METODOLOGIAS E ESTRATÉGIAS DE MIGRAÇÃO DE DADOS

Cleber Silva de Oliveira [1]
Marcio Abud Marcelino [2]

Este trabalho identifica metodologias de manipulação de dados, descritas por vários pesquisadores e profissionais em diversos países, realizando um levantamento das metodologias gerais para migração, atualização ou integração de sistemas legados, especificamente, Sistemas Integrados de Gestão, conhecidos ERPs (Enterprise Resource Planning). O trabalho apoia uma direção em metodologias para construção de ferramentas que automatizem os processos de migração, atualização ou integração de dados, evitando falhas humanas na construção dos códigos e rotinas responsáveis por essa operação. O objetivo deste trabalho é manter a integridade dos dados, minimizando custos e consequências, uma vez que, geralmente, os dados de um sistema legado é o patrimônio mais importante de uma organização, onde o fluxo intenso de transações é controlado por meio digital. Por meio de levantamentos bibliográficos, são comparadas três metodologias de migração de dados e identificadas suas principais características, apontando-se qual destas metodologias apresenta uma melhor abordagem para uma migração de dados mais segura.

Palavras-chave: Migração de dados. Sistemas legados. ERP. ETL. **Butterfly Methodology. Chicken Little.**

This paper identifies data manipulation methodologies, described by many researchers and practitioners in several countries, performing a survey of the general methodologies for migration, upgrade or integration of legacy systems, specifically, Integrated Management Systems, known as ERP (Enterprise Resource Planning). The work supports a direction on methodologies for building tools to automate the processes of migration, upgrade or data integration, avoiding human errors in building codes and routines responsible for this operation. The aim of this work is to maintain data integrity by minimizing costs and consequences, since, generally, data from a legacy system are the most important assets of an organization, where the heavy flow of transactions are controlled by digital means. Through literature surveys, three methods of data migration are compared and its main characteristics identified, indicating which of these methodologies provides a better approach to a safer data migration.

Keywords: Data Migration. Legacy Systems. ERP. ETL. Butterfly Methodology. Chicken Little.

1 INTRODUÇÃO

Em muitas indústrias, comércios, instituições públicas ou privadas existem sistemas implantados e funcionais que trazem consigo todos os dados históricos e atuais daquela instituição, tornando-a altamente dependente dos dados contidos em seus sistemas, os quais são comumente chamados de "Sistemas Legados". Esses sistemas de informação são normalmente responsáveis por controlar e integrar todos os processos de uma instituição, conhecidos normalmente como ERP (*Enterprise Resource Planning)*

1 Mestrando pela Faculdade de Engenharia Mecânica – Universidade de Taubaté - Professor da Área de Informática do Instituto Federal de Educação Ciência e Tecnologia de São Paulo – *Campus* Guarulhos. E-mail: <cleber@ifsp.edu.br>.
2 Doutor em Engenharia Eletrônica e Computação pelo Instituto Tecnológico de Aeronáutica - Professor da Universidade de Taubaté e da UNESP. E-mail: <abud@feg.unesp.br>.
Data de entrega dos originais à redação em 23/11/2011 e aceito para diagramação em 25/10/2012.

ou Sistemas Integrados de Gestão. Passaram a ser largamente utilizados no início da década de noventa e são cada vez mais indispensáveis dentro das instituições. Tendo, como característica fundamental, o fato de serem constituídos por pacotes, com módulos integrados e interligados, em tempo real, que utilizam um único banco de dados, os sistemas ERP objetivam dar suporte à maioria das operações de uma empresa (VALENTE, 2004).

Alguns dos princípios mais importantes na qualidade de *software*, segundo a NBR ISO/IEC9126-1 (ABNT, 2003), é a manutenção, que representa a capacidade de um *software* ou sistema ser modificado para adição de melhorias ou funcionalidades, para correção de erros e para portabilidade, que é a capacidade do *software* ser transferido de ambiente ou tecnologia; porém muitos sistemas não foram desenvolvidos observando esses critérios e, por falta de documentação, por descontinuidade de suas tecnologias ou por modificação de *hardware*, estes sistemas precisam ser atualizados ou reescritos. Nesse momento as instituições se deparam com uma das tarefas mais críticas da tecnologia da informação, que é realizar grandes alterações e garantir a integridade de informações tão vitais. Deve-se então realizar um planejamento adequado, mesmo que não haja a troca de sistema, mas que a mudança estrutural seja suficiente para caracterizar uma nova versão, como um sistema distinto do anterior, e assim construir uma metodologia de atualização da estrutura e/ou migração de dados.

