UNIVERSIDADE FEDERAL DE SERGIPE
CAMPUS ALBERTO CARVALHO
DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO

IGOR PETERSON OLIVEIRA SANTOS

AVALIAÇÃO DA GERAÇÃO AUTOMÁTICA DE CÓDIGO PARA SUBSISTEMAS ETL EM UM AMBIENTE DE DATA WAREHOUSE

ITABAIANA
2013

UNIVERSIDADE FEDERAL DE SERGIPE
CAMPUS ALBERTO CARVALHO
DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO

IGOR PETERSON OLIVEIRA SANTOS

AVALIAÇÃO DA GERAÇÃO AUTOMÁTICA DE CÓDIGO PARA SUBSISTEMAS ETL EM UM AMBIENTE DE DATA WAREHOUSE

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

Orientador: Msc. ANDRÉ VINICIUS RODRIGUES PASSOS NASCIMENTO
Coorientador: Dr. METHANIAS COLAÇO RODRIGUES JÚNIOR

ITABAIANA
2013

SANTOS, Igor Peterson Oliveira.

Avaliação Da Geração Automática De Código Para Subsistemas Etl Em Um Ambiente De Data Warehouse/ Igor Peterson Oliveira Santos – Itabaiana: UFS, 2013. 79f.

Trabalho de Conclusão de Curso em Bacharel em Sistemas de Informação – Universidade Federal de Sergipe, Curso de Sistemas de Informação, 2013.

1. Data Warehouse.
2. 2. Banco de Dados.
3. 3. Sistemas de Informação.

I. Avaliação Da Geração Automática De Código Para Subsistemas Etl Em Um Ambiente De Data Warehouse

IGOR PETERSON OLIVEIRA SANTOS

AVALIAÇÃO DA GERAÇÃO AUTOMÁTICA DE CÓDIGO PARA SUBSISTEMAS ETL EM UM AMBIENTE DE DATA WAREHOUSE

Trabalho de Conclusão de Curso submetido ao corpo docente do Departamento de Sistemas de Informação da Universidade Federal de Sergipe (DSIITA/UFS) como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Itabaiana, 16 de abril de 2013.

BANCA EXAMINADORA:

_____

Prof. André Vinicius Rodrigues Passos Nascimento, Mestre
Orientador
DSIITA/UFS

_____

Prof. Methanias Colaço Rodrigues Júnior, Doutor
Coorientador
DSIITA/UFS

_____

Prof. Eugênio Rubens Cardoso Braz, Doutor
DSIITA/UFS

DEDICATÓRIA

Dedico esse trabalho à minha maior educadora, minha mãe Zenaide. Mãe, seus esforços e dedicação trouxeram-me a esta grande vitória. Te amo por todo seu amor, carinho e amizade.

Aos outros amores de minha vida: meu pai Petrus; minha madrinha Sônia; meu irmão Rone Peterson; e minha namorada Darla. Vocês estiveram sempre presentes e dispostos a me ajudar nessa trajetória. Sem o amor e o suporte de vocês não chegaria tão longe. Amo vocês!

AGRADECIMENTOS

“Tudo posso Naquele que me fortalece.” (Filipenses 4:13)

Antes de qualquer pessoa, sou grato Àquele quem me deu toda a força e coragem para concluir esse trabalho, meu maior amigo, Deus. Não chegaria tão longe sem a presença Dele em minha vida. Aba Pai, Te amo!

Agradeço ao meu orientador, prof. Msc. André Vinícius Rodrigues Passos Nascimento. Suas orientações conduziram-me ao melhor caminho, sempre buscando tirar o melhor de mim. Seus ensinamentos acadêmicos e de vida serão lembrados por mim com muito afeto e carinho por toda a minha vida. Obrigado pela sua amizade.

Aos professores doutores da banca: Methanias Colaço Rodrigues Júnior e Eugênio Rubens Cardoso Braz. Obrigado pelas orientações e disposição em contribuir para a conclusão desse trabalho.

A todos os professores que compõem o Departamento de Sistemas de Informação, assim como a Adilton que sempre esteve disposto a me ajudar em muitos momentos da graduação.

Aos colegas de curso que estiveram dispostos a contribuir com esse trabalho participando dos experimentos: Rafael Menezes Santos, Rodrigo Lima Aragão, Wenderson Campos Pereira e Weverton dos Santos.

Agradeço também, a uma grande amiga que fiz durante a graduação, Juli Kelle. Só você sabe o que passávamos no curso. Realizamos muitos projetos juntos: Itatech Jr, Iniciação Científica, estágio e os MUITOS trabalhos. Obrigado pela sua amizade e apoio.

E aos demais colegas e amigos de turma Damião e Lucas, que também sempre estiveram dispostos a me ajudar. Uhuuuuu.... nós conseguimos! \o/

Muito obrigado a todos.

SANTOS, Igor Peterson Oliveira. **Avaliação Da Geração Automática De Código Para Subsistemas Etl Em Um Ambiente De Data Warehouse.** 2013. Trabalho de Conclusão de Curso – Curso de Sistemas de Informação, Departamento de Sistemas de Informação, Universidade Federal de Sergipe, Itabaiana, 2013.

RESUMO

O termo *Data Warehouse* é adotado pelas organizações como sinônimo de repositório de dados de alta qualidade. Desse modo, identificar problemas com validade, consistência e integridade dos dados representa uma preocupação constante das empresas no processo de utilização dos sistemas de suporte à decisão. A codificação manual de rotinas para povoamento de dimensões é apontada como uma das principais causas da má qualidade dos dados em um *Data Warehouse*. Erros de codificação, estratégias incorretas de atualização, e má interpretação das estratégias de armazenamento de histórico representam os principais problemas que podem ser gerados pela codificação manual. Acreditamos que o uso da geração automática de código para rotinas de povoamento possa substituir a codificação manual, reduzir o número de erros de codificação e eliminar interpretações incorretas na codificação de estratégias de atualização. Nesse trabalho, apresentamos os resultados da utilização de uma ferramenta para geração automática de código para rotinas de povoamento de hierarquias, agregados e dimensões com tratamento de histórico híbrido. Os experimentos apresentam fortes evidências de que é possível capturar a semântica necessária para gerar automaticamente rotinas de povoamento para os tipos de dimensões, agregados e hierarquias, além de contribuir para o aumento da produtividade e redução de erros de codificação na construção dessas rotinas.

Palavras-chave:
Data Warehouse, Qualidade de Dados, Geração Automática de Código.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1. INTRODUÇÃO

A informação representa fator crucial para as empresas no processo de tomada de decisões. Visto que as mudanças no mundo dos negócios são constantes, as organizações veem que a informação não representa um mero resultado de transações, mas um propulsor para atingir melhores resultados (COLAÇO, 2004). O ambiente responsável por fornecer essas informações, de forma eficiente, prática e com qualidade, é chamado de *Data Warehouse* (DW). Embora esse conceito tenha surgido nos anos 90, as empresas reconhecem hoje a grande importância de um DW em suas atividades diárias.

Data Warehouse representa um banco de dados histórico, separado lógica e fisicamente do ambiente de produção de uma organização, concebido para dar suporte às análises e decisões gerenciais. A ideia por trás dessa abordagem é selecionar, integrar e organizar dados provenientes do ambiente operacional e fontes externas para que possam ser acessados de forma mais eficiente e para que possam representar uma única realidade da organização (KIMBALL, 2008) (INMON, 2005) (COLAÇO, 2004).

Segundo (COLAÇO, 2004), ao implantar um DW, os administradores esperam alcançar benefícios, como:

a) Recursos para acessar de modo rápido e flexível as informações do negócio;

b) Disponibilidade de mecanismos que incorporam a inteligência do negócio e permitem efetuar o acompanhamento do desempenho e identificar as exceções no padrão de comportamento esperado;

c) Criação do conhecimento com base na análise de diversos cenários e identificação de padrões de comportamento ou preferências/hábitos comuns;

d) Redução de riscos associados aos negócios, através das facilidades de análise de risco e avaliação de alternativas;

e) Desenvolvimento de um "*marketing* de relacionamento" efetivo, permitindo a definição de estratégias com foco nos clientes e atendimento das suas expectativas, visando à elevação da taxa de retenção dos mesmos.

O ambiente de DW é projetado para ser uma coleção de dados não-volátil, variante no tempo e que possa corretamente armazenar dados de diferentes sistemas da organização (SANTOS; BELO, 2011a).

A importância de um ambiente de *Data Warehouse* está diretamente relacionada com a qualidade [1] dos dados que são armazenados. Desse modo, identificar e solucionar problemas com validade, consistência e integridade dos dados representam preocupações constantes das empresas no processo de utilização dos sistemas de suporte à decisão.

Os problemas com qualidade de dados podem surgir em várias fases do processo de carga, conhecido como ETL (*Extract, Transform and Load*), especialmente no estágio de povoamento (RANJIT; KAWALJEET SINGH, 2010). Durante o processo de ETL, os dados são extraídos do ambiente operacional, transformados e carregados para estruturas específicas, conhecidas como dimensões e fatos. Essas estruturas podem sofrer diversas atualizações de acordo com as regras do negócio da organização. O carregamento correto dos dados para as dimensões e fatos finais representa um fator crítico para o sucesso de um DW.

As rotinas de povoamento de dimensões e fatos, desenvolvidas na última fase do processo de carga desse ambiente são, geralmente, codificadas manualmente com o objetivo de capturar os tratamentos específicos que devem ser dados a esses artefatos. Essa codificação manual, assim como os erros nas estratégias de implementação de dimensões que armazenam histórico são apontadas, dentre outras, como principais causas para a má qualidade de dados em um *Data Warehouse* (RANJIT; KAWALJEET SINGH, 2010).

Este trabalho representa uma extensão de um projeto de pesquisa, iniciado em Setembro de 2011, que tem como objetivo verificar se a utilização de uma ferramenta de geração automática de código pode substituir a codificação manual, aumentar a produtividade e reduzir o número de erros de codificação durante a fase de povoamento de dados em um ambiente de *Data Warehouse*. Neste projeto, o objetivo é avaliar a geração automática de código para determinados subsistemas (KIMBALL, 2008) (BECKER, 2007): Dimensões do tipo 4 e 6, Hierarquias em Dimensões, e Agregados.

[1] Nesse contexto, o termo qualidade está relacionado com as seguintes propriedades: disponibilidade, consistência, validade, conformidade, precisão e integridade (RUDRA, 1999).

1.1. Objetivos

1.1.1. Objetivo geral

O objetivo desse trabalho é avaliar o uso da geração automática de código para rotinas de povoamento em um ambiente de *Data Warehouse.*

1.1.2. Objetivos específicos

- Definição de metadados para capturar a semântica para o povoamento de tipos de Dimensões não convencionais;
- Definição de metadados para capturar a semântica para o povoamento de hierarquias;
- Definição de metadados para capturar a semântica para o povoamento de agregados;
- Desenvolvimento da ferramenta de geração automática de código;
- Realização de experimentos a fim de avaliar a geração automática de código para as rotinas de povoamento;
- Avaliação dos resultados dos experimentos.

1.2. Relevância do trabalho

O presente trabalho apresenta importantes contribuições. A principal contribuição é a investigação da utilização de uma ferramenta de geração automática de código para rotinas de povoamento em um ambiente de *Data Warehouse*. Os resultados dessa investigação, assim como a ferramenta construída e utilizada, poderão ser utilizados para a escolha de uma abordagem de projeto para Sistemas de Suporte à Decisão. Ao estender um projeto de pesquisa, o trabalho também contribui para uma melhor formação do aluno em relação a trabalhos de pesquisa, condução de experimentos, análise e discussão de resultados. A geração automática de código impõe a necessidade de conhecimentos sólidos sobre a disciplina Modelagem Dimensional. Com o decorrer do trabalho, o aluno passa a conhecer

profundamente a semântica envolvida nessa técnica de modelagem. A atividade de definição de metadados exercita a competência de abstração, necessária para a construção de esquemas conceituais que constituem a base da ferramenta de geração de código. Uma outra contribuição é que a ferramenta, fruto do trabalho, poderá ser utilizada por outros alunos que estejam cursando as disciplina de Sistemas de Informação, Sistemas de Suporte à Decisão e Mineração de Dados para a construção de rotinas de povoamento para um esquema dimensional. Essas rotinas, fruto da aplicação das melhores práticas e padrões, podem contribuir para uma melhor formação dos alunos nessas disciplinas.

1.3. Metodologia

Inicialmente, será realizada a revisão da literatura e pesquisa bibliográfica sobre os novos subsistemas ETL que irão compor este trabalho: dimensões com comportamento tipo 4 e 6, hierarquias e agregados.

Após a revisão da literatura, serão definidos os metadados necessários para capturar a semântica necessária para o povoamento de cada subsistema. Definidos os metadados, será realizado o desenvolvimento da ferramenta, de forma iterativa e incremental. Cada incremento será responsável por um subsistema.

Após o desenvolvimento da ferramenta, serão executados os experimentos a fim de acatar ou refutar as hipóteses do trabalho. Abaixo seguem as etapas que serão realizadas para a execução dos experimentos.

1. *Criação do ambiente de Data Warehouse:* nessa fase será definido e criado o ambiente de DW que contenha o esquema dimensional, a Área de *Staging* e a área de Violação. Estes servirão como base central para o início de todo o experimento.
2. *Definição dos Casos de Uso para as rotinas de povoamento:* serão definidas especificações para povoamento de dados a serem seguidas pelos programadores que farão parte do experimento.
3. *Revisão de conceitos básicos sobre rotinas de povoamento para o grupo de programadores:* antes de iniciar qualquer etapa junto com os colaboradores, haverá uma revisão sobre as rotinas de povoamento para ambientes de DW.

4. *Treinamento da ferramenta de geração automática de código:* um treinamento será dado para os programadores, a fim de que eles possam se familiarizar com a ferramenta.

5. *Solicitação de execução dos Casos de Uso de forma manual:* os programadores deverão programar as rotinas de povoamento manualmente. Isto servirá para fazer a comparação com as rotinas geradas pela ferramenta deste projeto.

6. *Solicitação de execução dos Casos de Uso de forma automática:* nesta etapa os colaboradores irão utilizar a ferramenta e verificar se o código consegue atingir o propósito de povoar o esquema dimensional corretamente.

Ao término do experimento, serão analisadas as métricas que servirão de base para avaliar as hipóteses. Neste momento serão analisadas duas métricas: *a) produtividade de desenvolvimento* – verificar o tempo gasto para a criação do procedimento; *b) número de erros de codificação* – número de erros de codificação encontrados nas fases de criação e manutenção dos procedimentos.

1.4. Estrutura do trabalho

O restante do trabalho está estruturado como segue. No capítulo 2 são apresentados os principais conceitos sobre o tema do trabalho: Ambiente de *Data Warehouse*, Modelagem Dimensional, Fatos, Dimensões, Tipos de Dimensões, Agregados e Hierarquias. No capítulo 3, serão descritas as principais atividades realizadas durante o processo de desenvolvimento da Ferramenta de Geração Automática de Código. O capítulo 4 descreve como foram realizados os experimentos do trabalho: objetivo, definição do ambiente, planejamento, execução e coleta de dados. No capítulo 5 serão apresentadas as contribuições e os resultados alcançados do trabalho. Finalmente, no capítulo 6 são apresentadas as conclusões do trabalho.

2. REVISÃO BIBLIOGRÁFICA

O uso inicial do termo *Data Warehouse* é creditado a (INMON, 2005), que o define como sendo uma coleção de dados orientada por assunto, integrada, não-volátil, variante no tempo, que dá apoio às decisões da administração. (KIMBALL, 2008) dá uma definição mais sucinta: "uma cópia das transações de dados especificamente estruturada para consultas e análises". Ambas as definições destacam o histórico de dados que é encontrado em um DW, destinado a análises e suporte à tomada de decisões gerenciais.

A figura 1 ilustra a arquitetura genérica de um ambiente de *Data Warehouse*. Nela podemos observar três principais componentes da arquitetura: o ambiente OLTP, a Área de *Staging* e o *Data Warehouse*.

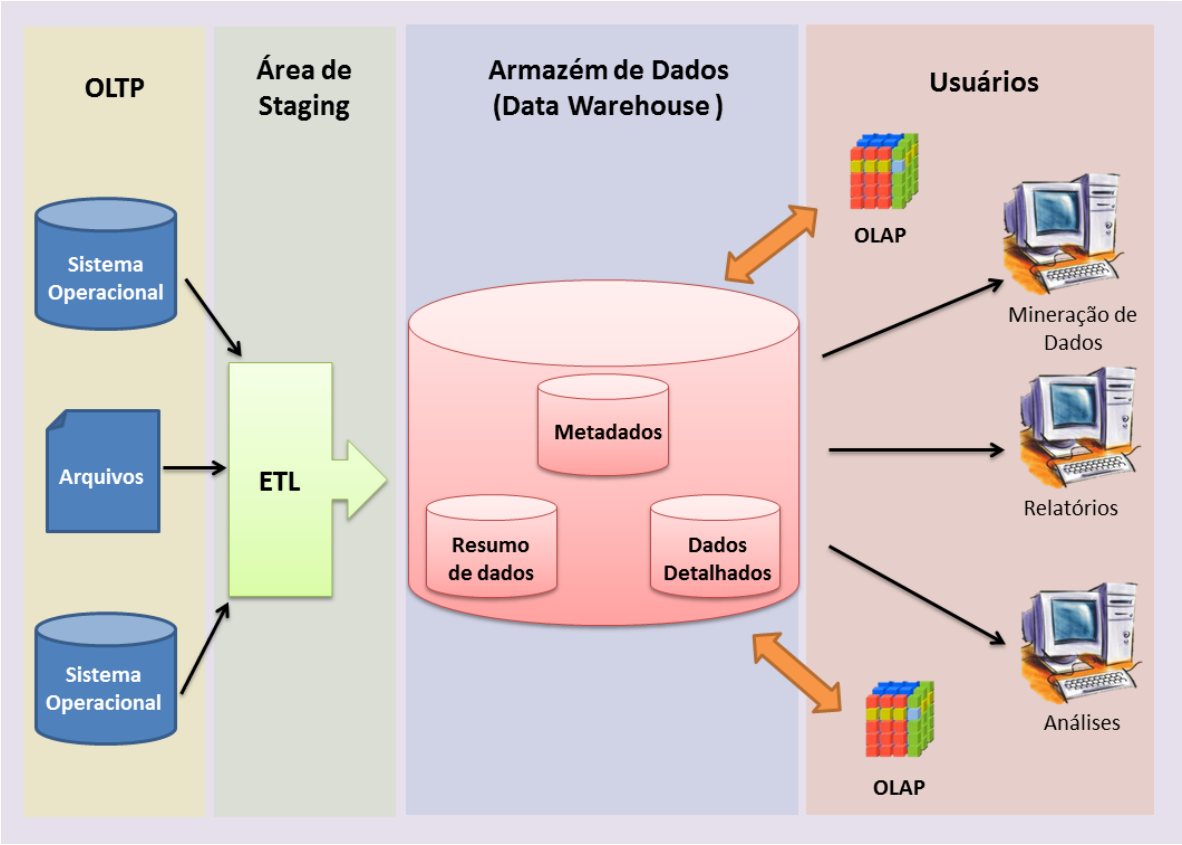


Figura 1: *Arquitetura de um DW.*

O ambiente **OLTP** (*Online Transaction Processing*, em português, Processamento de Transações em Tempo Real), representa os sistemas envolvidos nos processos diários das

organizações. Esses sistemas representam as fontes de informações para o *Data Warehouse*. Antes de serem carregados para as estruturas finais e disponibilizados para os tomadores de decisões, é necessário que esses dados passem primeiramente por uma área intermediária, conhecida como **Área de *Staging* (**KIMBALL,2008) (COLAÇO, 2004). Essa área intermediária representa a base do DW. É na Área de *Staging* que ocorrem as principais transformações, integrações e tratamento dos dados provenientes do ambiente transacional. O termo **ETL** (Acrônimo de *Extract-Transform-Load*, que pode ser traduzido como Extração-Transformação-Carga) representa os processos responsáveis pela extração de dados do ambiente transacional, pela transformação dos dados, e pela carga ou povoamento para o esquema de dados do *Data Warehouse*.

