integration Studio, 2014. [Online] Available at: <http://support.sas.com/software/products/etls/> [Accessed: abril 2014].

Talend Studio for Data Integration, 2014. [Online] Available at: <http://www.talend.com/products/data-integration> [Accessed on 28 April 2014].