Dentro desse contexto, identificam-se metodologias por meio de uma revisão da literatura, em que estas poderão apoiar novas metodologias. Infelizmente em geral observa-se que a maioria dos sistemas está há algum tempo em uso e não foi influenciada pelo desenvolvimento ágil de *software* (*agile software development)* ou simplesmente conhecida pelos profissionais da tecnologia de informação como método ágil, formalizado a partir de 2001, e que, pela ausência de padrões, os sistemas apresentam uma característica de construção artesanal, baseando-se nas experiências empíricas dos programadores, ou nas expectativas autoritárias dos usuários, somando-se a uma documentação deficitária que dificulta ainda mais a continuidade e escalabilidade do sistema. Assim, destaca-se a importância dos requisitos de sistema legado como ponto de partida para a compreensão do sistema atual e um devido levantamento de novos requisitos. Entretanto, a imprecisão ou ausência de documentação adequada, bem como a indisponibilidade das pessoas que desenvolveram o sistema, podem impedir a recuperação dos requisitos originais do sistema e dificultar a tarefa de elicitação dos novos requisitos (ESPINDOLA et al. 2004).

Um processo fundamental durante a implantação é o reaproveitamento dos dados de outros sistemas que normalmente estão sendo substituídos pelo ERP. Para a realização desta tarefa, torna-se necessário desenvolver uma ferramenta específica para acesso ao sistema anterior, e isto envolve analisar as estruturas deste sistema e criar um novo projeto somente para esta migração, demandando tempo e todos os custos relacionados ao desenvolvimento (RIBEIRO & OLIVEIRA, 2010).

Para garantir a integridade de dados não basta apenas destruir a estrutura anterior e reconstruir a nova estrutura, mas devem-se criar algoritmos de migração de dados do banco da versão anterior para a nova versão, possibilitando um tratamento adequado dos dados legados e, em consequência, causando o mínimo de impacto ou prejuízo ao sistema atual.

A seguir apresenta-se a definição de sistemas legados, sistemas de gerenciamento de banco de dados, os tipos de dados e a equivalência que estes dados podem possuir entre os sistemas gerenciadores de banco de dados. Depois se definem as regras gerais para padronização de código para gerar um código de portável a qualquer SGBD. Em seguida apresenta-se a descrição de três metodologias de migração de dados: *Extract, Transform and Loader* (ETL), *Butterfly* e *Cold-Turkey* e a análise da comparação entre elas.

2 SISTEMAS LEGADOS

Os sistemas legados são programas computacionais que mantêm dados vitais sobre uma instituição. Quando a instituição passa a depender destes dados para ação de seu processo administrativo ou operacional, passa a existir a necessidade da manutenção de continuidade do sistema. Segundo ALLIEVI (2007), qualquer sistema em produção é considerado legado.

Os sistemas legados são uma fonte de preocupação para qualquer organização. Porém é consenso, por parte das organizações, a importância do conhecimento do modelo de dados como fator de qualidade dos sistemas de informação, ressaltando a sua contribuição para a manutenção, o desenvolvimento ou melhoria do *software* legado, e/ou a migração dos sistemas para outras plataformas tecnológicas (ALMEDA et al., 1998).

É comum se encontrar sistemas que estão em uso em instituições financeiras, desde a década de 70 e, em indústrias, desde a década de 90, que sofreram pequenas adaptações e ajustes periódicos que não são suficientes para a evolução da demanda, exigindo a troca ou o desenvolvimento de outro sistema.

2.1 Sistema de gerenciamento de banco de dados

O Sistema de Gerenciamento de Banco de Dados (SGBD) é um programa responsável pelo gerenciamento de recursos de alocação, controle e manipulação de objetos como as entidades e seus atributos, contadores, chamadas de eventos, visões definidas por consultas e tipos de dados. O SGBD mantém uma estrutura paralela aos objetos criados, que guardam todas as informações estruturais deste objeto, e estas informações estruturais são chamadas de metadados. Eles são úteis no processo de migração ou atualização de um modelo de dados por conterem os dados sobre a precisão de um campo, o tipo de dado, se ele pode ser nulo ou não, além de muitas outras (DATE, 1998).