Os **Metadados** representam uma espécie de dicionário de dados, que agrega dados sobre outros dados. Os metadados permitem aos usuários finais ou analistas de Sistemas de Suporte à Decisão navegar através das possibilidades. Dito de outro modo, quando um usuário trabalha com um *Data Warehouse* que não possui metadados, perde-se um tempo considerável tentando descobrir as informações que existem no *Data Warehouse* assim como as análises que podem ser realizadas (COLAÇO, 2004) (INMON, 2005).

Os Dados Detalhados nada mais são que os dados provenientes do ambiente operacional, relevantes para a tomada de decisões, que passaram por um processo de carga, limpeza e integração. Os dados de resumo são dados muito menos volumosos e mais fáceis de gerenciar do que os dados detalhados. A partir de uma perspectiva de acesso e apresentação, os dados de resumo são ideais para a gestão. Afinal, eles são resumos de grandes volumes de dados detalhados. Isso contribui para evitar novas análises no futuro sobre os mesmos dados e tornar as análises mais eficientes (INMON, 2005).

Após os dados serem carregados efetivamente para o DW, serão feitas manipulações e analises de um grande volume de dados sob múltiplas perspectivas através de ferramentas **OLAP** (*On-line Analytical Processing*). Essa classe de ferramenta é utilizada para realizar analises estratégicas, mineração de dados e elaboração de relatórios. As aplicações OLAP são usadas pelos gestores em qualquer nível da organização para lhes permitir análises comparativas que facilitem a tomada de decisões.

2.1. Modelagem dimensional

No ambiente de suporte à decisão, onde os dados são cruzados a fim de obter informações importantes para o negócio, os volumes de dados envolvidos nas consultas são maiores em comparação com o ambiente operacional. Por esse motivo, a criação de esquemas de dados que seguem os padrões e objetivos do ambiente operacional, eliminar redundâncias e favorecer pequenas transações, não são adequados para a modelagem de um *Data Warehouse*. Para Kimball, a Modelagem Dimensional (MD), técnica de projeto lógico de banco de dados, é mais recomendada para este tipo de aplicação, pois os esquemas criados através dessa técnica contêm a mesma informação que os esquemas derivados de um DER, mas agrupam os dados em um formato que melhora a clareza do usuário e desempenho das consultas através da diminuição do grau de normalização. Os esquemas de dados criados com a Modelagem Dimensional recebem o nome de Esquema Estrela ou Esquema em Estrela (KIMBALL, 2008).

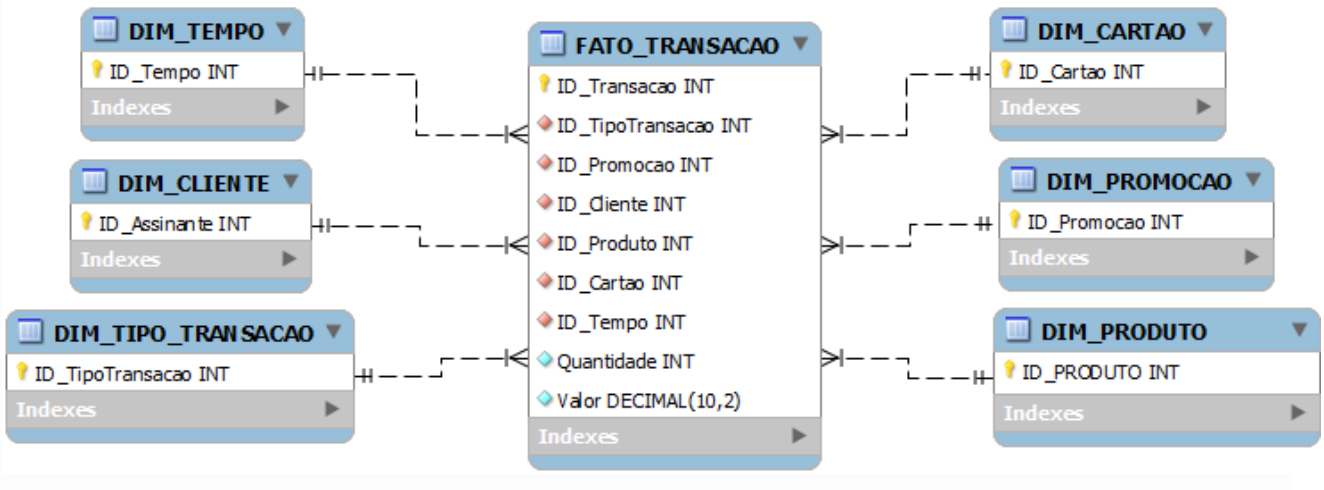


Figura 2: *Exemplo de um Esquema Estrela produzido através da Modelagem Dimensional.*

A Figura 2 apresenta um exemplo de um esquema estrela. Basicamente, um esquema estrela consiste de uma tabela principal, denominada tabela de fatos, e de tabelas auxiliares conhecidas como dimensões. As tabelas de fatos são responsáveis por registrar as transações, ocorrências ou fatos de um processo de negócio. As dimensões representam os elementos que irão contextualizar os fatos registrados. A fase de povoamento do processo ETL busca, então, povoar as tabelas de fatos e dimensões. As rotinas responsáveis por essa fase são as principais responsáveis pelo tratamento e armazenamento correto do histórico de

dados dentro de um ambiente de suporte à decisão. Nas próximas seções, buscamos apresentar as principais características dos fatos e dimensões.

2.1.1. Tabela de fatos

As tabelas de fatos são as principais tabelas em um esquema estrela (KIMBALL, 2008). Nela são guardadas as medidas numéricas mais importantes do processo de negócio sendo modelado. As tabelas de fato armazenam grande quantidade de dados históricos em função do tempo, obtidos a partir da intersecção de todas as dimensões da estrela (COLAÇO, 2004). As dimensões ligam-se aos fatos através de suas chaves primárias (*Surrogate Keys*). Essas chaves são atributos fundamentais para qualidade e composição de uma tabela de fato.

Toda tabela de fatos possui uma propriedade conhecida como grão ou granularidade. A granularidade de uma tabela de fatos representa o nível de detalhe associado com as medidas armazenadas. O nível de granularidade é mais baixo quando são apresentados mais detalhes, e mais alto quando são apresentados menos detalhes. A granularidade também pode ser entendida como o significado de uma linha em uma tabela de fatos. Pode-se dizer, por exemplo, que a granularidade de uma tabela de fatos que representa vendas é: uma venda de um produto, realizada por um vendedor para um determinado cliente em um determinado dia. Nesse exemplo, possíveis medidas como quantidade e valor terão seus significados associados à granularidade.

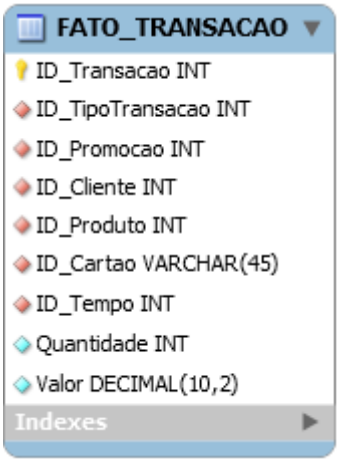


Figura 3: *Exemplo de Tabela Fato (Tabela Fato de Transações).*

A figura 3 é um exemplo de tabela fato cuja granularidade é a venda de um produto a um cliente em determinado dia. Uma tabela de fatos é composta por dois tipos de atributos: a) Os atributos que representam as ligações com as dimensões; b) Os atributos que representam as medidas do negócio. Na Figura 03, por exemplo, temos os seguintes atributos que representam as ligações com as dimensões: Id_TipoTransacao, Id_Promocao, Id_Tempo, Id_Cliente, Id_Produto, Id_Cartao. Os atributos Quantidade e Valor representam as medidas para o fato registrado.

2.1.2. Dimensões

As dimensões representam tabelas cujos atributos são utilizados para contextualizar, restringir e agrupar consultas direcionadas para tabelas de fatos dentro de um esquema dimensional. Em outras palavras, representam a principal fonte de informações e restrições para o processo de análise. Em função disso, os atributos das dimensões devem apresentar valores descritivos e, geralmente, apresentam algum relacionamento hierárquico (IMHOFF; GALEMMO; GEIGER, 2003).

Na sua forma mais simples, as tabelas de dimensões são compostas basicamente por colunas que contêm elementos textuais que descrevem o negócio, e uma chave primária, na forma de uma *Surrogate Key* (KIMBALL, 2008). A depender da abordagem para tratamento de histórico, outras colunas podem surgir como padrão para compor as tabelas de dimensões. A figura 04 apresenta um exemplo de uma dimensão. Nela, podemos identificar o atributo que representa a chave primária, Id_produto, e os atributos que descrevem as características do produto modelado.

A Figura 4 representa um exemplo de tabela dimensão de produto.

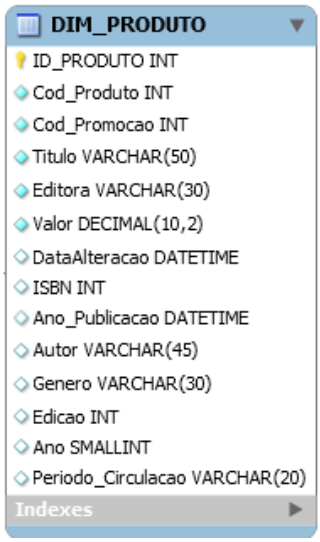


Figura 4: *Exemplo de Tabela Dimensão (Tabela Dimensão Produto).*

2.1.3. Tratamento de histórico em dimensões

Embora os valores dos atributos das dimensões sejam relativamente estáticos, suas mudanças podem ser relevantes para a análise de dados e tomada de decisões. A maneira como essas mudanças são tratadas pelo esquema de dados do *Data Warehouse* determina o tratamento de histórico que será dado para cada atributo da dimensão. Podemos classificar os atributos de uma dimensão, em função do tratamento de histórico, nos seguintes tipos: Tipo 0, Tipo 1, Tipo 2 e Tipo 3 (SANTOS; BELO, 2011b). É comum encontrarmos na literatura a classificação anterior dada às dimensões e não aos atributos (KIMBALL, 2004) (KIMBALL, 2008). Essa prática, embora utilizada para fins didáticos, pode levar a erros de interpretação, pois uma dimensão pode apresentar atributos com comportamentos variados em relação ao tratamento de histórico. A combinação dos tipos de comportamento citados anteriormente dão origem às abordagens conhecidas como Tipo 4 e Tipo 6. Nas seguintes seções, serão detalhados todos esses comportamentos.

2.1.3.1. Tipo 0

Quando um atributo tem seu comportamento, em relação ao histórico, classificado como Tipo 0 significa dizer que será descartada qualquer mudança que possa ocorrer com o

atributo (SANTOS; BELO, 2011b). Em outras palavras, o atributo armazena sempre o valor original.

Na figura 5 (a) é apresentado um resumo da tabela Tb_Funcionario no ambiente operacional. Na figura 5 (b) é apresentado um resumo da dimensão Funcionário. Nesse exemplo, o atributo Salario_Inicial da dimensão Funcionário é povoado com valores do atributo Salario da tabela Tb_Funcionario, e armazena o primeiro salário do funcionário na empresa. É um requisito do negócio que o valor do atributo Salario_Inicial permaneça sem alterações, mesmo que ocorram modificações no ambiente operacional. Logo, ele é classificado como Tipo 0. As alterações subsequentes nos valores do atributo Salario na tabela Tb_Funcionario não devem afetar os valores do atributo Salario_Inicial na dimensão Funcionario. A figura 6, por exemplo, apresenta o resultado de modificações em registros da tabela Tb_Funcionario e os reflexos dessas modificações na dimensão Funcionário.

Id_funcionario	Cod_funcionario	Nome_funcionario	Salario
1	10	Sérgio	720.00
2	11	Marcos	1050.00
3	12	Joana	680.00
4	13	Simone	894.00

(a)

Id_funcionario	Cod_funcionario	Nome_funcionario	Salario	Salario_Inicial
1	10	Sérgio	720.00	720.00
2	11	Marcos	1050.00	1050.00
3	12	Joana	680.00	680.00
4	13	Simone	894.00	894.00

(b)

Figura 5: *(a) Tabela Tb_Funcionario no ambiente operacional. (b) Dimensão Funcionário.*

Id_funcionario	Cod_funcionario	Nome_funcionario	Salario
1	10	Sérgio	790.00
2	11	Marcos	1050.00
3	12	Joana	850.00
4	13	Simone	894.00

(a)

Id_funcionario	Cod_funcionario	Nome_funcionario	Salario	Salario_Inicial
1	10	Sérgio	790.00	720.00
2	11	Marcos	1050.00	1050.00
3	12	Joana	850.00	680.00
4	13	Simone	894.00	894.00

(b)

Figura 6: *(a) Tabela Tb_Funcionario no ambiente operacional após atualizações. (b) Dimensão Funcionário após atualizações.*

2.1.3.2. Tipo 1

No Tipo 1 não há armazenamento de histórico. O comportamento do atributo, para este tipo, sobrescreve o valor antigo de uma linha da dimensão pelo valor atual proveniente da Área de *Staging*. Dessa forma, o atributo sempre irá refletir o valor mais recente proveniente do ambiente operacional. Para atualização do atributo é utilizada a chave natural, uma vez que esta deve permanecer inviolável e constante. Afinal, ela servirá como referência entre os ambientes transacional e de suporte à decisão para a atualização dos dados. Ao serem identificados novos valores, o procedimento cria um novo registro no DW.

Na figura 7 é demonstrado um resumo de valores de uma dimensão de Cartão de Crédito. Nesse exemplo o atributo Validade_cartao possui comportamento do Tipo 1. Na figura 7 em (a) podemos encontrar os registros para o primeiro carregamento da dimensão. Se o valor da validade de cartão, por exemplo, for alterado no ambiente operacional, essa alteração será refletida na dimensão e o valor antigo da validade será perdido. Essa alteração pode ser vista na figura 7 em (b) nos registros 1 e 4, onde os valores 11/11 e 03/10 passaram a ser 10/16 e 01/15, respectivamente.

Id_cartao	Cod_cartao	Nome_titular	Validade_cartao
1	10	João	11/11
2	11	Carlos	02/14
3	12	Maria	05/16
4	13	Fabiana	03/10

(a)

Id_cartao	Cod_cartao	Nome_titular	Validade_cartao
1	10	João	10/16
2	11	Carlos	02/14
3	12	Maria	05/16
4	13	Fabiana	01/15

(b)

Figura 7: *Tabela Dim_Cartao com valores*.

Embora a abordagem para atributos do Tipo 1 seja considerada a mais fácil e rápida a ser implantada, há o problema com a perda de histórico de mudanças do atributo. A abordagem de Tipo 1 é apropriada quando as mudanças que ocorrem nos valores dos atributos são correções, ou quando não há interesse em armazenar valores antigos. Para determinar a possibilidade de atribuir o comportamento do Tipo 1 a um atributo, é importante assegurar que não existem análises e relatórios que levem em consideração os valores antigos desse atributo. (KIMBALL, 2002).

2.1.3.3. Tipo 2

O comportamento Tipo 2, ao contrário do tipo anterior, é utilizado quando há a necessidade de armazenar histórico na dimensão. O tratamento de histórico é realizado através da criação de novos registros. É gerado um novo valor do identificador da dimensão (*Surrogate Key*) quando há alteração em atributos que necessitam armazenar todo o seu histórico. Como o comportamento Tipo 2 gera novas linhas, uma desvantagem dessa abordagem é a possibilidade do crescimento acelerado da dimensão.

No esquema da dimensão que possui atributos do Tipo 2 surgem novos atributos que não existem na dimensão que contempla somente atributos do Tipo 1. Esses novos atributos são utilizados para o tratamento dado ao histórico nesse tipo de dimensão. São eles: a data inicial, que representa a data em que o registro foi gravado; a data final, que representa a data em que o registro deixou de ser corrente; e por fim, o atributo que identifica se é um

registro atual ou não. Estes são apresentados respectivamente na figura 08com os nomes de: DT_INICIO; DT_FIM e FL_CORRENTE.

Na figura 8 é apresentado um resumo com valores de uma Dimensão de Clientes. Nesse exemplo todos são atributos que armazenam histórico, com exceção da *Surrogate Key* (Id_cliente), Cod_cliente, DT_inicio, DT_fim e FL_corrente.

Quando um registro é inserido, um novo identificador (*Surrogate Key*) é criado e os dados são carregados para a dimensão. Com isso, o atributo DT_inicio armazena a data de quando o registro foi inserido na dimensão. No atributo DT_fim é registrado a ausência de valor (NULL), afinal tal registro não deixou de ser atual. E por fim, há o FL_Corrente que apresenta o valor 'S' para identificar o registro atual. O que foi descrito aqui pode ser visto na figura 8 em (a).

Id_cliente	Cod_cliente	Nome	Endereco	Cidade	DT_inicio	DT_fim	FL_corrente
1	11	João	Rua Alagoas	São Paulo	2011-12-29	NULL	S
2	12	Carlos	Rua Sergipe	Rio de Janeiro	2011-12-29	NULL	S
3	13	Maria	Rua Bahia	São Paulo	2011-12-29	NULL	S
4	14	Fabiana	Rua Piauí	São Paulo	2011-12-29	NULL	S

(a)

Id_cliente	Cod_cliente	Nome	Endereco	Cidade	DT_inicio	DT_fim	FL_corrente
1	11	João	Rua Alagoas	São Paulo	2011-12-29	2012-01-10	N
2	12	Carlos	Rua Sergipe	Rio de Janeiro	2011-12-29	NULL	S
3	13	Maria	Rua Bahia	São Paulo	2011-12-29	NULL	S
4	14	Fabiana	Rua Piauí	São Paulo	2011-12-29	2012-01-10	N
5	11	João Carlos	Rua Alagoas	São Paulo	2012-01-10	NULL	S
6	14	Fabiana	Rua Pará	Rio de Janeiro	2012-01-10	NULL	S

(b)

Id_cliente	Cod_cliente	Nome	Endereco	Cidade	DT_inicio	DT_fim	FL_corrente
1	11	João	Rua Alagoas	São Paulo	2011-12-29	2012-01-10	N
2	12	Carlos	Rua Sergipe	Rio de Janeiro	2011-12-29	NULL	S
3	13	Maria	Rua Bahia	São Paulo	2011-12-29	2012-01-25	N
4	14	Fabiana	Rua Piauí	São Paulo	2011-12-29	2012-01-10	N
5	11	João Carlos	Rua Alagoas	São Paulo	2012-01-10	NULL	S
6	14	Fabiana	Rua Pará	Rio de Janeiro	2012-01-10	NULL	S
7	13	Maria Clara	Rua Acre	São Paulo	2012-01-25	NULL	S

(c)

Figura 8: *Tabela Dim_Clientes com valores.*

Quando alguns dos atributos que armazenam histórico sofrem alterações, o atributo DT_Fim irá registrar a data da alteração. Consequentemente o valor do atributo FL_Corrente será alterado para 'N', o que indica que aquele registro deixou de ser atual. Uma nova linha, com um novo identificador (*Surrogate Key*), é inserida com as alterações feitas. Como exemplo disso, temos os registros com Id_cliente iguais a 1 e 4 que foram alterados no dia

2012-01-10, figura 8 em (b). Para o primeiro, houve uma mudança no nome do cliente, de João para João Carlos (visto no registro com Id_cliente igual a 5). Para o segundo, foram alterados os campos de endereço e cidade, e foi adicionado o registro com Id_cliente igual a 6 com a alteração.

Por fim, uma nova atualização na data 2012-01-25 identificou alteração no registro com Id_cliente igual a 3, figura 8 em (c). Houve as mudanças no nome e endereço da cliente - de Maria para Maria Clara e Rua Pará para Rua Acre, respectivamente - e foi inserido o registro de Id_cliente igual a 7.

2.1.3.4. Tipo 3

É utilizado quando há uma mudança e existe a necessidade de manter o histórico sem a criação de um novo registro. Essa característica é alcançada através do armazenamento do histórico em vários atributos de um mesmo registro de dados. Logo, a partir da necessidade, podem ser criadas quantas colunas forem desejadas para a dimensão. Uma desvantagem para esse tipo de técnica é o aumento do número de colunas na tabela da dimensão.

Na figura 9, é apresentado um resumo com valores de uma Dimensão de Produtos. Nesse exemplo, temos o atributo nome que armazena histórico. O tratamento de histórico aqui é realizado através da utilização de novos atributos: Nome_antepenultimo, Nome_penultimo e Nome_atual que armazenam os valores referentes às três últimas alterações feitas no atributo. Nessa abordagem, o histórico é armazenado no mesmo registro da dimensão e não há criação de novos registros.

Sempre que houver um novo registro, o atributo Nome_atual receberá esse valor, refletindo, dessa forma, o valor mais atual referente ao de produção. O atributo Nome_original conterá sempre o primeiro valor que foi atribuído ao nome do produto, figura 9 em (a). Os outros atributos, Nome_antepenultimo e Nome_penultimo, permanecem com ausência de valor.

Id_produto	Cod_produto	Nome_original	Nome_antepenultimo	Nome_penultimo	Nome_atual
1	11	Brisas	NULL	NULL	Brisas
2	12	Due	NULL	NULL	Due
3	13	Joop!	NULL	NULL	Joop!
4	14	Versace	NULL	NULL	Versace

(a)

Id_produto	Cod_produto	Nome_original	Nome_antepenultimo	Nome_penultimo	Nome_atual
1	11	Brisas	NULL	Brisas	Suaves Brisas
2	12	Due	NULL	Due	Due Feminino
3	13	Joop!	NULL	NULL	Joop!
4	14	Versace	NULL	NULL	Versace

(b)

Id_produto	Cod_produto	Nome_original	Nome_antepenultimo	Nome_penultimo	Nome_atual
1	11	Brisas	Brisas	Suaves Brisas	Novas Brisas
2	12	Due	Due	Due Feminino	Due Masculino
3	13	Joop!	NULL	NULL	Joop!
4	14	Versace	NULL	Versace	Versace Bright

(c)

Figura 9: *Tabela Dim_Produto com valores.*

Ao receber uma nova atualização dos dados, pôde-se perceber, em (b), que os nomes dos perfumes mudaram nos registros 1 e 2, em Id_produto. Dessa forma, o atributo Nome_atual recebeu os novos valores e os valores mais velhos foram armazenados no atributo Nome_antepenultimo. Em (c), houve nova atualização e surgiu novas alterações nos valores dos registros 1, 2 e 4. Para os dois primeiros, o atributo Nome_antepenultimo recebeu o valor que era do Nome_penultimo, que por sua vez, recebeu o valor do Nome_atual. Este atributo passa a ter o valor mais atual do registro. Para o registro 4, em Id_produto, o penúltimo nome recebeu o valor mais antigo, Versace, e o nome atual passou a ser Versace Bright.

2.1.4. Tratamento de histórico híbrido

Os comportamentos para o tratamento de histórico descritos anteriormente podem ser combinados para que tenhamos diferentes abordagens em uma mesma dimensão. As dimensões que apresentam atributos com comportamento diferente em relação ao tratamento de histórico são consideradas dimensões híbridas. Embora essa flexibilidade possa ser vista, por muitos profissionais, como o melhor dos mundos, a abordagem híbrida apresenta certa complexidade para o usuário final. Logo, essas abordagens só devem ser utilizadas quando forem realmente necessárias para atender os requisitos do negócio (Kimball, 2002).

2.1.4.1. Tipo 4

A ideia do Tipo 4 é considerar uma dimensão como sendo formada por duas tabelas: uma tabela com a situação corrente (considerando que seus atributos apresentam o comportamento Tipo 1) e uma tabela armazenando o histórico (considerando que alguns de seus atributos possuem o comportamento Tipo 2). Logo, a dimensão Tipo 4 representa uma abordagem híbrida que engloba as técnicas Tipo 1 e Tipo 2. Uma das vantagens desse tipo de dimensão é que a tabela de dimensão que corresponde à situação corrente se mantem pequena, permitindo que análises futuras e pesquisas possam ser realizadas com alto desempenho. Além disso, todo o histórico de mudanças também fica disponível caso haja necessidade.

A literatura apresenta diferentes abordagens para a implementação de uma dimensão Tipo 4. Em (SANTOS; BELO, 2011a), a dimensão que armazena histórico não possui a mesma estrutura que uma dimensão Tipo 2 descrita anteriormente. Apenas o atributo que representa a data de início do registro é utilizado. Além disso, a dimensão que armazena histórico não armazena a situação atual, apenas as mudanças. Em (ROSS, 2005) a dimensão que armazena histórico possui a mesma estrutura que uma dimensão Tipo 2 (apresenta os atributos para data inicial , data final e um flag corrente) e pode armazenar todo o histórico. Embora as duas abordagens apresentem a mesma semântica, suas rotinas de povoamento e atualização apresentam características bem distintas. As figuras 10 e 11 apresentam exemplos dessas duas abordagens.

Na figura 10 temos a abordagem para o Tipo 4 descrita em (SANTOS; BELO, 2011a). Nesse exemplo, são apresentados os valores em uma Dimensão de Filmes, cujos atributos que correspondem ao comportamento tipo 4 são: Nome_filme e Gênero. Em (a) é apresentado os registros para os primeiros valores na dimensão corrente. Os primeiros valores, provenientes da Área de *Staging*, serão inseridos na dimensão corrente, incluindo a data de carregamento do mesmo, representado por Dt_inicio.

Dim_Corrente_Filmes

Id_filme	Cod_filme	Nome_filme	Gênero	Dt_inicio
1	101	Titanic	Romance/ História	2011-12-29
2	102	Avatar	Ficção Científica	2011-12-29
3	103	Harry Potter e A Ordem da Fênix	Aventura	2011-12-29
4	104	Walle	Infantil / Desenho	2011-12-29

Dim_Historica_Filmes

Id_filme	Dt_inicio	Cod_filme	Nome_filme	Gênero

(a)

Dim_Corrente_Filmes

Id_filme	Cod_filme	Nome_filme	Gênero	Dt_inicio
1	101	Titanic	Romance / História	2011-12-29
2	102	Avatar	Ficção Científica	2011-12-29
3	103	Harry Potter 5	Aventura	2012-01-15
4	104	Wall-e	Infantil	2012-01-15

Dim_Historica_Filmes

Id_filme	Dt_inicio	Cod_filme	Nome_filme	Gênero
3	2011-12-29	103	Harry Potter e A Ordem da Fênix	Aventura
4	2011-12-29	104	Walle	Infantil / Desenho

(b)

Dim_Corrente_Filmes

Id_filme	Cod_filme	Nome_filme	Gênero	Dt_inicio
1	101	Titanic	Romance / História	2011-12-29
2	102	Avatar	Ficção Científica	2011-12-29
3	103	Harry Potter 5	Aventura / Ficção	2012-02-23
4	104	Wall-e	Infantil	2012-01-15
5	105	A múmia	Aventura	2012-02-23

Dim_Historica_Filmes

Id_filme	Dt_inicio	Cod_filme	Nome_filme	Gênero
3	2011-12-29	103	Harry Potter e A Ordem da Fênix	Aventura
4	2011-12-29	104	Walle	Infantil / Desenho
3	2012-01-15	103	Harry Potter 5	Aventura

(c)

Figura 10: *Tabela de dimensão de filmes com valores, baseado em (*SANTOS; BELO, 2011a*).*

Ao realizar um novo carregamento para a dimensão e identificando que alguns registros sofreram alterações, os dados antigos são armazenados na *Dim_Historica_Filmes*, e os dados mais recentes são atualizados na *Dim_Corrente_Filmes.* A chave primária (Primary Key - PK) da dimensão histórica sempre será a *Surrogate Key* (SK) mais a data de início (Dt_inicio) do registro na dimensão corrente (PK = SK + Dt_incio). Como exemplo disso, na figura 10 em (b), ao carregar dados na data de 2012-01-15, pôde-se identificar que os dados eram diferentes para os registros 3 e 4, em Id_filme, da dimensão corrente. Para o primeiro, houve alteração apenas no nome do filme. Para o segundo, as alterações seguem nos atributos do gênero e nome do filme. Seus valores antigos foram adicionados na dimensão histórica com o mesmo Id_filme e a data de início do registro antigo.

O mesmo ocorre na figura 10 em (c), onde foi atualizado o gênero do registro 3, em Id_filme da dimensão corrente e armazenado seu valor antigo na dimensão histórica. O valor do atributo que era ‘Aventura’ passou a ser ‘Aventura / Ficção’ na data 2012-02-23. Nesta data de carregamentohouve o registro de uma nova linha na dimensão corrente, de Id_filme igual a 5, no qual foi adicionado o filme ‘A múmia’.

O relacionamento da tabela de fatos com a dimensão corrente ocorre através da *Surrogate Key* (SK). A tabela dimensão de dados correntes relaciona-se, por sua vez, com a de histórico pela SK, uma vez que no histórico poderemos ter vários registos da mesma SK. Caso seja necessário ter acesso ao histórico dos atributos, cria-se uma visão (View) como resultado da união da dimensão corrente e histórica.

Na figura 11 temos o comportamento para o Tipo 4 descrita em (ROSS, 2005). O exemplo é composto por duas tabelas de Dimensões de Filmes com valores. A tabela corrente (Dim_Corrente_Filmes) utiliza o conceito do Tipo 1 para os atributos. Esse comportamento é representado, em Dim_Corrente_Filmes, pelo atributo Filme. Para a tabela que armazena o histórico, Dim_Historica_Filme, os atributos (no exemplo, representado por Filme) se comportam como sendo do Tipo 2, além de conter os atributos referentes à data inicial, data final e flag corrente.

Ao ser realizada a primeira carga, serão adicionados os valores em ambas as tabelas de dimensão: corrente e histórica. Esse registro inicial pode ser visto na figura 11 em (a). Os dados são inseridos na tabela de dimensão corrente refletindo a mesma estrutura do ambiente operacional. A tabela histórica recebe a data de inicial do registro e o valor ‘S’ para o flag corrente.

Dim_Corrente_Filmes

Id_filme	Cod_filme	Filme
1	10	Pânico
2	11	Os Vingadores
3	12	A Era do Gelo

Dim_Historica_Filmes

Id_filme	Cod_filme	Filme	Dt_inicio	Dt_fim	Fl_corrente
1	10	Pânico	2012-02-29	NULL	'S'
2	11	Os Vingadores	2012-02-29	NULL	'S'
3	12	A Era do Gelo	2012-02-29	NULL	'S'

(a)

Dim_Corrente_Filmes

Id_filme	Cod_filme	Filme
1	10	Pânico 1
2	11	Os Vingadores
3	12	A Era do Gelo 4

Dim_Historica_Filmes

Id_filme	Cod_filme	Filme	Dt_inicio	Dt_fim	Fl_corrente
1	10	Pânico	2012-02-29	2012-06-30	'N'
2	11	Os Vingadores	2012-02-29	NULL	'S'
3	12	A Era do Gelo	2012-02-29	2012-06-30	'N'
4	10	Pânico 1	2012-06-30	NULL	'S'
5	12	A Era do Gelo 4	2012-06-30	NULL	'S'

(b)

Figura 11: *Tabela de dimensão de filmes com valores, baseado em (*ROSS, 2005*).*

Na figura 11, em (b), ocorrem atualizações para os registros. O atributo Filme da dimensão corrente será atualizado refletindo os valores atuais. A alteração ocorre onde os Id_filmes são iguais a 1 e 3 e seus novos valores passam a ser, respectivamente, 'Pânico 1' e 'A Era do Gelo 4'. Para a dimensão histórica, o flag corrente dos registros antigos recebe o valor 'N' e o atributo data_final recebe o valor de data igual a 2012-06-30. Após isso, dois novos registos, de Id_filme igual a 4 e 5, são inseridos com os dados mais atuais.

2.1.4.2. Tipo 6

A necessidade de adicionar diferentes comportamentos para o histórico de atributos, em uma única dimensão, deu origem à dimensão Tipo 6. Basicamente, essa estratégia é a combinação de atributos com diferentes tipos: 1,2 e 3 (1+2+3 = 6), com o objetivo de usufruir das vantagens de cada um deles.

Na figura 12, em (a), segue um exemplo de uma dimensão de funcionários que contém atributos do Tipo 1, 2 e 3, compondo dessa forma uma dimensão do Tipo 6. O atributo "Nome" tem como comportamento, para armazenamento de histórico, o Tipo 1, "Função" o Tipo 2 e "Salario" o Tipo 3.

Id_funcionario	Cod_funcionario	Nome	Salario_original	Salario_atual	Funcao	Dt_inicio	Dt_fim	Fl_corrente
1	10	Marcos	800.00	NULL	Faxineiro	2011-12-29	NULL	'S'
2	11	Joana	680.00	NULL	Recepcionista	2011-12-29	NULL	'S'
3	12	Maria	1900.00	NULL	Programador	2011-12-29	NULL	'S'

(a)

Id_funcionario	Cod_funcionario	Nome	Salario_original	Salario_atual	Funcao	Dt_inicio	Dt_fim	Fl_corrente
1	10	Marcos	800.00	NULL	Faxineiro	2011-12-29	NULL	'S'
2	11	Joana	680.00	NULL	Recepcionista	2011-12-29	NULL	'S'
3	12	Maria Estela	1900.00	NULL	Programador	2011-12-29	NULL	'S'

(b)

Id_funcionario	Cod_funcionario	Nome	Salario_original	Salario_atual	Funcao	Dt_inicio	Dt_fim	Fl_corrente
1	10	Carlos	800.00	NULL	Faxineiro	2011-12-29	NULL	'S'
2	11	Joana	680.00	NULL	Recepcionista	2011-12-29	2012-04-12	'N'
3	12	Maria Estela	1900.00	NULL	Programador	2011-12-29	NULL	'S'
4	11	Joana	680.00	NULL	Analista de Sistemas	2012-04-12	NULL	'S'

(c)

Id_funcionario	Cod_funcionario	Nome	Salario_original	Salario_atual	Funcao	Dt_inicio	Dt_fim	Fl_corrente
1	10	Carlos	800.00	NULL	Faxineiro	2011-12-29	NULL	'S'
2	11	Joana	680.00	NULL	Recepcionista	2011-12-29	2012-04-12	'N'
3	12	Maria Estela	1900.00	NULL	Programador	2011-12-29	NULL	'S'
4	11	Joana	680.00	2300.00	Analista de Sistemas	2012-04-12	NULL	'S'

(d)

Figura 12: *Dimensão Funcionário com atributos do Tipo 1, 2 e 3.*

Em (b), da figura 12, houve alteração em um registro, no qual o Id_funcionario é igual a 3, no atributo "Nome". Como o mesmo tem comportamento do Tipo 1, então será feita a atualização do registro, sem a criação de um novo. Como pode ser observado na figura 12, em (b), o valor do atributo "Nome", que era "Maria", foi substituído pelo novo valor "Maria Estela".

Posteriormente, em (c), a atualização foi realizada no atributo "Função", de um registro, no qual o Id_funcionario é igual a 2. Como este atributo é do tipo 2, será criado um novo registro, para a inserção do novo valor.Ver figura 12, em (c). O registro de identificador 2 recebe a data final e o flag corrente é marcado como "N". O novo registro, de identificador 4, receberá o valor de "Analista de Sistemas"no atributo "Função".Os valores dos demais atributos, como não sofreram alterações, serão repetidos.

Mais uma atualização pode ser vista na figura 12 em (d).Desta vez, no atributo Salario, em um registro, no qual o Id_funcionario é igual a 4. Como este atributo é do tipo 3, o

registro de salário inicial não soferá alterações. O novo salário será armazenado no atributo “Salario_atual” o valor de “2300.00”.Desta forma, para esta dimensão de funcionário, sempre ficará armazenado o salário original do funcionário, ou seja, o primeiro salário a ele atribuído, e o salário atual.

2.1.5. Hierarquias

No projeto de uma dimensão, é interessante que seus atributos possam formar estruturas hierárquicas onde os diferentes níveis sejam traduzidos como pontos de agregação[2] sobre os quais o usuário possa explorar os fatos (ADAMSON, 2006).

Segundo a literatura, existem duas formas de criar hierarquias em uma dimensão. A primeira delas é criar hierarquias na própria dimensão. Nesta, a hierarquia é representada por diferentes tipos de registros na dimensão. Cada tipo de registro representa um nível que é identificado através de um atributo, na figura 13 representado por ‘Nivel’. A segunda forma é criar diferentes tabelas, dimensões, para representar cada nível de uma hierarquia. A figura 15 contém uma hierarquia entre dimensões para a dimensão produto (ADAMSON, 2006) (KIMBALL, 2008).

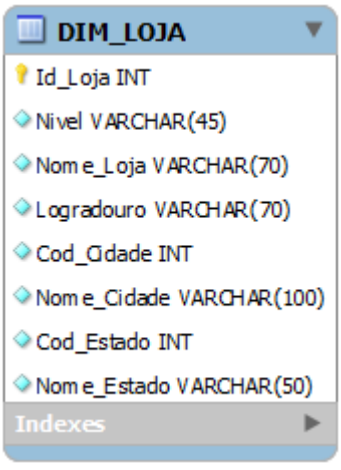


Figura 13: *Hierarquia dentro de uma Dimensão de Loja.*

[2] Um ponto de agregação descreve um potencial resumo em uma dimensão. Por exemplo, Categoria e Marca podem ser definidos como potenciais pontos de agregação dentro de uma dimensão de produto (ADAMSON, 2006).

Muitos atributos podem estar organizados em hierarquias dentro da dimensão. Na figura 13 são apresentados alguns possíveis níveis para a hierarquia de endereços, como por exemplo, Estado e Cidade. O primeiro, Estado, contém os seguintes atributos: Cod_Estado e Nome_Estado. O segundo, Cidade, é composto por Cod_Cidade e Nome_Cidade, além dos atributos de Estado. As dimensões devem conter um atributo que represente o seu nível de hierarquia. No exemplo da figura 14, o 'Nível' representa esse atributo. A carga para essa hierarquia é vista na figura 14.

Id_loja	Nivel	Nome_Loja	Logradouro	Cod_Cidade	Nome_Cidade	Cod_Estado	Nome_Estado
100	NULL	Loja Aperipê	Rua X	1	Aracaju	10	Sergipe
101	NULL	Loja Atalaia	Rua Y	1	Aracaju	10	Sergipe
102	NULL	Loja Centro	Rua Z	2	Itabaiana	10	Sergipe

(a)

Id_loja	Nivel	Nome_Loja	Logradouro	Cod_Cidade	Nome_Cidade	Cod_Estado	Nome_Estado
100	NULL	Loja Aperipê	Rua X	1	Aracaju	10	Sergipe
101	NULL	Loja Atalaia	Rua Y	1	Aracaju	10	Sergipe
102	NULL	Loja Centro	Rua Z	2	Itabaiana	10	Sergipe
103	Estado	NULL	NULL	NULL	NULL	10	Sergipe

(b)

Id_loja	Nivel	Nome_Loja	Logradouro	Cod_Cidade	Nome_Cidade	Cod_Estado	Nome_Estado
100	NULL	Loja Aperipê	Rua X	1	Aracaju	10	Sergipe
101	NULL	Loja Atalaia	Rua Y	1	Aracaju	10	Sergipe
102	NULL	Loja Centro	Rua Z	2	Itabaiana	10	Sergipe
103	Estado	NULL	NULL	NULL	NULL	10	Sergipe
104	Cidade	NULL	NULL	1	Aracaju	10	Sergipe
105	Cidade	NULL	NULL	2	Itabaiana	10	Sergipe

(c)

Figura 14: *Dimensão Loja com uma hierarquia de endereços para os níveis Cidade e Loja.*

A primeira carga da dimensão é feita em (a) da figura 14. Nesta, todos os registros são inseridos na dimensão. Após o preenchimento dos valores, serão adicionados os níveis da hierarquia. Para isso, em (b) há uma varredura em todos os dados do atributo que compõem o nível Estado (Cod_Estado e Nome_Estado), para identificar seus diferentes valores. Neste Exemplo, somente há o valor de 'Sergipe' para Estado. Logo, um novo registro (103) será adicionado à dimensão com os valores dos atributos do nível, que são: 10 e 'Sergipe' para Cod_Estado e Nome_Estado, respectivamente. Para identificar que esse registro corresponde a um estado, o atributo Nivel, receberá o valor de 'Estado'. Os demais atributos recebem a ausência de valor, por que não fazem parte do nível.