Com o surgimento em 1974 na IBM da SQL (Structured Query Language) ou linguagem de consulta estruturada, os SGBDs começaram a utilizá-la para tarefas de criação de objetos e estrutura, manipulação de dados e regras de acesso. A linguagem passou a ser padronizada pela American National Standards Institute (ANSI) no ano de 1986 e pela International Organization for Standardization (ISO) em 1987. Após a padronização existiram cinco revisões como padrões oficiais ANSI/ISO da SQL nos anos de 1989, 1992, 1999, 2003 e 2008. Atualmente todos os SGBDs utilizam a linguagem SQL em suas atividades. Porém cada um especializa estas atividades com objetos próprios e linguagens adicionais para gerenciamento dos dados. Uma grande vantagem da Linguagem SQL é que toda a informação é passada de forma textual e cada SGBD ajusta os dados conforme padrões indicados pelos programadores, o que simplificou o acesso a dados que anteriormente deveria ser realizado pelo aplicativo final de forma binária. Como o módulo de gerenciamento reside no próprio SGBD, existe uma grande facilidade na adequação do modelo para plataformas diferentes. A similaridade da linguagem SQL também facilita a implementação do modelo nos diversos bancos de dados existentes (HUTH, 2002).

Hoje os SGBDs mais difundidos no mercado são os *softwares* proprietários Oracle, IBM DB2, Microsoft SQL-Server, e os livres são o MySQL, o PostgreSQL e o SQL-Lite, e este último em crescente utilização por ser mais adequado a dispositivos móveis.

2.2 Tipos de dados dos SGBDs

Os dados de cada campo de uma tabela podem variar em tamanho (precisão binária) ou formato. Embora existam padrões de tipos de dados no SQL ANSI/ISO, os desenvolvedores de cada SGBD criaram diversos objetos de apoio ao gerenciamento de estrutura e dados, o que inevitavelmente gera novos formatos independentes em cada um, como um campo com tipo numérico podendo

ter a classificação NUMERIC, NUMBER, FLOAT, INT, INTEGER, INT8, INT16 etc. E o programador da estrutura terá que, por meio do manual do fabricante, identificar qual é o tipo de dado mais adequado a sua necessidade conforme a tecnologia utilizada. Como resultado da análise de todos os tipos de dados dos SGBDs, a tabela 1 foi construída objetivando a comparação entre eles.

um código de SQL ANSI e converte para o seu formato na precisão devida. Foi verificado que alguns SGBDs não possuem os mesmos tipos de dados que outros, necessitando de adaptações como a do tipo DATE para VARCHAR(10), para caber a mesma informação "DD/MM/YYYY", ou TIME para VARCHAR(5) ou VARCHAR(7) como "HH:MM:SS".

Tabela 1 – Equivalência de tipos de dados

TIPOS DE DADOS	SQL ANSI	DB2	ORACLE	POSTGRESQL	MYSQL	SQLLITE
INTEIRO	SMALLINT, INTEGER	SMALLINT, INTEGER	NUMBER	SMALLINT, INTEGER, BIGINT,	TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT	INTEGER
REAL	REAL, DOUBLE, PRECISION, FLOAT	DECIMAL	BINARY_FLOAT	REAL, DOUBLE, PRECISION	FLOAT, DOUBLE, REAL	REAL, FLOAT, DOUBLE
DECIMAL	DECIMAL	DECIMAL	BINARY_DOUBLE, NUMBER	DECIMAL NUMERIC	DECIMAL	REAL, FLOAT, DOUBLE
TEXTO	CHARACTER, CHARACTER, VARYING, NATIONAL, CHARACTER, NATIONAL, CHARACTER, VARYING	VARCHAR, CHARACTER	CHAR, VARCHAR2, CLOB, NCLOB, NVARCHAR2, NCHAR	CHAR, VARCHAR, TEXT	CHAR, BINARY, VARCHAR, VARBINARY, TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT	TEXT CHAR, CLOB
BINÁRIO	BIT, BIT, VARYING	TIMESTAMP	BLOB, RAW, LONGRAW, BFILE	BYTEA	TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB	NÃO EXISTE
DATA/ TEMPO	DATE, TIME, TIMESTAMP	BLOB	DATE, TIMESTAMP	DATE, TIME, TIMESTAMP	DATETIME, DATE, TIMESTAMP, YEAR	NÃO EXISTE
LÓGICO	NÃO EXISTE	NÃO EXISTE	NÃO EXISTE	BOOLEAN	BOOLEAN, BOOL	NÃO EXISTE

Cada SGBD possui características quanto à precisão, que está relacionada ao número de bits utilizados para cada tipo de dado, porém a maioria aceita

Também os dados do tipo BOOLEAN poderiam ser exportados como CHAR(1), com a informação "F" ou "V".

2.3 Padronização de Códigos

Como o foco de uma migração é plenamente ligado à portabilidade, torna-se necessário privilegiar padrões e formatos extremamente genéricos, e o uso dos métodos e ferramentas padronizadas evitam problemas externos aos recursos utilizados na implementação da solução, não só para migração de dados, mas também para qualquer outra finalidade. Os SGBDs utilizam internamente a linguagem SQL para definição de objetos, manipulação de dados e controle do acesso a dados, e apesar de não existir uma padronização de fato, o SQL-92 é o padrão mais utilizado. As pequenas diferenças de sintaxe não comprometem a portabilidade de aplicações desenvolvidas (HUTH, 2002). Para que qualquer interpretador ou qualquer compilador leia os arquivos de comandos sem problemas, recomendá-se salvá-los em um arquivo texto com formato UTF8 (8 bit *Unicode Transformation Format*). Este formato tem caracter universal do padrão Unicode e é compatível com o ASCII. Em qualquer projeto de portabilidade é altamente recomendável não variar estes formatos, utilizando-se esses recursos universais nas tecnologias existentes, pois são comuns e evitam que ocorram erros de sintaxe por interpretação de caracteres indevidos.

3 METODOLOGIAS DE MIGRAÇÃO DE DADOS

As migrações de dados podem ter dois perfis: a transposição de arquivos ou objetos digitais para uma nova tecnologia entre servidor, e a reutilização dos dados ou atualização de uma estrutura de um sistema legado em seu SGBD. A migração da informação, de uma parte essencial de um sistema para outro, e de transferência de dados, de um ambiente para outro, estão estreitamente inter-relacionadas com os processos, podendo ser divididas em migração de dados e migração de sistemas (BROY, 2005).

A seguir são apresentadas as principais técnicas de migração levantadas na bibliografia, suas definições e comparações entre elas:

3.1 Processos de migração de dados: *Extract, Transform* e *Loader*

O ETL refere-se à sigla de Extração, Transformação e Carga (*Extract, Transform and Loader)*, é uma metodologia que se divide em três estágios: um para leitura e identificação dos dados, um para manipulação, conversão ou ajuste, e outro para preparação para inserção na tecnologia de destino.

No processo de ETL torna-se necessária a extração e carga dos dados, caracterizando-se como opcionais a transformação e o tratamento de erros dos dados de origem. Esta metodologia de não aplicar a validação ou tratamento dos dados a serem carregados deve ser adotada somente se os dados de origem estiverem em conformidade com o escopo do processo de carga (RIBEIRO & OLIVEIRA, 2010).

Pode-se identificar que esta metodologia é adequada a um ambiente homogêneo, pois servirá como um processo prático de transposição de dados, quando a estrutura de destino for semelhante ou compatível com a de origem, criando-se somente uma camada para tratamento simples dos dados e já disponibilizando uma saída própria para carregamento direto dos dados no sistema de destino.

3.1.1 Extração

Cada sistema oferece uma origem de dados, podendo ser: arquivos texto, como padrões de retorno e remessa, planilhas, arquivos xml etc. Porém o mais importante não é o formato em que o dado será exportado, visto que isso pode ser ajustado por qualquer profissional de programação, o importante mesmo é documentar muito bem como estes dados estão organizados no sistema de origem/legado.

Para isso deve-se realizar um levantamento minucioso sobre todo o modelo da aplicação e a regra utilizada, para que a implementação das operações de transformação explore todos os recursos existentes.

3.1.2 Transformação

A qualidade da documentação existente tanto do modelo de dados do sistema de origem quanto de destino fará toda diferença na construção de uma engrenagem de transformação dos dados, também conhecidas como *middleware* ou *software* intermediário. Dentro da definição da metodologia, não cabe detalhar regras, tecnologias e funções necessárias para esta transformação, e como um número muito significativo de sistemas não possui padrões de desenvolvimento, cria-se uma variação de possibilidades que necessitam ser avaliadas caso a caso, quanto a sua estrutura e regras para transformação.