A próxima iteração será para inserir os dados do nível cidade. Em (c) pode ser visto que existem duas cidades diferentes para as lojas, são elas: Aracaju e Itabaiana. Dessa forma, esses dois registros serão adicionados na dimensão. O primeiro deles receberá 1 e

‘Aracaju’, para os atributos Cod_Cidade e Nome_Cidade, respectivamente, além dos valores correspondentes a seu estado (Sergipe) e a atualização do campo Nivel com o valor ‘Cidade’. Os outros atributos receberão os valores nulos. O segundo ocorre para o registro 105 que serão adicionados os seguintes dados: 2, ‘Itabaiana’ e ‘Cidade’; para Cod_Cidade, Nome_Cidade e Nivel, respectivamente.

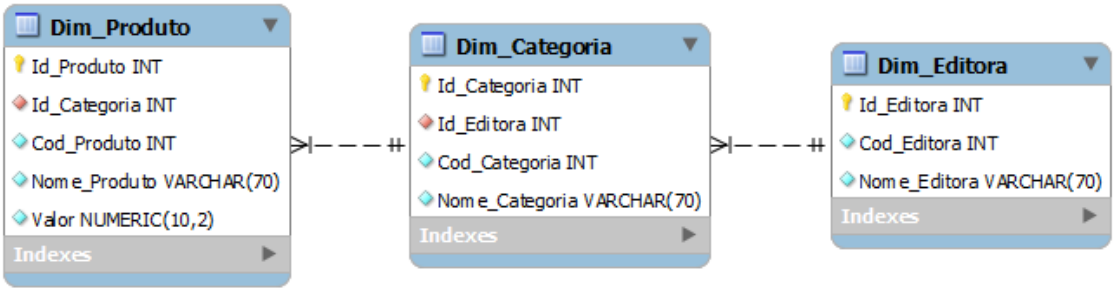
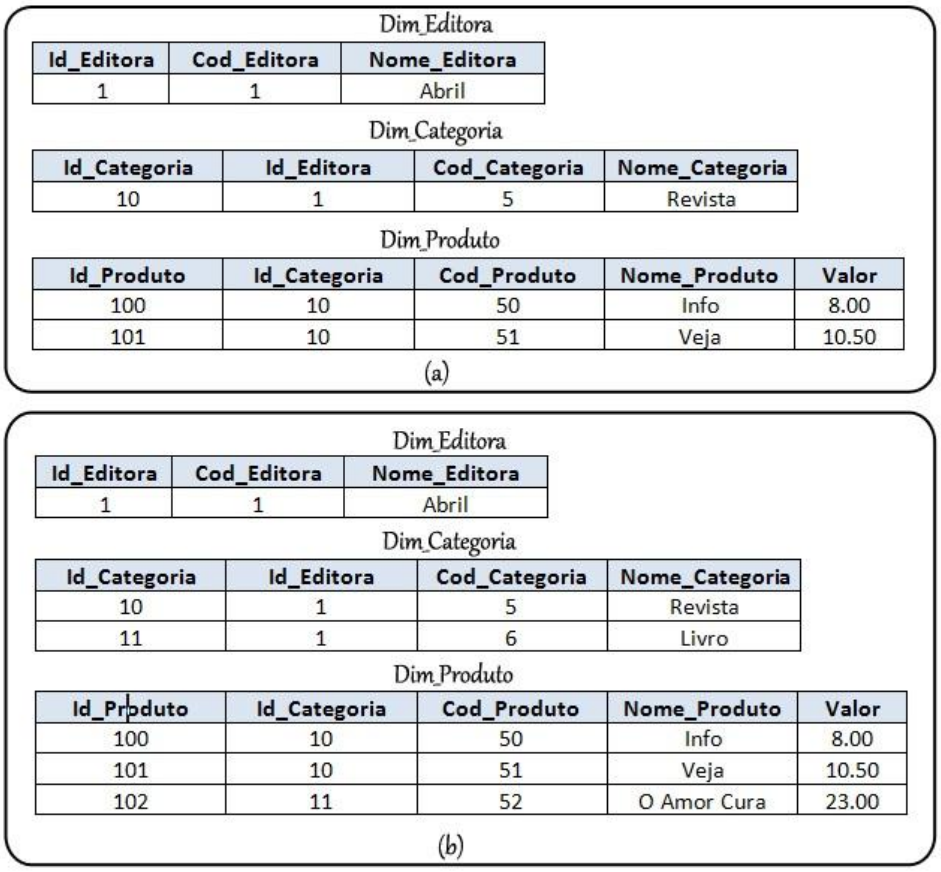


Figura 15: *Dimensão de Produto com hierarquias entre várias dimensões.*

A figura 15 retrata quando uma hierarquia pode estar contida entre dimensões. Nestas, o nível da hierarquia que possui a menor granularidade recebe a Surrogate Key da dimensão com maior granularidade, e assim por diante. Neste exemplo, a Dim_Editora possui a maior granularidade, que representa o registro de todas editoras. A Dim_Categoria recebe a Surrogate Key da editora, assim como seus atributos que representam os dados da categoria de um produto. Por fim, a Dim_Produto que representa os detalhes mais simples da dimensão de produtos, além de conter a chave da Categoria. A carga para essas dimensões pode ser vista na figura 16.

As atualizações ocorrem na medida em que a granularidade da dimensão vai diminuindo. Na figura 16, em (a), a Dim_Editora recebe os valores de 1 e ‘Abril’ para os atributos Cod_Editora e Nome_Editora, respectivamente. Em Seguida, a Dim_Categoria recebe os valores de 5 e ‘Revista’ para Cod_Categoria e Nome_Categoria, respectivamente, além de adicionar o valor da editora que o compõe representado pelo registro de Id igual a 1. Por fim, a Dim_Produto contém os dados ‘Info’ e ‘Veja’, para os dois registros da dimensão (100 e 101), em Nome_Produto. Ambos os registros de produto possuem o valor 10 em Id_Categoria, que representa a categoria ‘Revista’ à que estão associados.



Dim_Editora

Id_Editora	Cod_Editora	Nome_Editora
1	1	Abril

Dim_Categoria

Id_Categoria	Id_Editora	Cod_Categoria	Nome_Categoria
10	1	5	Revista

Dim_Produto

Id_Produto	Id_Categoria	Cod_Produto	Nome_Produto	Valor
100	10	50	Info	8.00
101	10	51	Veja	10.50

(a)

Dim_Editora

Id_Editora	Cod_Editora	Nome_Editora
1	1	Abril

Dim_Categoria

Id_Categoria	Id_Editora	Cod_Categoria	Nome_Categoria
10	1	5	Revista
11	1	6	Livro

Dim_Produto

Id_Produto	Id_Categoria	Cod_Produto	Nome_Produto	Valor
100	10	50	Info	8.00
101	10	51	Veja	10.50
102	11	52	O Amor Cura	23.00

(b)

Figura 16: *Dimensão de Produto que possui hierarquia, com os níveis Categoria e Editora, respectivamente, para as dimensões Dim_Categoria e Dim_Editora.*

Em (b) da figura 16, ocorre uma nova atualização para a hierarquia na dimensão de produtos. Em Dim_Editora não houve inserção ou alteração de seus valores. Na Dim_Categoria um novo registro, de Id_Categoria igual a 11, foi adicionado. Seus valores são 1, 6 e 'Livro' para os atributos Id_editora, Cod_Categoria e Nome_Categoria, respectivamente. Em Dim_Produto foi adicionado o livro (referente ao Id_Categoria igual a 11) 'O Amor Cura'.

2.1.6. Agregados

A agregação é o processo de resumir dados granulares ou detalhados e de armazenar fisicamente as agregações na tabela de fatos ao longo de uma hierarquia. Como a sumarização dos dados é uma redundância por excelência, os agregados têm um efeito muito significativo sobre o desempenho (KIMBALL, 2008) (COLAÇO, 2004).

Para definir quais agregados devam ser criados, deve-se considerar o ambiente, visando quais consultas, nas tabelas de fato, serão mais utilizadas e as mais críticas. Após definidos os agregados, o processo de agregação consiste em criar novas tabelas com os dados da tabela de fatos, mas alterando a granularidade da mesma, gerando assim, tabelas menores com dados resumidos.

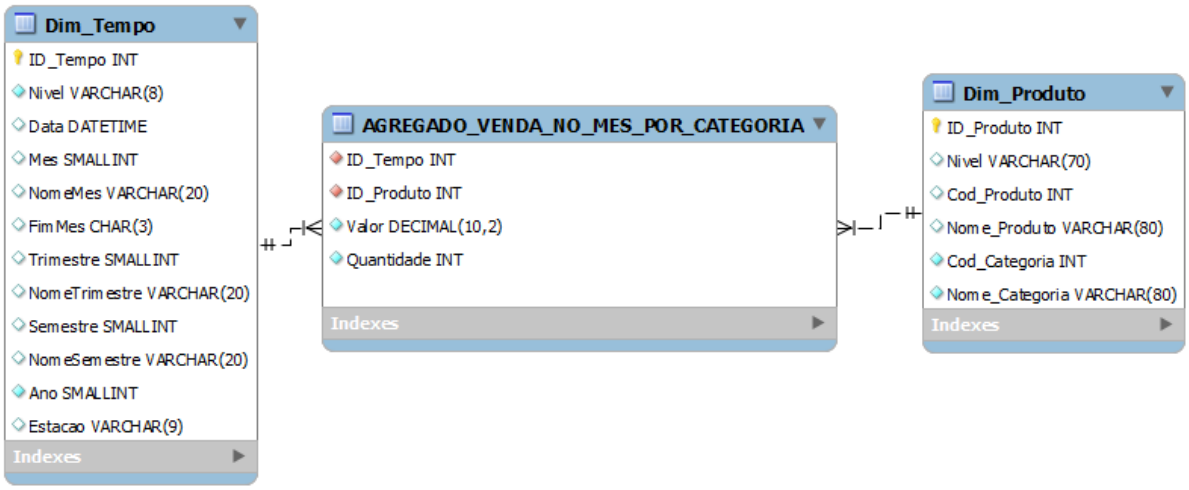


Figura 17: *Exemplo de um Esquema Agregado (Tabela agregado de venda no mês por categoria).*

O agregado e as dimensões que o compõe são organizados em um novo esquema estrela no *Data Warehouse* (KIMBALL, 2008) (ADAMSON, 2006). A figura 17 é um exemplo de uma agregação para a tabela de fatos de venda de um produto. Este agregado representa o faturamento das vendas, num determinado mês, por uma categoria de produto. As ligações com as dimensões são representadas pelos atributos ID_Tempo e ID_Produto. As medidas quantitativas, provenientes da tabela de fato, para o agregado são representadas pelos atributos Quantidade e Valor. O conceito de agregados está vinculado ao conceito de hierarquias, uma vez que estas podem estar inseridas nos esquemas de agregados com granularidades diferentes para as dimensões (ADAMSON, 2006).

Em uma hierarquia, representada por várias tabelas, a dimensão que representa um determinado nível pode estar contida no agregado. Como exemplo disso, a figura 18 apresenta uma tabela de agregado das vendas das categorias de produtos, num determinado mês, por diversos clientes. O mesmo pode ocorrer para hierarquias na mesma dimensão, como pode ser visto na figura 17. O atributo Nivel, na imagem, é quem define o nível da hierarquia da dimensão. Logo, neste deve estar contido a que nível o agregado está vinculado.

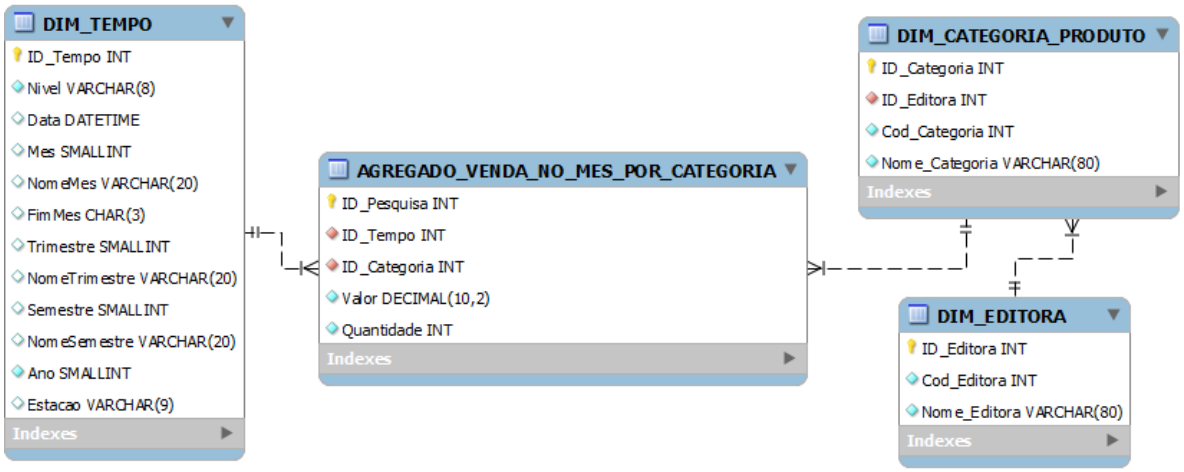


Figura 18: *Exemplo de uma tabela de agregado de venda no mês por categoria de produto.*

2.2. Trabalhos relacionados

Nesta seção apresentaremos algumas propostas que possuem objetivo semelhante ao nosso trabalho: automatizar processos ETL. As propostas apresentadas representam trabalhos moderadamente relacionados. Não foram encontrados, até o momento, trabalhos fortemente relacionados que busquem gerar automaticamente rotinas de povoamento em dialetos SQL e que apresentem metadados que possam capturar a semântica necessária para essa geração.

2.2.1. Automatic generation of ETL processes from conceptual models.

Em (MUNOS; MAZÓN; TRUJILLO, 2009) é proposto um framework para o desenvolvimento de processos ETL. Esse framework é baseado na ideia de desenvolvimento orientado a modelos, *Model Driven Architecture* (MDA) (OMG, 2003). A ideia é desenvolver um processo ou aplicação através de uma especificação independente de tecnologia. Isso é alcançado na MDA através da utilização de vários tipos de modelos e mapeamentos entre esses modelos. Temos um modelo da aplicação ou processo independente de tecnologia, chamado de PIM (*Plataform Independent Model*), que pode ser transformado em modelos dependentes de tecnologia, PSM (*Plataform Specific Model*). O PSM, embora específico, ainda é um modelo da aplicação ou processo em um nível

relativamente alto de abstração, mas já pode ser transformado em código por algum processo automático. Em resumo, o objetivo é conseguir especificar um processo, independente de tecnologia, e mais tarde decidir qual a tecnologia especifica que será utilizada para implementar aquele processo. A proposta de (MUNOS; MAZÓN; TRUJILLO, 2009) busca aumentar a produtividade e diminuir o tempo e custos com desenvolvimento e manutenção de processos ETL.

2.2.2. Pygrametl: A powerful programming framework for extract–transform–load programmers

Em (THOMSEN; PEDERSEN, 2009) é proposto um framework para o desenvolvimento de processos ETL utilizando programação ao invés de ferramentas com interface gráfica. O framework é baseado em Python. Os autores apresentam cenários nos quais a programação de processos ETL pode ser realizada melhor através de programação. Alguns desses cenários são: a) Com uma interface gráfica você pode fazer transformações simples, mas transformações complexas exigem um esforço e criatividade muito grande; b) Algumas transformações podem ser resolvidas com apenas algumas linhas de código. c) Integrações são resolvidas mais facilmente com linhas de código (com a liberdade de programação). Como existe muito código comum nos processos ETL, é proposto um framework para ajudar no desenvolvimento. O framework apresenta procedimentos prontos para alguns processos bastante conhecidos como a carga de dimensões e fatos. Os tipos de dimensões suportados são 1 e 2 apenas. Os autores compararam o framework com uma única ferramenta (Pentaho) e afirmam que conseguiram reduzir o tempo de desenvolvimento.

3. FERRAMENTA DE GERAÇÃO AUTOMÁTICA DE CÓDIGO

A elaboração da ferramenta foi realizada em duas principais etapas. A primeira delas teve como objetivo definir os metadados necessários para gerar o código sql. Na segunda etapa foi realizada a construção da ferramenta, que foi desenvolvida de modo iterativo e incremental.

3.1. Definição de Metadados

Nesta seção são apresentados os passos que foram utilizados para a definição dos metadados necessários para a geração automática de código. Cada subsistema etl (histórico em dimensões, hierarquias e agregados) recebeu o seguinte tratamento: a) Inicialmente foi definido o fluxo que traduz as atividades ou tarefas necessárias para o povoamento de dados; b) Com base no fluxo de dados, foi criado um pseudocódigo para representar o código que deveria ser gerado pela ferramenta; c) Com base no pseudocódigo e nas suas possíveis variações, foi realizada uma análise para identificar os metadados necessários.

3.1.1. Tipo 1

A figura 19 representa o fluxo do algoritmo para atributos de uma dimensão com comportamento do tipo 1.

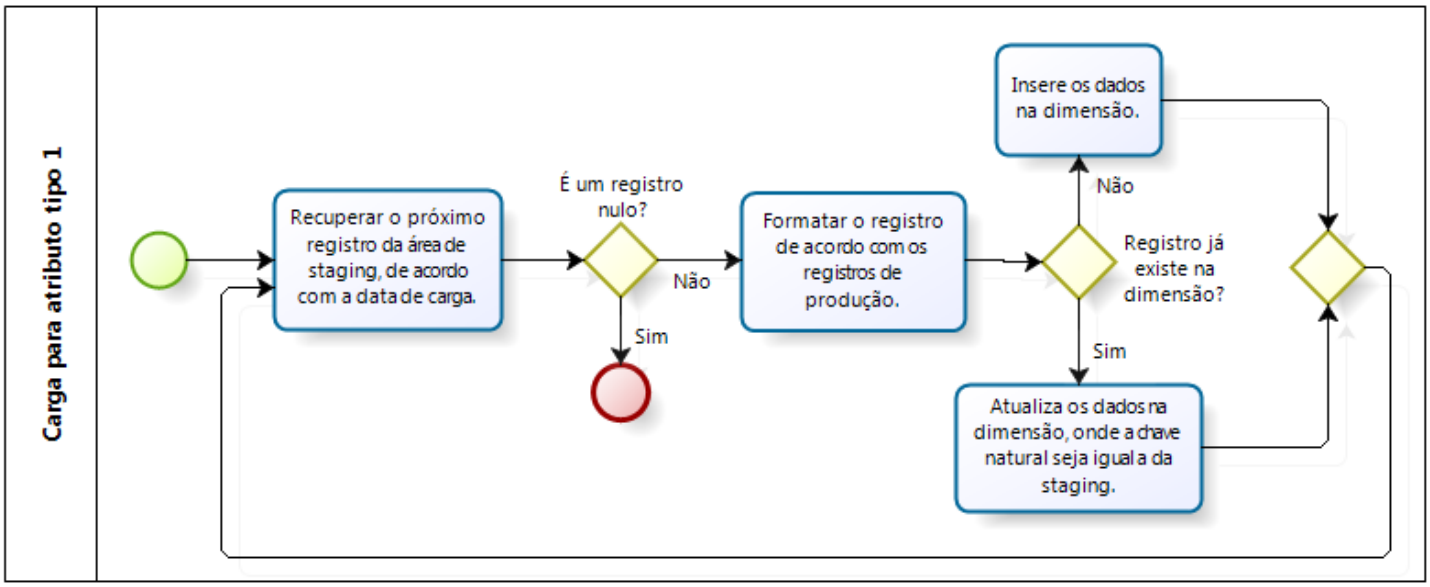


Figura 19: *Fluxograma do algoritmo de povoamento para atributos do tipo 1.*

Inicialmente, o povoamento busca recuperar o registro da Área de *Staging* de acordo com sua data de carregamento. Ao obtê-lo verifica se esse registro é nulo. Caso afirmativo, não há mais atributos na tabela da Área de *Staging* e o carregamento é encerrado. Se negativo, irá ser formatado o registro para ser inserido na dimensão. Nesse momento é feita uma verificação para saber se o registro já está contido na dimensão. Se já estiver, os dados da dimensão são atualizados de acordo com o registro examinado da Área de *Staging*. Caso contrário, insere o registro na dimensão.