Cabe à engrenagem de transformação dos dados possuir tabelas temporárias que controlem um mapa de equivalência de quais colunas ou conjunto de colunas do modelo de dados no sistema de origem são respectivas às colunas ou conjunto de colunas do modelo de dados no sistema de destino, bem como as conversões e reindexações dos relacionamentos que estas terão que sofrer.

Para exemplificar, via código SQL, a criação do módulo temporário que vai servir de engrenagem de migração necessita inicialmente de uma análise dos dois modelos de dados, e a exportação dos dados é analisada a seguir:

- Origem: a tabela ORIGEM.tbl_pessoa agrega tanto os dados de todos os telefones quanto os de endereço.
- Destino: a tabela DESTINO.tbl_pessoa não precisa de todos os dados de endereço, pois já possui uma tabela tbl_cep que se relacionará com tbl_pessoa. Já os dados telefônicos teriam que possuir quatro colunas e ser separados em quatro linhas que referenciassem cada uma delas ao identificador da pessoa.

3.1.3 Carga

Uma vez executas as rotinas de transformação, se faz necessária a saída em linguagem SQL, através de comandos de inserção, em que é indicado o uso para SQL ANSI99 em UTF8. Quando o sistema de destino for um único SGBD, deverá simplesmente executar um arquivo texto contendo os comandos. Após esta tarefa sugere-se, mesmo que nenhum erro tenha sido indicado durante as operações de migração, uma validação por amostragem dos dados.

3.2 A metodologia Chicken Little

Esta estratégia de migração consiste de onze pequenos passos, que são realizadas

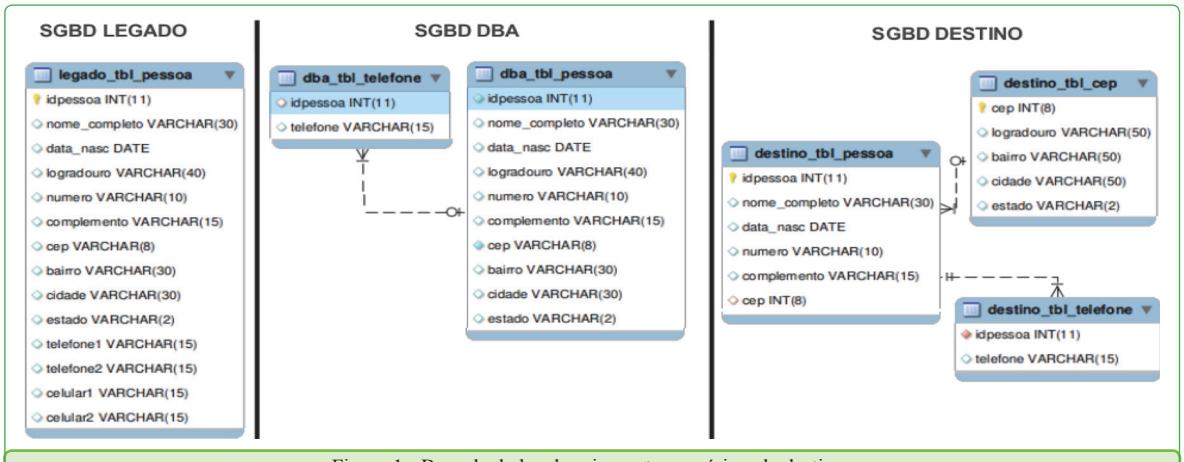

Figura 1 - Base de dados de origem, temporária e de destino

de forma iterativa, em que a migração se torna realizável, uma vez que a migração real é dividida em partes menores (princípio da *Divide and Conquer*). *Chicken Little* ainda significa uma reformulação completa do sistema. (BROY, 2005)

Essas onze partes citadas por Broy são:

1 - Analisar o sistema legado;
2 - Decompor a estrutura do sistema legado;
3 - Projetar a interface destino;
4 - Projetar a aplicação destino;
5 - Projetar a base de dados destino;
6 - Instalar o ambiente destino;
7 - Criar e instalar os *gateways* necessários;
8 - Migrar as bases legadas;
9 - Migrar as aplicações legadas;
10 - Migrar as interfaces legadas;
11 - Mudar para o sistema destino.

Os *gateways* são programas intermediários responsáveis pela transformação da informação, porém, diferentemente das engrenagens descritas no ETL, eles trabalham de forma passiva quanto às requisições e chamadas feitas pelo sistema de origem ou destino, o que permite que o administrador do projeto de migração decida por uma das estratégias a seguir:

- *Forward* ou *Database First*: em que os dados são direcionados do sistema legado para uma nova estrutura de modelo de dados. O *forward*, ou adiante, denota que o sistema de origem envia os dados para o destino.
- *Reverse* ou *Database Last*: o reverso é realmente a forma contrária ao *forward*, em que o sistema de destino busca a informação intermediada pelo *gateway* no sistema de origem.

3.3 Cold Turkey

Broy é o único autor que apresenta uma metodologia classificada por ele como *Cold Turkey* que é, como a *Chicken Little*, uma reescrita completa do sistema legado utilizando métodos de desenvolvimento moderno, em que o sistema legado é totalmente parado para evitar riscos, e começa a implantar o novo sistema utilizando uma estratégia *Big Bang*, ou seja, tudo de uma vez. O maior problema é que este método sugere parar o sistema para começar a migração e, conforme os profissionais vão encontrando erros, serão gerados os códigos adaptativos no sistema de destino, e isso cria um problema geral, uma vez que estas alterações são inseridas em partes no sistema legado que já haviam sido concluídas. O autor destaca que este método está sujeito a erros e é dispendioso.

É identificado que normalmente isso ocorre em sistemas que possuem um volume muito grande de informações, inviabilizando um planejamento adequado ou suficiente pela ausência de documentação, e que não se pode previamente criar nenhuma regra de migração e nem prever seus detalhes, tendo-se que utilizar o código fonte como documentação dos processos e requisitos, o que aumenta de forma imensurável o tempo da migração.

A *Cold Turkey* não parece ser uma metodologia e sim uma atividade aventureira de desbravamento de códigos, e só seria realizada quando nenhuma outra fosse possível, pela exploração linha a linha de seu código fonte para extrair as regras da organização de relacionamentos e tentar salvar o que fosse possível dos dados legados da instituição.

3.4 Butterfly

Esta é uma metodologia em que, diferentemente da *Chicken Little* ou *Cold Turkey*, o sistema legado permanece em operação, enquanto que, em um ambiente de teste, o novo sistema pode ser desenvolvido e testado previamente a uma migração, sem afetar o funcionamento ou mesmo causar transtornos. Sua abordagem é muito semelhante à ETL. Inclusive em literaturas que abordam a ETL não se observa a abordagem da *Butterfly*, ou vice-versa. Para uso desta abordagem, necessita-se de um bom conhecimento sobre o modelo de dados e

suas regras, tanto no sistema de origem quanto no sistema de destino dos dados, utilizando-se uma engrenagem ou motor de migração de dados que na *Butterfly* os autores classificam como *Chrysaliser*.

módulo de migração de dados entre modelos de dados homogêneos, para migração dos dados nas atualizações de versão dos módulos de um mesmo sistema. Portanto, a metodologia ETL pode ser especializada

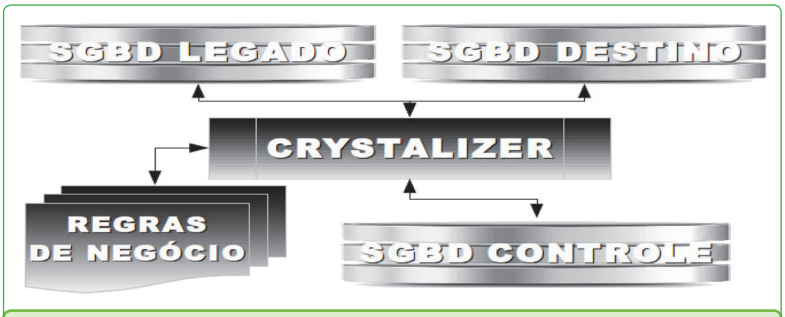

Figura 2 - *Butterfly*

O *Christalizer* é um sistema intermediário e tem a função de converter o modelo do banco transpondo a informação e realizando os ajustes necessários, para que os dados presentes no SGBD LEGADO sejam transferidos para o SGBD DESTINO com a redução de prejuízo. Como as inserções no banco de destino necessitam de regras customizadas, existe a necessidade de o administrador de banco de dados incluir estas regras manualmente no *Christalizer*, pois é necessário completar os campos de destino, que não podem ser nulos ou serão completados por consultas de relacionamentos entre duas ou mais tabelas com dados já existentes no destino.