A figura 20 contém um pseudocódigo para carga de atributos do tipo 1. Na primeira linha é apresentada a assinatura do procedimento. Nessa assinatura podemos observar a utilização de um parâmetro formal que representa a data utilizada com restrição para recuperação de dados na Área de *Staging*. Em seguida, na linha 3, há a declaração de variáveis que condizem com os atributos da tabela auxiliar. Em 5 há um laço de repetição que será verdadeiro enquanto houver registros na Área de *Staging* onde a data de carga seja igual a @DATACARGA. Para todo registro examinado pelo laço, o procedimento verifica, na linha 9, se esse já está contido na dimensão. Se verdadeiro, atualiza o registro com os dados mais recentes, de acordo com sua chave natural. Caso contrário, os dados serão inseridos na dimensão.

```
1  CRIAR PROCEDIMENTO Nome_procedimento (@DATACARGA DATETIME)
2  INÍCIO PROCEDIMENTO
3      DECLARAR VARIÁVEIS QUE CORRESPONDEM AOS ATRIBUTOS DA TABELA AUXILIAR
4
5      PARA TODA LINHA DA TABELA AUXILIAR CUJA DATA DE CARGA É IGUAL À DATA PASSADA COMO PARÂMETRO
6      FAÇA
7          INÍCIO
8
9              SE EXISTIR O REGISTRO ATUAL NA DIMENSÃO
10             ENTÃO
11                 ATUALIZA O REGISTRO DA DIMENSAO COM OS DADOS DA TABELA AUXILIAR
12                 DE ACORDO COM SUA CHAVE NATURAL
13             SENÃO
14                 INSERE O REGISTRO, DA TABELA AUXILIAR, NA DIMENSÃO
15
16         FIM
17
18 FIM PROCEDIMENTO
```

Figura 20: *Pseudocódigo do procedimento para povoamento de atributo tipo 1.*

Na primeira linha da figura 20 é definida a criação do procedimento passando como parâmetro a data de carga. O nome dado ao procedimento e o esquema a que pertence são os primeiros metadados identificados. Em 3 é declarada as variáveis que condizem aos atributos da tabela auxiliar. Os metadados que compõem esta tabela e seus atributos são explicados mais adiante, assim como os elementos que formam as dimensões, essenciais para atualização de seus registros, como visto nas linhas 11 e 12.

Em uma tabela auxiliar, os metadados fundamentais são o nome e esquema, assim como os atributos que a constituem. Esses atributos são compostos por: chave natural; nome do atributo; tipo dos dados; tamanho; precisão; escala; restrição de nulidade; atributo que representa data de carregamento. Para a dimensão, os metadados são: *Surrogate Key*, chave natural; nome do atributo; tipo dos dados; tamanho; precisão; escala e restrição de nulidade. Este último, contido em ambas as tabelas serve para indicar se o atributo é ou não um registro nulo.

A dimensão e a tabela auxiliar são cruciais para o funcionamento do procedimento. Dessa forma, este irá conter essas duas tabelas, assim como seus relacionamentos. Afinal, é necessário identificar qual atributo da tabela auxiliar corresponde ao atributo da dimensão, para realizar as atualizações dos registros.

Para contemplar esses metadados, foi criado um esquema conceitual que pode ser visto na figura 21. Nesta, em (a), estão contidos os metadados para o procedimento. Por fim, em (b) e (c) estão os metadados da tabela auxiliar e dimensão, respectivamente.

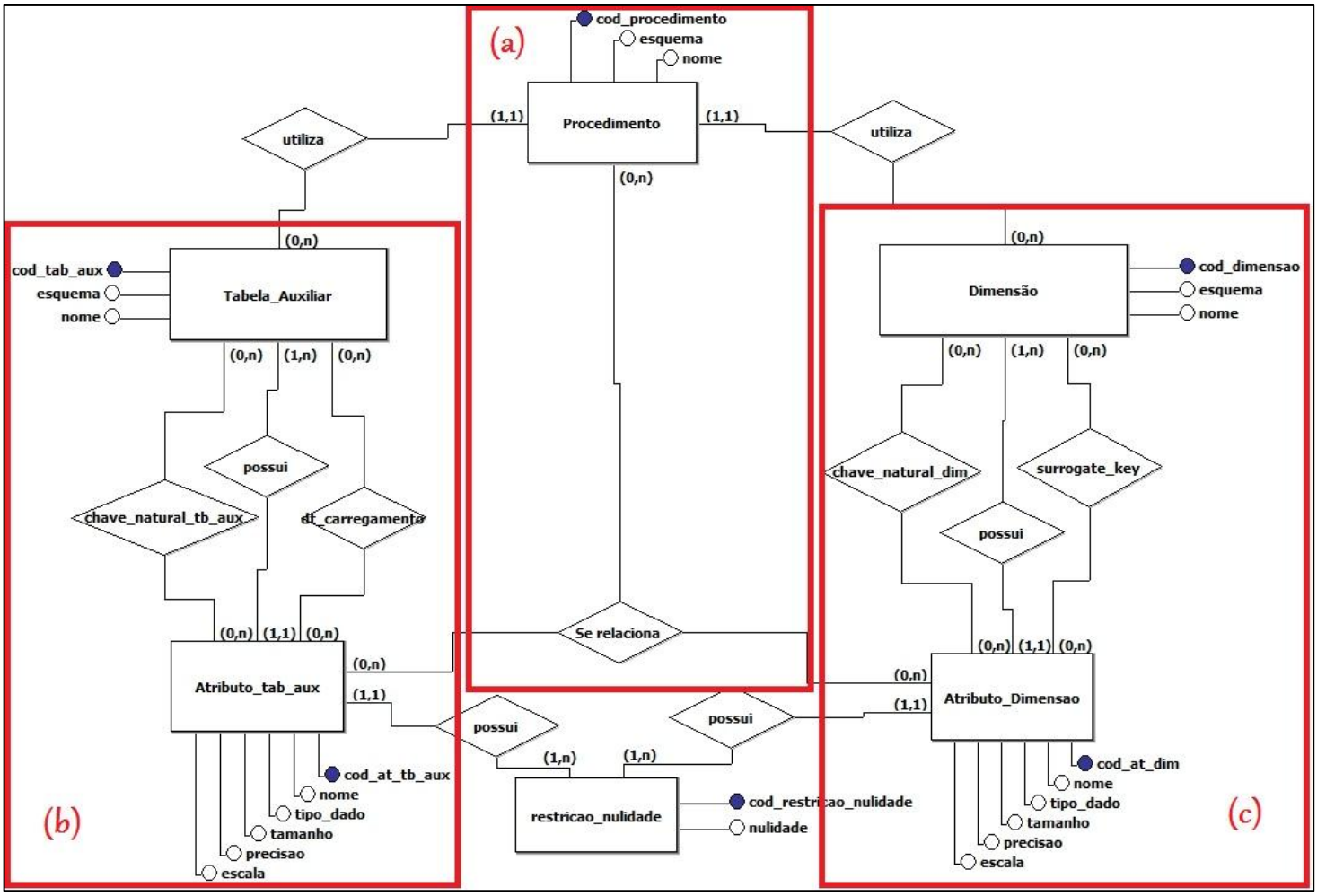


Figura 21: *Esquema Conceitual dos metadados – para o procedimento, tabela auxiliar e dimensão – identificados para o povoamento de atributo tipo 1.*

3.1.2. Tipo 2

O fluxograma para o algoritmo de povoamento de atributos do tipo 2 pode ser visto na figura 22.

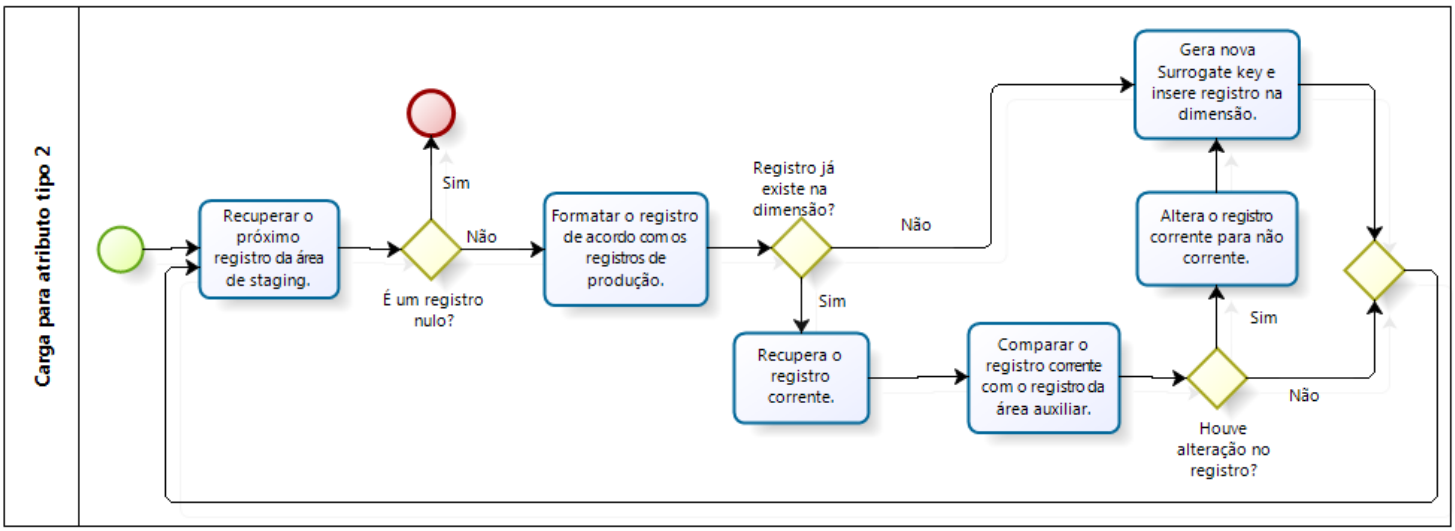


Figura 22: *Fluxograma do algoritmo de povoamento para atributos do tipo 2.*

Assim como no fluxograma anterior, serão recuperados e preparados os registros provenientes da Área de *Staging*. Se o registro não estiver contido na dimensão, então é gerada uma nova linha para a mesma, contendo os dados provenientes da tabela auxiliar, e retornará para o começo do fluxo. Caso contrário, compara os dados correntes da dimensão com o da área auxiliar. Se houve alteração, o procedimento gera um novo registro corrente para a dimensão e o antigo recebe o valor de não corrente. Caso o registro tenha se mantido o mesmo, então irá para o início do fluxo.

```
1  CRIAR PROCEDIMENTO Nome_procedimento (@DATACARGA DATETIME)
2  INÍCIO PROCEDIMENTO
3      DECLARAR VARIÁVEIS QUE CORRESPONDEM AOS ATRIBUTOS DA TABELA AUXILIAR
4
5      PARA TODA LINHA DA TABELA AUXILIAR CUJA DATA DE CARGA É IGUAL À DATA PASSADA COMO PARÂMETRO
6      FAÇA
7          INÍCIO
8
9              SE O REGISTRO DA TABELA AUXILIAR JÁ ESTIVER NA DIMENSÃO
10             ENTÃO
11                 INÍCIO ENTÃO
12                     SE O REGISTRO CORRENTE DA DIMENSÃO FOR DIFERENTE DO REGISTRO
13                     PROVENIENTE DA TABELA AUXILIAR DE ACORDO COM SUA CHAVE NATURAL.
14                     ENTÃO
15                         ALTERA O REGISTRO CORRENTE DA DIMENSÃO PARA NÃO CORRENTE
16                         E INSERE O NOVO REGISTRO, DA TABELA AUXILIAR, NA DIMENSÃO.
17
18                 FIM ENTÃO
19             SENÃO
20                 INSERE O REGISTRO, DA TABELA AUXILIAR, NA DIMENSÃO.
21
22         FIM
23
24 FIM PROCEDIMENTO
```

Figura 23: *Pseudocódigo do procedimento para povoamento de atributos do tipo 2.*

A figura 23 apresenta um pseudocódigo de um procedimento para povoamento de atributos do tipo 2. A diferença desse código para o da figura 20 são as linhas de 9 a 20. Em 9 verifica se o registro, proveniente da tabela auxiliar, já se encontra na dimensão. Caso não tenha o registro, então insere o novo na dimensão (linha 20). Caso esteja contido, verifica se registro corrente da dimensão é diferente do novo registro, que é proveniente da tabela auxiliar. Se for diferente, altera o registro corrente da dimensão para não corrente e insere o novo como sendo corrente, visto nas linhas 15 e 16.

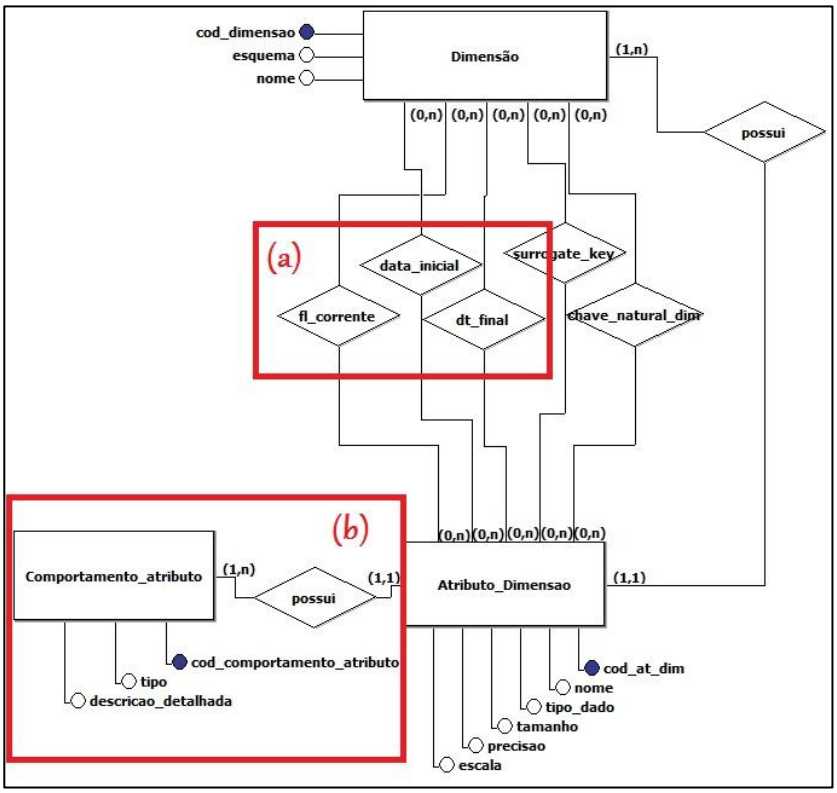


Figura 24: *Esquema Conceitual dos metadados identificados, que contemplam a dimensão, para o povoamento de atributo tipo 2.*

Os metadados da dimensão que contemplam o povoamento de uma dimensão tipo 2 são – além dos identificados no tipo 1 – os seguintes: data inicial, data final e flag corrente (vistos em (a) da figura 24); comportamento do atributo. Este último surgiu a partir do momento que houve a necessidade de coletar os metadados para os diversos tipos de povoamento em uma dimensão. Logo, tornou-se necessário criar uma entidade que armazenasse o comportamento do atributo, que corresponde ao tipo 0, 1, 2 ou 3. Essa entidade pode ser vista em (b) da figura 24.

3.1.3. Tipo 3

Na figura 25, está contido o fluxograma do algoritmo para povoamento de atributos do tipo 3 da dimensão.

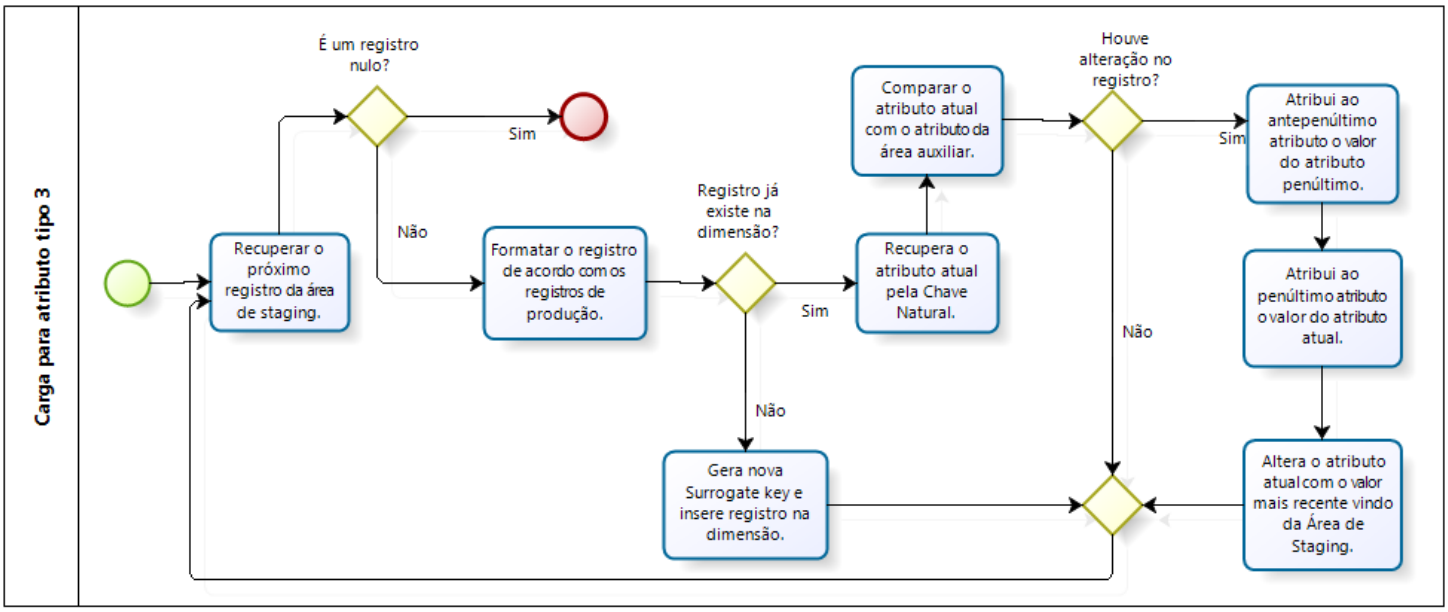


Figura 25: *Fluxograma do algoritmo para povoamento de atributos do tipo 3.*

Inicialmente, o fluxo para povoamento busca recuperar e formatar os registros não nulos provenientes da Área de *Staging*. Posteriormente, verifica se o registro já se encontra na dimensão. Se não existir, insere o novo registro na dimensão e recupera o próximo registro da área auxiliar. Caso contenha, recupera o registro atual pela chave natural da dimensão e compara com o da *Staging*. Se não tiver alteração, irá retornar para o início do fluxo. Caso contrário, ocorrerá a atualização dos atributos que correspondem ao tipo 3. O atributo antepenúltimo atribui o valor do penúltimo que receberá o do atual, e por fim, este último altera seu valor para o mais recente vindo da tabela auxiliar.

O código gerado para este tipo de povoamento pode ser visto na figura 26. Após a criação do procedimento e da declaração de variáveis, o código verifica para toda linha da tabela auxiliar cuja data de carga é igual à data passada por parâmetro. Caso seja verdadeiro, confere se o registro da tabela auxiliar já está contido na dimensão (linha 9). Se já estiver, atribui os valores que possuem comportamento tipo 3 às variáveis do procedimento. Após isso, na linha 15, compara o registro atual do atributo tipo 3 com o valor atual da tabela

auxiliar. Se for diferente, atualiza os registros que correspondem ao tipo 3 de acordo com os novos registros.

```
CRIAR PROCEDIMENTO Nome_procedimento (@DATACARGA DATETIME)
INÍCIO PROCEDIMENTO
    DECLARAR VARIÁVEIS QUE CORRESPONDEM AOS ATRIBUTOS DA TABELA AUXILIAR

    PARA TODA LINHA DA TABELA AUXILIAR CUJA DATA DE CARGA É IGUAL À DATA PASSADA COMO PARÂMETRO
    FAÇA
        INÍCIO

            SE O REGISTRO DA TABELA AUXILIAR JÁ ESTIVER NA DIMENSÃO
            ENTÃO
                INÍCIO ENTÃO
                    ATRIBUIR ÀS VARIÁVEIS OS VALORES QUE POSSUEM COMPORTAMENTO TIPO 3,
                    A PARTIR DA CHAVE NATURAL DA TABELA AUXILIAR.

                    SE O REGISTRO ATUAL DO ATRIBUTO TIPO 3 DA DIMENSÃO FOR DIFERENTE DO REGISTRO
                    PROVENIENTE DA TABELA AUXILIAR DE ACORDO COM SUA CHAVE NATURAL.
                    ENTÃO
                        ATUALIZA OS REGISTROS QUE CORRESPONDEM AO TIPO 3
                        DE ACORDO COM A TABELA AUXILIAR.

                FIM ENTÃO
            SENÃO
                INSERE O REGISTRO, DA TABELA AUXILIAR, NA DIMENSÃO.

        FIM

FIM PROCEDIMENTO
```

Figura 26: *Pseudocódigo do procedimento para povoamento de atributo do tipo 3.*

Com base no código, foi identificado que para conceber os metadados para o atributo tipo 3, seria necessário adicionar uma entidade que servisse para armazenar os atributos atual, penúltimo e antepenúltimo de uma dimensão. O nome dessa entidade (visto em vermelho na figura 27) é a especificação do atributo tipo 3.