Dentre as descritas, pode-se observar que as metodologias ETL e *Butterfly* possuem os mesmos princípios e recomendações. A ETL possui uma abordagem mais moderna; a *Chicken Little* é mais burocrática e divide o processo de migração em diversas partes, o que pode até ser interessante quando se tem uma equipe de trabalho em que existe a possibilidade de distribuir melhor as funcionalidades de controle de cada camada; e que a *Cold Turkey* é aplicada não por recomendação, mas por necessidade quando não é possível uma análise detalhada antes de começar a migração. Como resultado desta análise, especificamente as diretrizes apontadas pela metodologia ETL serviram de base para o desenvolvimento de um

em contexto de migração, integração ou atualização de SGBDs, uma vez que suas generalidades são abertas a qualquer contexto de migração de dados para qualquer objeto no universo computacional.

5 CONCLUSÃO

Este trabalho realizou um levantamento por meio de revisão da literatura, sobre as metodologias utilizadas para migração de dados, e pode-se observar que essas metodologias, embora genéricas, servem de referência para outros trabalhos sobre integração de sistemas e atualização de estrutura e migração de dados em sistemas legados, e que cada sistema apresenta particularidades em seu desenvolvimento, que necessita de uma análise caso a caso. Durante o período de levantamento da bibliografia, foram identificados detalhes específicos em documentação de sistemas proprietários, que não se baseiam em nenhuma metodologia especificada e não puderam ser citadas neste trabalho, já que suas licenças só preveem que deverão ser utilizadas junto de processos de migração de suas ferramentas. Uma vez que as especificações dos sistemas geralmente não utilizaram padrões de desenvolvimento para uma automação independente de migração de dados e a interação entre o administrador de banco de dados e a ferramenta intermediária

de controle desta migração é muito requerida, o profissional praticamente indicará todas as relações entre as tabelas de origem e destino e quais os comandos necessários para incluir dados de relacionamentos na base de destino. Para melhorar o *Christalizer* como ferramenta de apoio ao administrador de banco de dados, deixa-se como proposta para trabalhos futuros a realização de estudos para aplicação de conceitos de classificação automática de dados, *de websemântica e arquétipos, gerando uma comparação de dados para uma descrição semântica processável pelo Christalizer, em que, por meio destas técnicas, possam diminuir o número de intervenções na transposição dos dados dos modelos estruturais de bancos diferentes.*

REFERÊNCIAS

ABNT. NBR ISO/IEC9126-1: *Engenharia de software*: qualidade de produto (Parte 1: Modelo de qualidade). 2003.

ALLIEVI, O. *Reflexões sobre manutenção de sistemas legados*, 2007. Acesso em: 27 out. 2011. Disponível em: <http://imasters.com.br/artigo/5919/desenvolvimento/reflexoes_sobre_manutencao_de_sistemas_legados>.

ALMEIDA, A. B et al. Levantamento de modelos de dados em sistemas legados. *Sistema de Informação* n.9. 1998. Acesso em: 27 out. 2011. <http://193.137.8.31/index.php/revista/index>.

BROY, M. Legacy Migrationsstrategien. *Hauptseminar Management von Softwaresystemen*. TU München. Fakultät für Informatik. Lehrstuhl IV: Software & Systems Engineering. München, 2005.

DATE, C. J. *Introdução a sistema de banco de dados*. Rio de Janeiro: Campus. 8. ed. 2004.

ESPINDOLA, R. et al. Uma análise crítica dos desafios para engenharia de requisitos em manutenção de software. *Proc. VII Workshop on Requirements Engineering*, 2004, Tandil, Argentina, 2004.

HUTH, G. *Um modelo para o gerenciamento de bancos de dados SQL através de Stored Procedures*. Florianópolis, 2002. Dissertação (Mestrado em Ciências da Computação). Universidade Federal de Santa Catarina, UFSC.

RIBEIRO, A. L.; OLIVEIRA, E. C. *Processos de implantação e migração de dados com utilização de ETL para um ERP comercial*. Universidade Luterana do Brasil. Curso de Sistemas de Informação. Campus Canoas. Canoas, RS. 2010.

VALENTE, N. T. Z. *Implementação de ERP em pequenas e médias empresas*: estudo de caso em empresa do setor da construção civil. São Paulo, 2004. (Mestrado em Controladoria e Contabilidade: Contabilidade), Faculdade de Economia, Administração e Contabilidade, Universidade de São Paulo, USP.