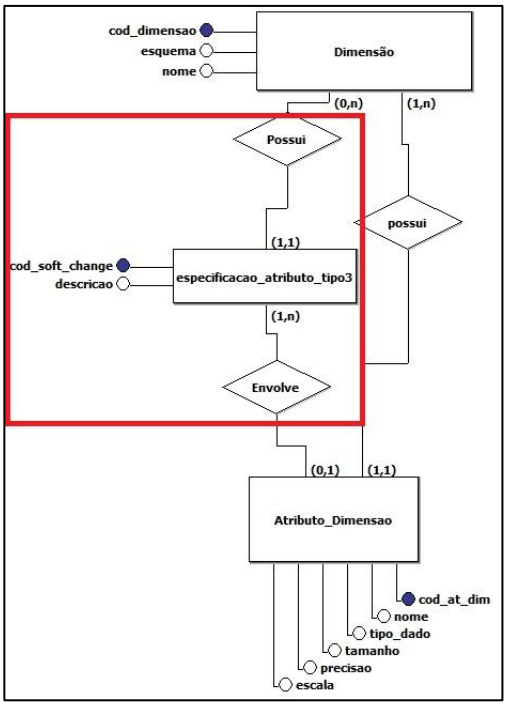


Figura 27: *Esquema Conceitual dos metadados identificados, que contemplam a dimensão, para o povoamento de atributo do tipo 3.*

3.1.4. Tipo 0

No início do fluxo para carga do tipo 0 (figura 28), o procedimento busca um registro na tabela da *Staging* de acordo com sua data de carregamento. Após isso, verifica se esse é um registro com valores nulos. Caso não seja, o procedimento preparara o dado e irá conferir se o registro já existe na dimensão. Se não possuir o valor, este irá ser persistido na dimensão. Caso já possua, o procedimento ignora a atualização e buscará recuperar o próximo registro da tabela auxiliar.

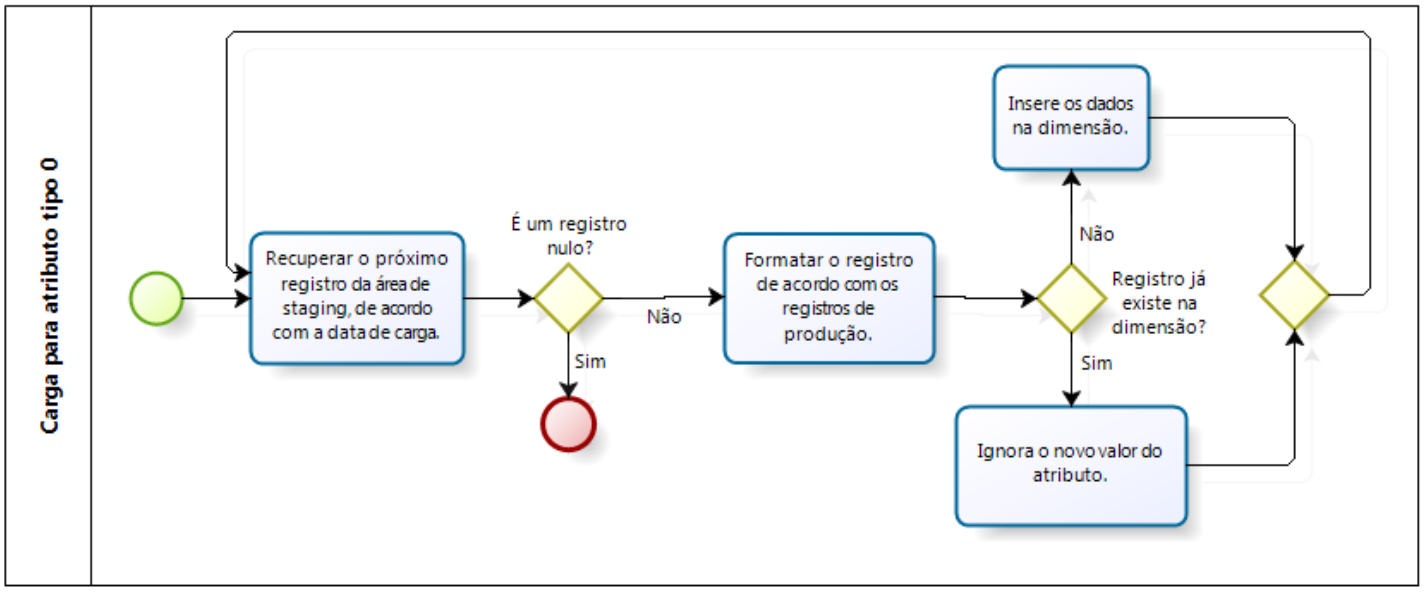


Figura 28: *Fluxograma do algoritmo para povoamento de atributo do tipo 0.*

Diante do que pode ser analisado no fluxo desse tipo do atributo, foi criado o pseudocódigo da figura 29. Nesse, a cada iteração, que ocorre na linha 5, o código verifica se o registro já foi inserido na dimensão. Caso não esteja, o novo registro vindo da área auxiliar será inserido na dimensão (linha 11). Caso contrário, esse registro será ignorado. Visto que esse trecho do código já se encontra nos demais pseudocódigos apresentados anteriormente, não houve descoberta de novos metadados para o projeto.

```
1  CRIAR PROCEDIMENTO Nome_procedimento (@DATACARGA DATETIME)
2  INÍCIO PROCEDIMENTO
3      DECLARAR VARIÁVEIS QUE CORRESPONDEM AOS ATRIBUTOS DA TABELA AUXILIAR
4
5      PARA TODA LINHA DA TABELA AUXILIAR CUJA DATA DE CARGA É IGUAL À DATA PASSADA COMO PARÂMETRO
6      FAÇA
7          INÍCIO
8
9              SE O REGISTRO DA TABELA AUXILIAR NÃO ESTIVER NA DIMENSÃO
10             ENTÃO
11                 INSERE O REGISTRO, DA TABELA AUXILIAR, NA DIMENSÃO.
12
13         FIM
14
15 FIM PROCEDIMENTO
```

Figura 29: *Pseudocódigo do procedimento para povoamento de atributos do tipo 0.*

3.1.5. Tipo 4

A figura 30 mostra, no fluxograma, como funciona a técnica para povoamento em uma dimensão do tipo 4. Após recuperar o registro atual da Área de *Staging* e formatá-los, o procedimento de carga para uma dimensão tipo 4 será dividido em dois fluxos distintos e paralelos. O primeiro deles serve para efetuar a carga dos atributos tipo 1 de uma dimensão corrente. A segunda para fazer o povoamento para uma dimensão com atributos do tipo 2. Após o termino de atualização das tabelas, reinicia todo o fluxo.

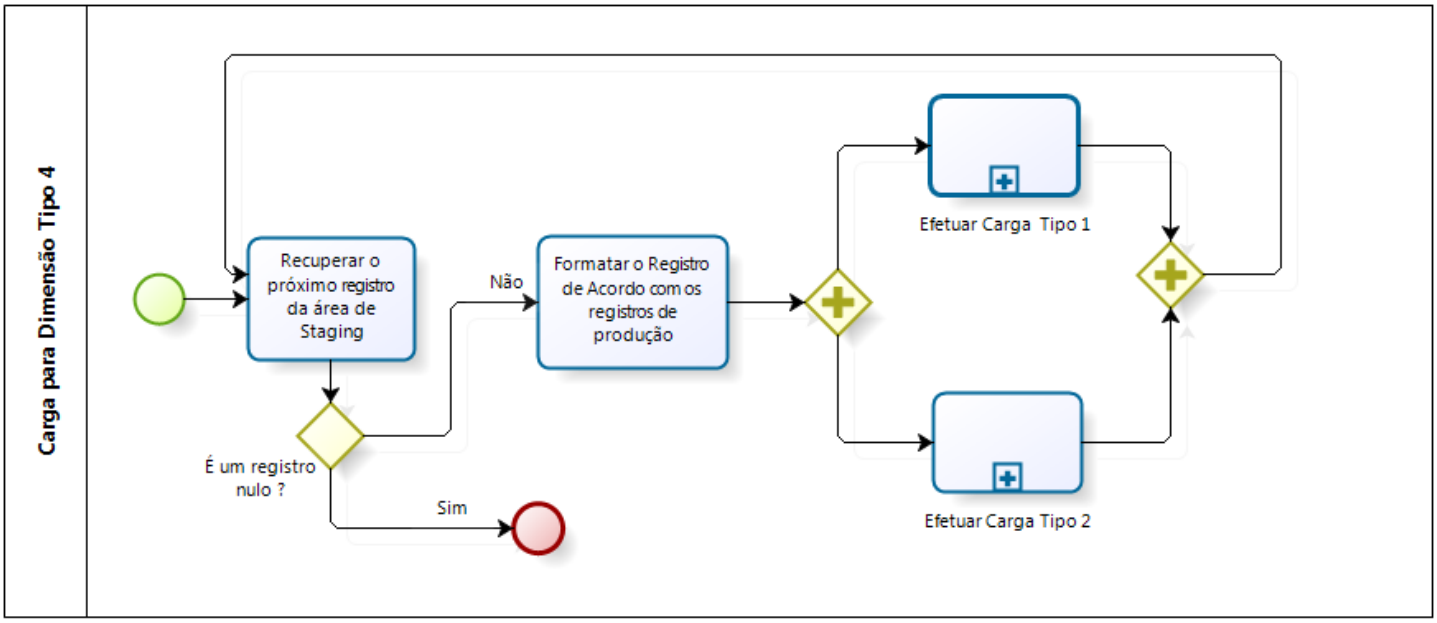


Figura 30: *Fluxograma do algoritmo para povoamento de dimensão tipo 4.*

Esses subprocessos, que ocorrem em paralelo no fluxo da figura 30, são similares aos fluxos apresentados nas seções de atributos do tipo 1 e 2, respectivamente representados nas figuras 1 e 2 do Apêndice I.

Na figura 31 é visto o pseudocódigo para esse tipo de povoamento. Para todo registro da tabela auxiliar é efetuado a carga do tipo 1 para a dimensão que não possui histórico (linhas 12 a 17) e em seguida para a dimensão com histórico (linhas 22 a 33).

```
CRIAR PROCEDIMENTO Nome_procedimento (@DATACARGA DATETIME)
INÍCIO PROCEDIMENTO
    DECLARAR VARIÁVEIS QUE CORRESPONDEM AOS ATRIBUTOS DA TABELA AUXILIAR

    PARA TODA LINHA DA TABELA AUXILIAR CUJA DATA DE CARGA É IGUAL À DATA PASSADA COMO PARÂMETRO
    FAÇA
        INÍCIO

            /**
            *   EFETUAR CARGA TIPO 1
            */
            SE EXISTIR O REGISTRO ATUAL NA DIMENSÃO
            ENTÃO
                ATUALIZA O REGISTRO DA DIMENSAO COM OS DADOS DA TABELA AUXILIAR
                DE ACORDO COM SUA CHAVE NATURAL
            SENÃO
                INSERE O REGISTRO, DA TABELA AUXILIAR, NA DIMENSÃO

            /**
            *   EFETUAR CARGA TIPO 2
            */
            SE O REGISTRO DA TABELA AUXILIAR JÁ ESTIVER NA DIMENSÃO
            ENTÃO
                INÍCIO ENTÃO
                    SE O REGISTRO CORRENTE DA DIMENSÃO FOR DIFERENTE DO REGISTRO
                    PROVENIENTE DA TABELA AUXILIAR DE ACORDO COM SUA CHAVE NATURAL.
                    ENTÃO
                        ALTERA O REGISTRO CORRENTE DA DIMENSÃO PARA NÃO CORRENTE
                        E INSERE O NOVO REGISTRO, DA TABELA AUXILIAR, NA DIMENSÃO.

                FIM ENTÃO
            SENÃO
                INSERE O REGISTRO, DA TABELA AUXILIAR, NA DIMENSÃO.
        FIM
FIM PROCEDIMENTO
```

Figura 31: *Pseudocódigo do procedimento para povoamento de dimensão tipo 4.*

O desafio para o código e definição dos metadados, desse tipo de dimensão, é que deve possuir a carga para diferentes dimensões. Portanto, os metadados aqui identificados são contemplados com as cargas para as dimensões tipo 1 e 2.

3.1.6. Tipo 6

O fluxo para a técnica da dimensão tipo 6 é apresentado na figura 32. Ao carregar e formatar um novo registro da tabela auxiliar, não nulo, é verificado o tipo de seu comportamento. A partir disso é efetivada a carga de seu comportamento que pode ser do tipo 1, 2 ou 3. Ao final da atualização do registro, o fluxo retorna para o início, recuperando o próximo registro na *Staging*.

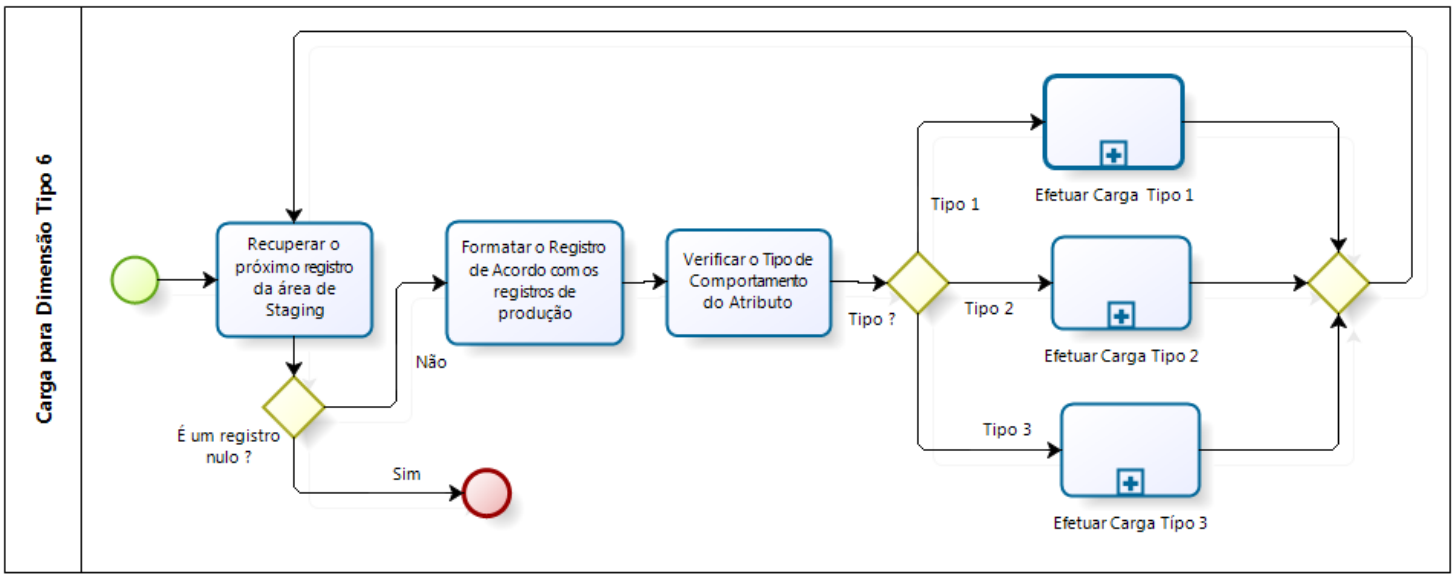


Figura 32: *Fluxograma do algoritmo para povoamento de dimensão tipo 6.*

Diferente do tipo 4 (que as cargas ocorriam para os tipos 1 e 2 em paralelo), as cargas para a dimensão tipo 6 ocorrem de forma exclusivas, ou seja, se o novo registro for do tipo 1, será efetivado o povoamento somente para esse tipo. Após isso, retorna para o início do fluxo. O subprocesso para a carga do tipo 3 pode ser vista na figura 1 do Apêndice II. Os outros dois primeiro, no Apêndice I.

O código de povoamento para a dimensão é visto na figura 33. Para todo novo registro da *Staging*, verifica se há valores para os atributos do tipo 1, 2 e 3 respectivamente nas linhas 16, 25 e 37. Caso haja, irão ocorrer as atualizações necessárias para os atributos. Os metadados, para esse tipo de dimensão são contemplados com as cargas para os atributos do tipo 1, 2 e 3.

```
CRIAR PROCEDIMENTO Nome_procedimento (@DATACARGA DATETIME)
INÍCIO PROCEDIMENTO
    DECLARAR VARIÁVEIS QUE CORRESPONDEM AOS ATRIBUTOS DA TABELA AUXILIAR

    PARA TODA LINHA DA TABELA AUXILIAR CUJA DATA DE CARGA É IGUAL À DATA PASSADA COMO PARÂMETRO
    FAÇA
        INÍCIO

            SE EXISTIR O REGISTRO ATUAL NA DIMENSÃO
            ENTÃO
                INÍCIO ENTÃO

                    /**
                    *   ATUALIZA OS DADOS PARA OS ATRIBUTOS TIPO 1
                    */
                    SE HOUVER ATRIBUTOS DO TIPO 1
                    ENTÃO
                        ATUALIZA O REGISTRO DA DIMENSAO COM OS DADOS DA TABELA AUXILIAR
                        DE ACORDO COM SUA CHAVE NATURAL
                    FIM ENTÃO

                    /**
                    *   ATUALIZA OS DADOS PARA OS ATRIBUTOS TIPO 2
                    */
                    SE HOUVER ATRIBUTOS DO TIPO 2
                    ENTÃO
                        SE O REGISTRO CORRENTE DA DIMENSÃO FOR DIFERENTE DO REGISTRO
                        PROVENIENTE DA TABELA AUXILIAR DE ACORDO COM SUA CHAVE NATURAL.
                            ENTÃO
                                ALTERA O REGISTRO CORRENTE DA DIMENSÃO PARA NÃO CORRENTE
                                E INSERE O NOVO REGISTRO, DA TABELA AUXILIAR, NA DIMENSÃO.
                    FIM ENTÃO

                    /**
                    *   ATUALIZA OS DADOS PARA OS ATRIBUTOS TIPO 3
                    */
                    SE HOUVER ATRIBUTOS DO TIPO 3
                    ENTÃO
                        ATRIBUIR ÀS VARIÁVEIS OS VALORES QUE POSSUEM COMPORTAMENTO TIPO 3,
                        A PARTIR DA CHAVE NATURAL DA TABELA AUXILIAR.

                        SE O REGISTRO ATUAL DO ATRIBUTO TIPO 3 DA DIMENSÃO FOR DIFERENTE DO REGISTRO
                        PROVENIENTE DA TABELA AUXILIAR DE ACORDO COM SUA CHAVE NATURAL.
                        ENTÃO
                            ATUALIZA OS REGISTROS QUE CORRESPONDEM AO TIPO 3
                            DE ACORDO COM A TABELA AUXILIAR.
                    FIM ENTÃO

                FIM ENTÃO
            SENÃO
                INSERE O REGISTRO, DA TABELA AUXILIAR, NA DIMENSÃO

        FIM

FIM PROCEDIMENTO
```

Figura 33: *Pseudocódigo do procedimento para povoamento de dimensão tipo 6.*

3.1.7. Hierarquias

O fluxograma para a carga de uma hierarquia da dimensão pode ser vista na figura 34. Inicialmente é recuperado e formatado o registro oriundo da tabela auxiliar. Após isso, é verificado se existe um nível para a hierarquia. Caso não haja retorna para o início do fluxo. Se houver, verifica se o nível da hierarquia já foi inserido na dimensão. Sendo verdade, irá buscar o próximo nível da hierarquia. Caso contrário, insere o nível da hierarquia na dimensão e em seguida retorna para verificar o próximo nível.

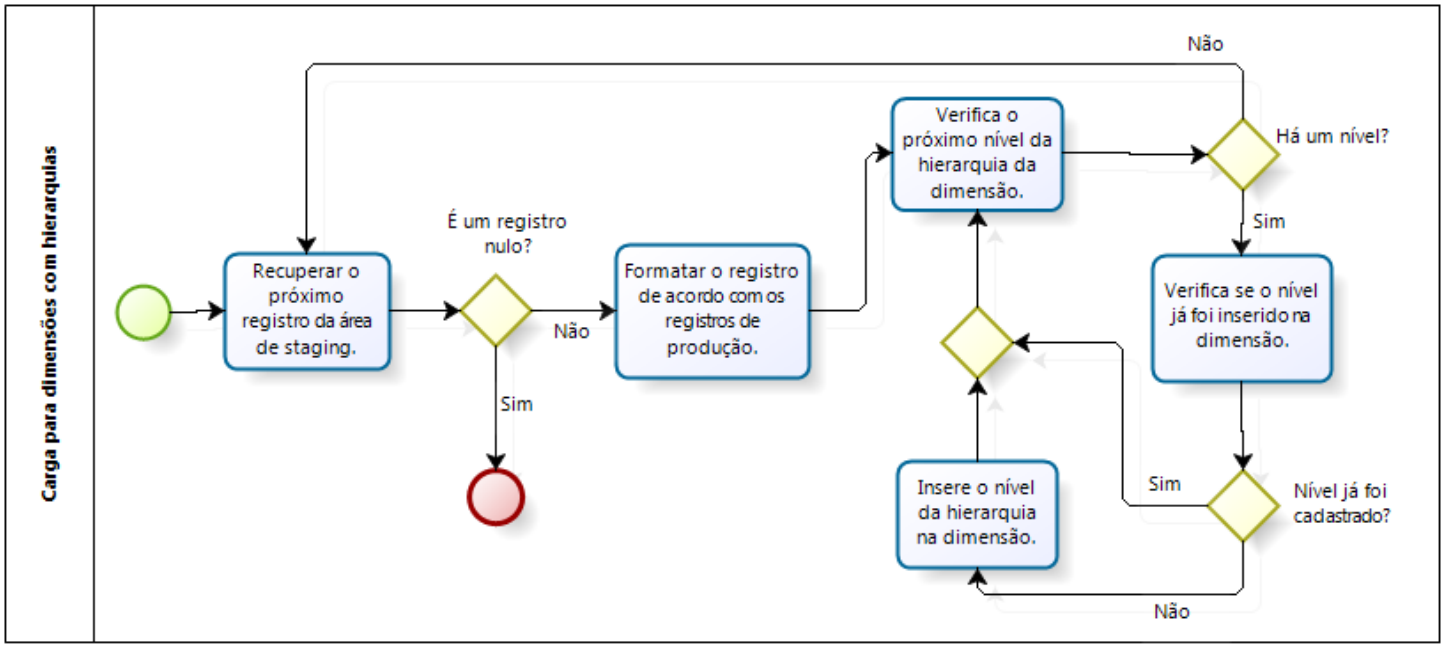


Figura 34: *Fluxograma do algoritmo para carga de uma Hierarquia.*

O código criado para carga de uma hierarquia de dimensão é vista na figura 35. Para o novo registro vindo da tabela auxiliar é verificado se existe nível. Enquanto existir, confere se o registro, que corresponde ao nível da hierarquia, não está contido na dimensão. Se não existir o nível, é inserido o nível da hierarquia na dimensão.

```
CRIAR PROCEDIMENTO Nome_procedimento (@DATACARGA DATETIME)
INÍCIO PROCEDIMENTO
    DECLARAR VARIÁVEIS QUE CORRESPONDEM AOS ATRIBUTOS DA TABELA AUXILIAR

    PARA TODA LINHA DA TABELA AUXILIAR CUJA DATA DE CARGA É IGUAL À DATA PASSADA COMO PARÂMETRO
    FAÇA
        INÍCIO
            PARA CADA NÍVEL DA HIERARQUIA
            FAÇA
                INÍCIO
                    SE O REGISTRO, QUE CORRESPONDE AO NÍVEL DA HIERARQUIA,
                    NÃO ESTIVER INSERIDO NA DIMENSÃO
                    ENTÃO
                        INSERE O REGISTRO DO NÍVEL DA HIERARQUIA NA DIMENSÃO
                FIM
        FIM

FIM PROCEDIMENTO
```

Figura 35: *Pseudocódigo do procedimento para carga de hierarquias.*

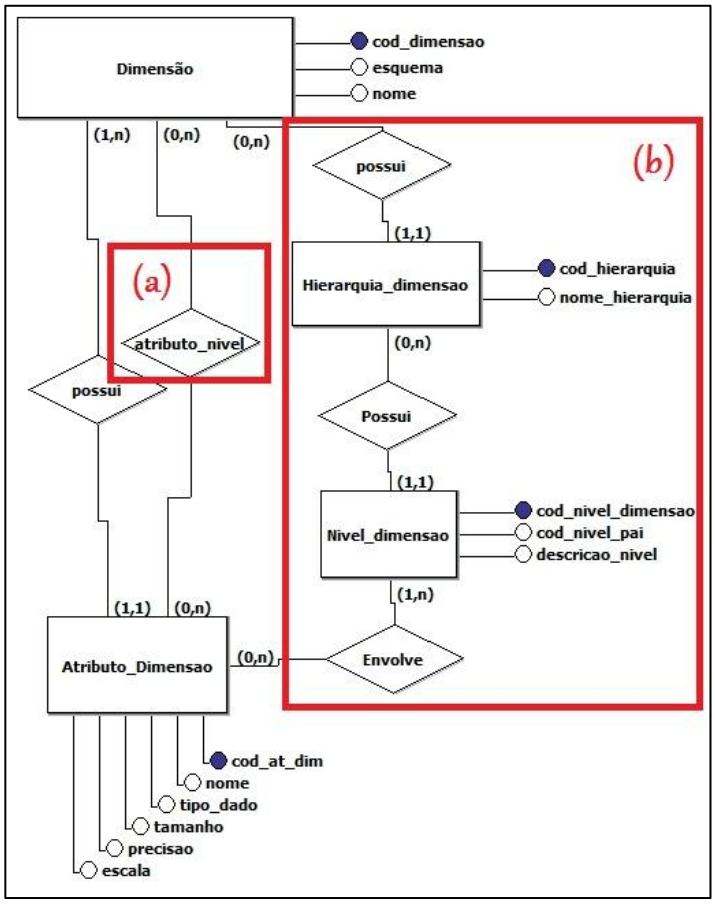


Figura 36: *Esquema Conceitual dos metadados identificados para carga de hierarquias.*

Definido o código de povoamento para as hierarquias, foram identificados os metadados que o compõem, são eles: o atributo da dimensão que corresponde ao nível da hierarquia, visto em (a) da figura 36; a hierarquia da dimensão; níveis da hierarquia, com seus atributos que são: a descrição do nível e o código do nível do pai. As entidades que correspondem às hierarquias e os níveis são encontrados na figura 36 em (b).

3.1.8. Agregado

A figura 37 representa o fluxograma para carga de agregado de uma tabela de fatos. A princípio, há a eliminação dos registros que se encontram no agregado. Nesse ponto, cabe uma observação. Embora a eliminação dos dados que formam um agregado possa ser um passo para o algoritmo, geralmente essa eliminação é realizada com base em alguma restrição. Com base em algum critério são eliminados registros do agregado que correspondem ao último mês, último trimestre, último ano, etc. Uma vez que esse critério é muito particular para cada ambiente, o mesmo não foi contemplado na geração de código. Todavia, essa escolha não impede que um código especificação para eliminação seja incluído posteriormente para completar a lógica de povoamento do agregado. Após a eliminação dos registros, é verificado se as dimensões que correspondem ao agregado possuem hierarquias. A partir disso, o fluxo se divide em dois subprocessos. O primeiro caso, se as dimensões possuírem hierarquias, pode ser visualizado de forma detalhada na figura 38. O segundo caso, se não possuírem hierarquias, é apresentado na figura 39. Ao termino de qualquer um dos fluxos termina o processo de povoamento para os agregados.

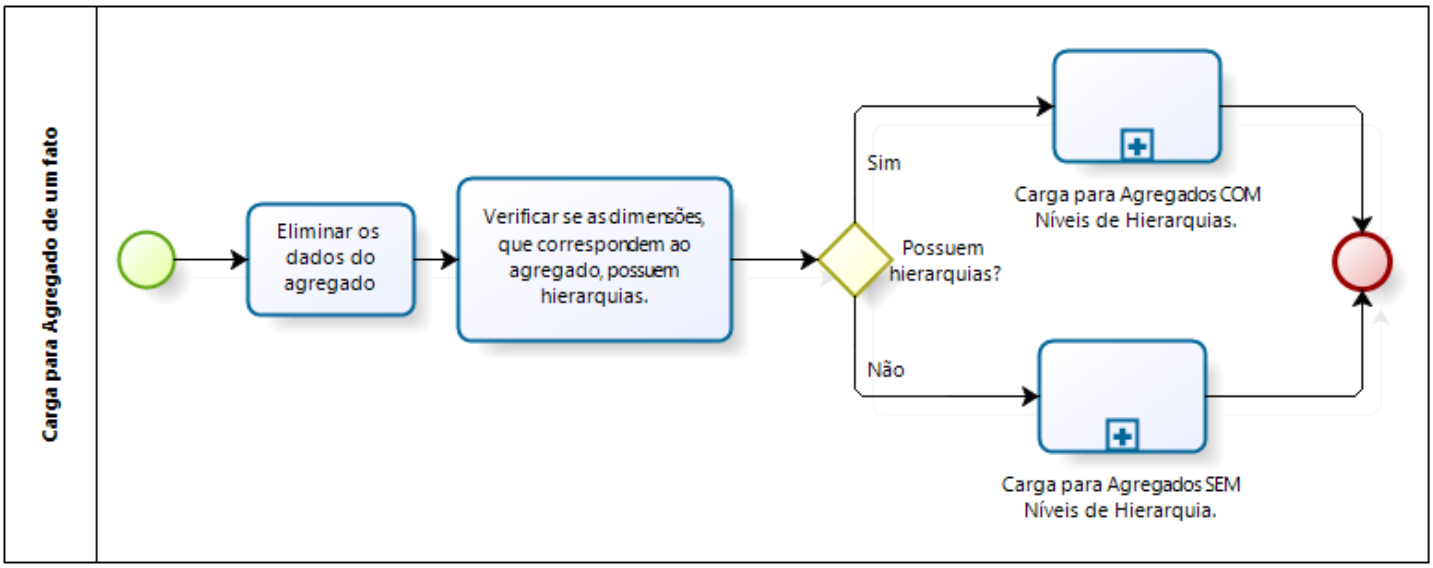


Figura 37: *Fluxograma do algoritmo para povoamento de um Agregado.*

Inicialmente, no fluxograma do algoritmo para efetuar povoamento do agregado com níveis, é selecionado o próximo nível da hierarquia da dimensão. Em seguida são inseridos os registros do nível em uma tabela temporária. Se houver mais níveis em dimensões retorna para o início do fluxo. Caso contrário segue no procedimento a seleção dos valores quantitativos dos fatos e faz as junções das dimensões com as tabelas temporárias. Consequentemente, é inserido o resumo dos registros do fato no agregado e finaliza o procedimento.

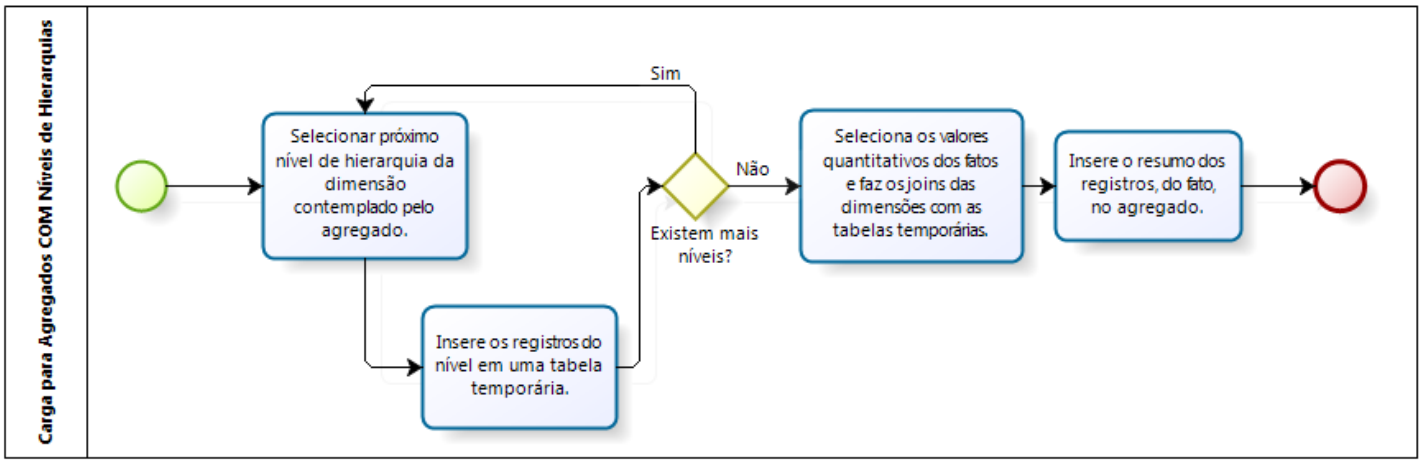


Figura 38: *Fluxograma do algoritmo para efetuar povoamento de Agregado com níveis de hierarquias.*

Para a carga de agregados sem níveis de hierarquias, simplesmente são inseridos os registros do fato no agregado. Isso pode ser visto na figura 39.

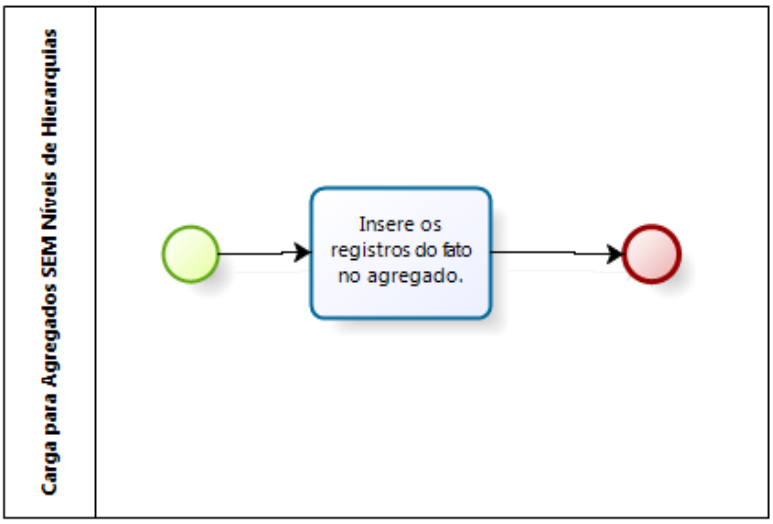


Figura 39: *Fluxograma do algoritmo para efetuar povoamento de Agregado sem níveis de hierarquias.*

A figura 40 contém o pseudocódigo para o povoamento de um agregado sem níveis de hierarquias. Inicialmente os dados do agregado são eliminados (linha 4) e insere no agregado os registros dos atributos que foram selecionados na tabela de fatos.

```
1  CRIAR PROCEDIMENTO Nome_procedimento
2  INÍCIO PROCEDIMENTO
3
4      ELIMINAR TODOS OS DADOS DO AGREGADO
5
6      INSERIR NO AGREGADO OS REGISTROS DOS ATRIBUTOS QUE FORAM SELECIONADOS DA TABELA DE FATOS
7
8  FIM PROCEDIMENTO
```

Figura 40: *Pseudocódigo do procedimento para povoamento de Agregado sem níveis de hierarquias.*

No código para a carga de algoritmos com níveis de hierarquias, as linhas 6 a 14 diferenciam do modelo anterior. Neste, para todo nível de hierarquia de dimensão no agregado, são selecionados os registros da dimensão que corresponde àquele nível e inseridos em uma tabela temporária. Quando não houver mais níveis, serão inseridos no agregado os resumos dos valores quantitativos do fato e dos registros correspondentes aos níveis da hierarquia.

```
CRIAR PROCEDIMENTO Nome_procedimento
INÍCIO PROCEDIMENTO

    ELIMINAR TODOS OS DADOS DO AGREGADO

    PARA TODO NÍVEL DE HIERARQUIA DA DIMENSÃO QUE ESTIVER SENDO CONTEMPLADO NO AGREGADO.
    FAÇA
        INÍCIO
            SELECIONA OS REGISTROS DA DIMENSÃO QUE CORRESPONDEM AO NÍVEL
            E INSERE EM UMA TABELA TEMPORÁRIA.
        FIM

    INSERE NA TABELA DE AGREGADO O RESUMO DOS VALORES QUANTITATIVOS, PROVENIENTES DO FATO,
    E DOS REGISTROS CORRESPONDENTES AOS NÍVEIS DA HIERARQUIA, A PARTIR DA TABELA TEMPORÁRIA.

FIM PROCEDIMENTO
```

Figura 41: *Pseudocódigo do procedimento para povoamento de Agregado com níveis de hierarquias.*

Os metadados identificados para os agregados podem ser visualizados na figura 42, são eles: a tabela de agregado e procedimento com seus nomes e esquemas, em (a). O relacionamento do agregado com a tabela de fatos e seus atributos, em (b). Por fim, visto em (c), a entidade especificação da dimensão no agregado, a qual contém o relacionamento do agregado com a dimensão e seus possíveis níveis da hierarquia.

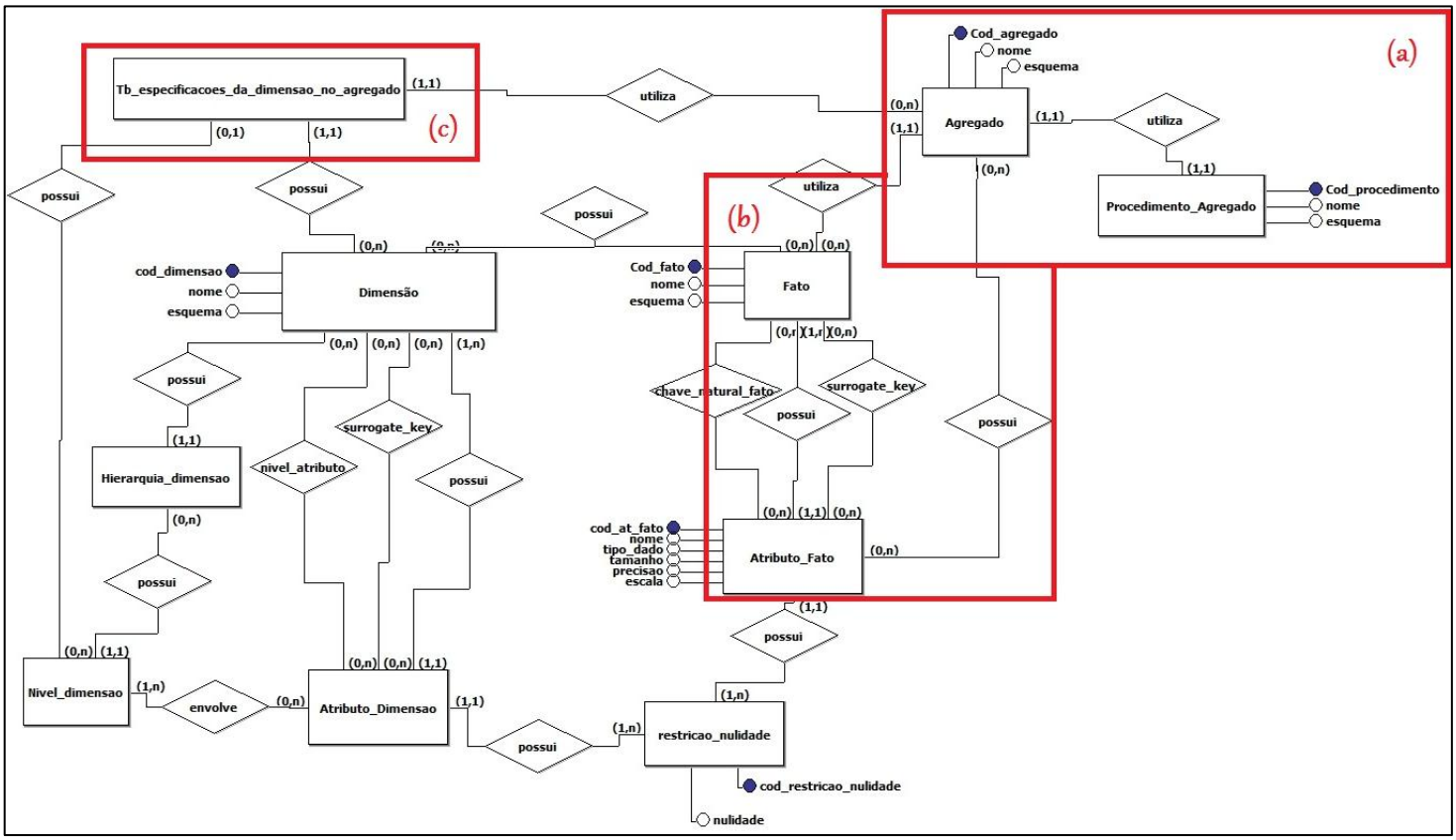


Figura 42: *Esquema Conceitual dos metadados identificados para o povoamento de Agregados.*

3.2. Análise e Projeto da Ferramenta

Após a definição dos metadados, iniciou-se o projeto da ferramenta, no qual foram analisados e definidos quais seriam seus requisitos funcionais e não funcionais. Estes podem ser vistos no quadro 1.

A ferramenta deve conter, como requisitos funcionais, o gerenciamento dos metadados fundamentais para as tabelas auxiliares, dimensões (tipos 4 e 6), agregados e hierarquias. Torna-se imprescindível, para efetivar a geração de código sql, realizar os relacionamentos dos atributos da tabela auxiliar com os da dimensão. Além de gerenciar os procedimentos de povoamento para agregados, hierarquias e dimensões.

Requisitos Funcionais	**R1:** Gerenciar os metadados necessários para a Tabela Auxilar. **R2:** Gerenciar os metadados necessários dos seguintes elementos do ambiente de *Data Warehouse:* 1. Dimensões; 2. Agregados; 3. Hierarquias. **R3:** Gerenciar o relacionamento dos atributos da Tabela Auxiliar com os atributos das tabelas correspondentes no ambiente de *Data Warehouse.* **R4:** Gerenciar procedimentos de povoamento de dados para agregados, hierarquias e dimensões. **R5:** Gerar o código SQL para os procedimentos.
Requisitos Não Funcionais	**R6:** A ferramenta será Desktop. **R7:** O código deverá ser gerado para a linguagem T-SQL. **R8:** A linguagem de desenvolvimento será o JAVA. **R9:** O sistema utilizará o SQL Server como banco de dados.

Quadro 1: *Requisitos funcionais e não funcionais da ferramenta.*

Além desses requisitos citados anteriormente, a ferramenta foi desenvolvida para ser Desktop, na linguagem Java e utilizar o SQL Server como banco de dados. E principalmente, gerar os códigos sql na linguagem T-SQL (*Transact-SQL*).

A concepção da aplicação se deu a partir um processo iterativo e incremental. Cada incremento foi responsável pela geração de código envolvendo um tipo de comportamento para o tratamento de histórico de dimensões do tipo 4 e 6. E posteriormente, para os agregados e hierarquias.

A cada incremento foram realizados testes com casos de uso para validar as funções sendo incorporadas à ferramenta. Os testes garantiram resultados positivos em relação aos objetivos propostos da ferramenta, que são: produtividade e redução em números de erros de codificação durante a fase de povoamento de dados em um ambiente de *Data Warehouse*. Os experimentos, vistos no próximo capítulo, terão um aprofundamento maior para avaliar os resultados da ferramenta.

4. EXPERIMENTOS

Com a finalização do projeto da ferramenta e todos os testes realizados na mesma, passamos para o planejamento e execução dos experimentos. A seguir, nas próximas seções, são apresentados os objetivos e definição dos casos de uso dos experimentos, assim como, seu planejamento e execução.

4.1. Objetivo

O principal objetivo dos experimentos é encontrar evidências que possam acatar ou refutar as seguintes hipóteses: a) que seja possível capturar e modelar a semântica necessária para criação de uma ferramenta de geração automática de código em linguagem SQL para as rotinas de povoamento em um ambiente de *Data Warehouse*; b) que a investigação experimental sobre a utilização da ferramenta possa apresentar indícios de que é possível substituir a codificação manual, total ou em parte, pela geração automática de códigos obtendo um ganho de produtividade, uma redução ou eliminação dos erros de codificação introduzidos nas fases de construção e manutenção, e, consequentemente, uma melhoria na qualidade dos dados.

4.2. Planejamento

A seguir, são enumeradas as etapas de planejamento dos experimentos.

1) *Criação do ambiente de Data Warehouse*: nessa fase foi definido e criado o ambiente de DW com os esquemas dimensionais e áreas de *Staging*. Estes artefatos serviram como base para todo o experimento.

2) *Definição dos Casos de Uso para as rotinas de povoamento:* foram definidas especificações para povoamento de dados a serem seguidas pelos programadores que fizeram parte do experimento. Foram criadas três especificações para o povoamento de hierarquias e

dimensões com características distintas em relação aos tipos de dados dos atributos, comportamento em relação ao histórico e composição de chaves naturais.

3) *Seleção dos programadores para execução dos experimentos:* Foram selecionados estudantes do curso de Bacharelado em Sistemas de Informação da Universidade Federal de Sergipe/Campus Itabaiana que possuíam conhecimentos sobre ambiente de Data Warehouse, tratamento de histórico e comportamento de atributos, e que já haviam desenvolvido rotinas de povoamento, de forma manual, em extensões da linguagem SQL. Os estudantes participaram dos experimentos de forma voluntária. A seleção buscou, apenas, identificar os estudantes que possuíam o conhecimento necessário para a realização dos experimentos. Dentre os estudantes selecionados, quatro participaram dos experimentos.

4) *Alocação de programadores a Casos de Uso:* Em função das grandes diferenças de desempenho que podem existir entre programadores, foi decidido não realizar o experimento utilizando dois grupos (um utilizando a abordagem manual e outro utilizando a ferramenta). Decidiu-se, então, que todos os programadores selecionados utilizariam todos os tratamentos.

5) *Revisão de conceitos básicos sobre rotinas de povoamento para o grupo de programadores*: Foi realizada uma revisão sobre as rotinas de povoamento para ambientes de DW com os programadores selecionados.

6) *Treinamento da ferramenta de geração automática de código:* foi realizado um treinamento com os programadores, a fim de que eles pudessem se familiarizar com a ferramenta de geração automática de código.

7) *Solicitação de execução dos Casos de Uso de forma manual:* os programadores criaram, com base nos casos de uso apresentados, procedimentos para as rotinas de povoamento manualmente. Isto serviu para fazer a comparação com as rotinas geradas pela ferramenta deste projeto.

8) *Solicitação de execução dos Casos de Uso de forma automática:* nesta etapa os programadores utilizaram a ferramenta para codificar os mesmos casos de uso já criados de forma manual.

4.3. Definição do ambiente

Algumas das tabelas que compõem o ambiente de *Data Warehouse*, usados nos experimentos, foram as seguintes: dimensão de produtos e funcionários que representam

dimensões do tipo 4 e 6, respectivamente; e uma hierarquia para a Dimensão de Clientes. Essas são descritas a seguir.

A representação de carga para uma dimensão do tipo 4, composto por duas tabelas, pode ser visto na figura 43. O povoamento de dados acontece da tabela auxiliar (Tb_Aux_Produto) para as dimensões corrente e histórica paralelamente, respectivamente representadas por Dim_Produto_Corrente e Dim_Produto_Historica na figura.

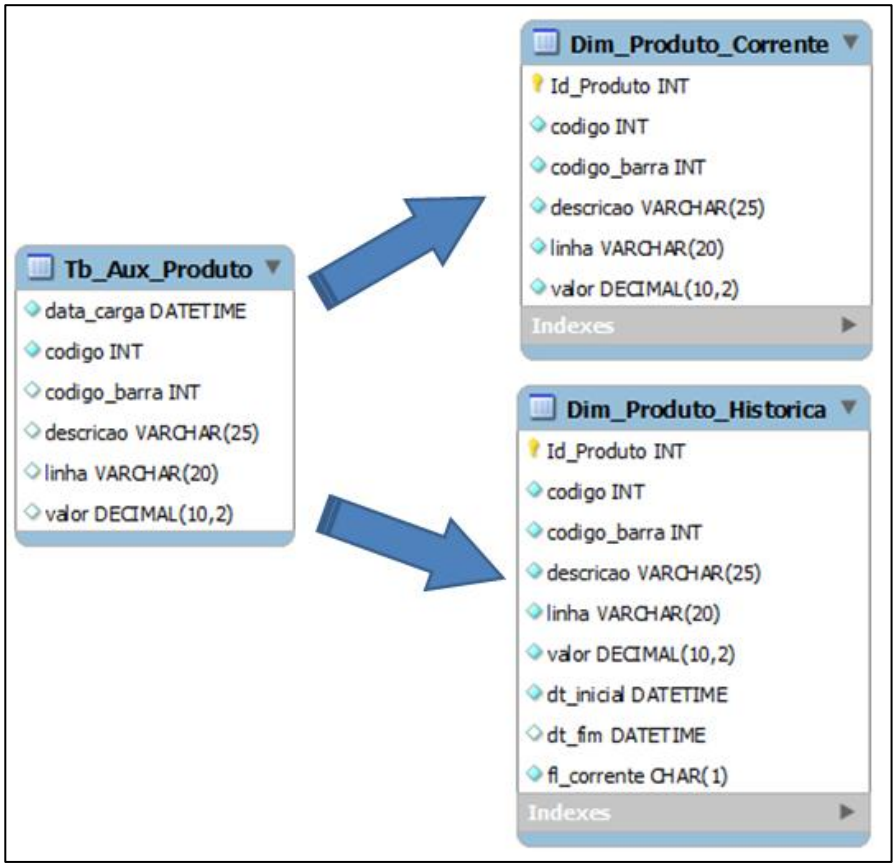


Figura 43: *Carga para uma dimensão do tipo 4, composto pelas tabelas Dim_Produto_Corrente e Dim_Produto_Historica.*

Aa figura 44 contém a representação de uma carga da TB_Aux_Funcionario (tabela auxiliar de funcionários) para a DIM_Funcionario (dimensão de funcionários) que representa uma dimensão do tipo 6. Para esta dimensão, os atributos que correspondem ao tipo 1 são: nome e data_nascimento. Para os atributos do tipo 2: endereço, cidade, bairro, dt_inicio, dt_fim e fl_corrente. Por fim, os atributos do tipo 3: funcao_original, funcao_penultima, função_atual, data_admissao_original e data_admissao_atual.

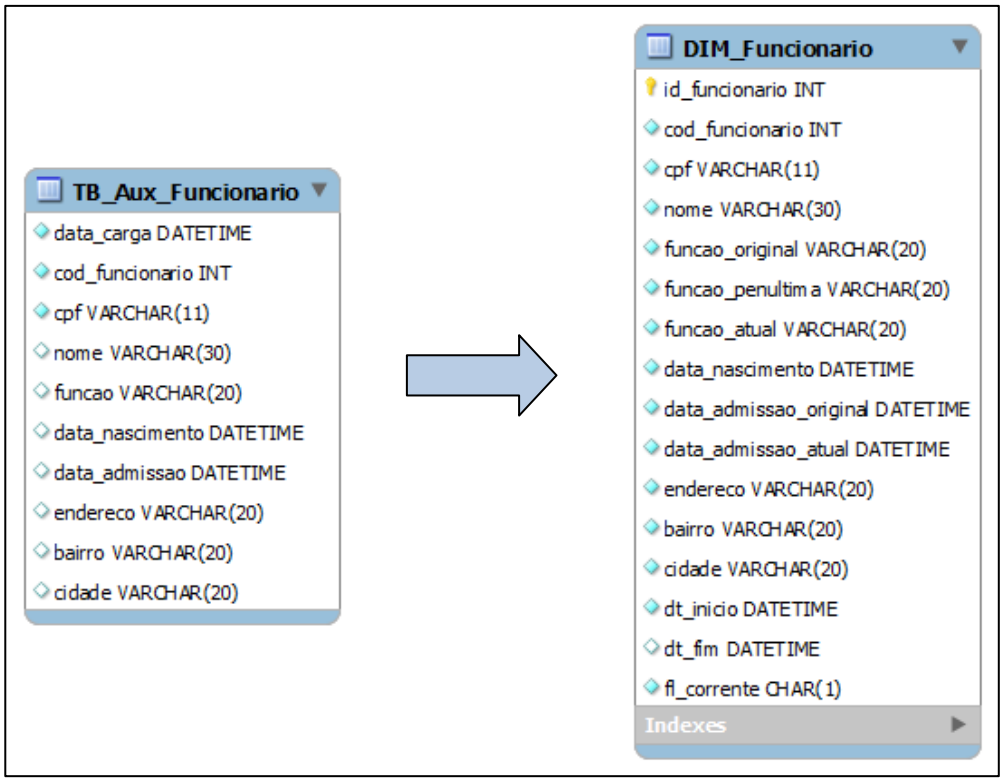


Figura 44: *Carga para uma Dimensão de funcionários com comportamento do tipo 6.*

O exemplo de carga para povoar hierarquias de uma dimensão é vista na figura 45. A dimensão de cliente possui uma hierarquia de endereços, onde o atributo que determina o seu nível é representado por Nivel_Hierarquia. O primeiro nível da hierarquia é o ‘Estado’, representado pelo atributo de mesmo nome. O segundo é a ‘Cidade’, que possui como atributos Cidade e Estado.

Após a definição do ambiente de DW para os experimentos, foram criadas as especificações para o povoamento de dados para os casos citados acima.

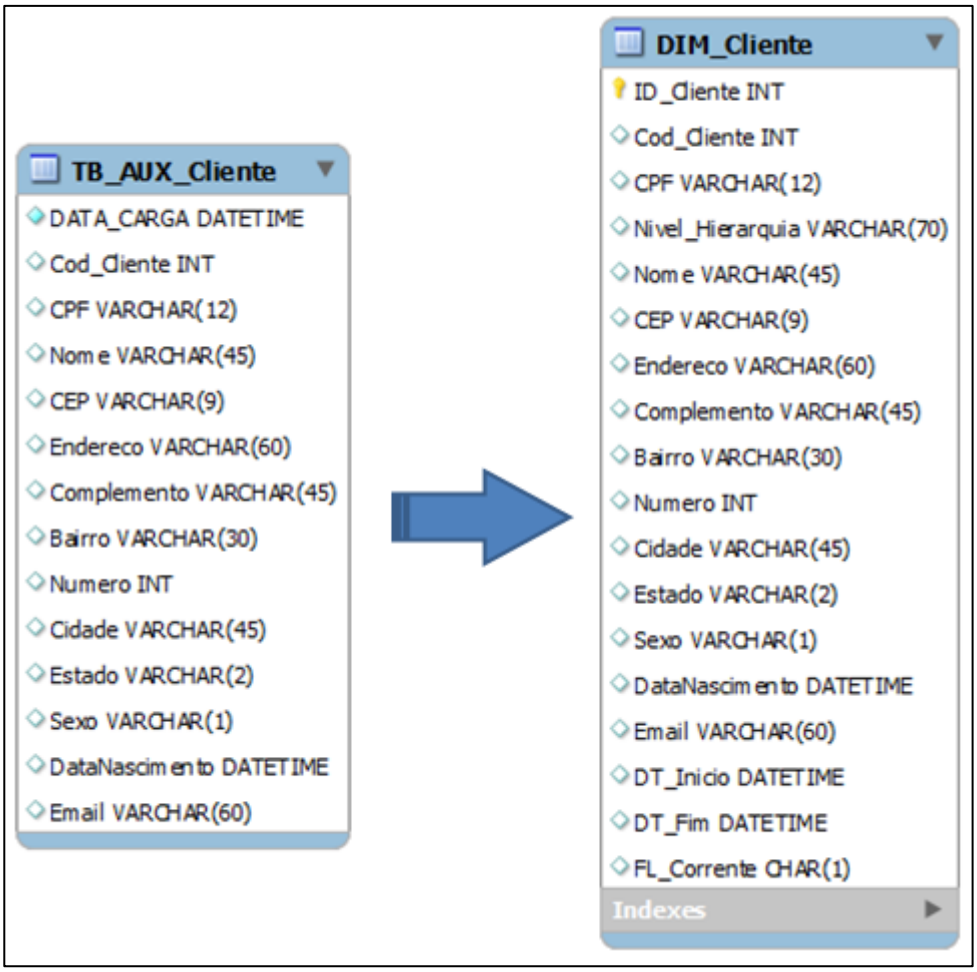


Figura 45: *Carga para uma Dimensão de Clientes com hierarquias.*

4.4. Execução

Durante a execução dos experimentos, foram registrados os tempos de início e fim de execução de cada caso de uso para cada programador, com e sem o uso da ferramenta.

Ao final da execução do experimento, os códigos gerados de forma manual e automática foram testados a fim verificar sua validade. Também foram identificados os erros de codificação inseridos durante o processo de criação dos procedimentos.

Ao final da execução do experimento, foram analisadas as métricas que servirão de base para avaliar as hipóteses: *a) produtividade* – verificar o tempo gasto para a criação do procedimento e tempo para a manutenção do mesmo; *b) número de erros de codificação* – número de erros de codificação encontrados nas fases de criação e manutenção dos procedimentos.

5. RESULTADOS

Após a execução dos experimentos e coleta dos dados, foram realizadas análises com o objetivo de acatar ou refutar as hipóteses do trabalho.

5.1. Substituição da Codificação Manual pela Codificação Automática

Todos os programadores conseguiram gerar automaticamente, através da ferramenta, as rotinas de povoamento para todos os casos de uso selecionados. As rotinas foram analisadas e testadas. Todas apresentam o mesmo código e são válidas em relação ao povoamento de dados. Uma vez que as especificações apresentavam todas as combinações possíveis para o tratamento de histórico em dimensões, assim como hierarquias, observa-se que os metadados identificados mostram-se suficientes para capturar a semântica necessária para a geração automática de código para rotinas de povoamento. A utilização correta da ferramenta por todos os programadores mostrou que os conceitos apresentados pelos metadados da ferramenta são de fácil entendimento e conseguiram, pelo menos para os casos de uso selecionados, expressar as diferentes necessidades de codificação. Desse modo, pode-se afirmar que é válida a hipótese de que a codificação manual pode ser substituída pela codificação automática. Outro benefício observado na codificação automática é a imposição de padrões de codificação. Os códigos gerados manualmente apresentaram diferentes padrões, mesmo para as rotinas mais simples. Essa padronização é altamente relevante para documentação, legibilidade de código e manutenção.

5.2. Tempo de Desenvolvimento

A variável *Tempo de Desenvolvimento* foi extraída da seguinte hipótese: que a geração automática de código pode aumentar a produtividade dos programadores. Quanto menor o tempo de desenvolvimento, maior será a produtividade. Os experimentos buscam identificar se existe uma relação entre o uso da ferramenta e o tempo necessário para a codificação das

rotinas de povoamento. Durante a execução dos experimentos foram coletados os tempos de codificação de todos os programadores para todos os casos de uso durante a codificação manual e durante a utilização da ferramenta. A média das diferenças entre o tempo necessário para construção das rotinas de forma manual, e o tempo necessário com a utilização da ferramenta foi de 23,3 minutos. Os dados coletados podem ser observados na figura 46.

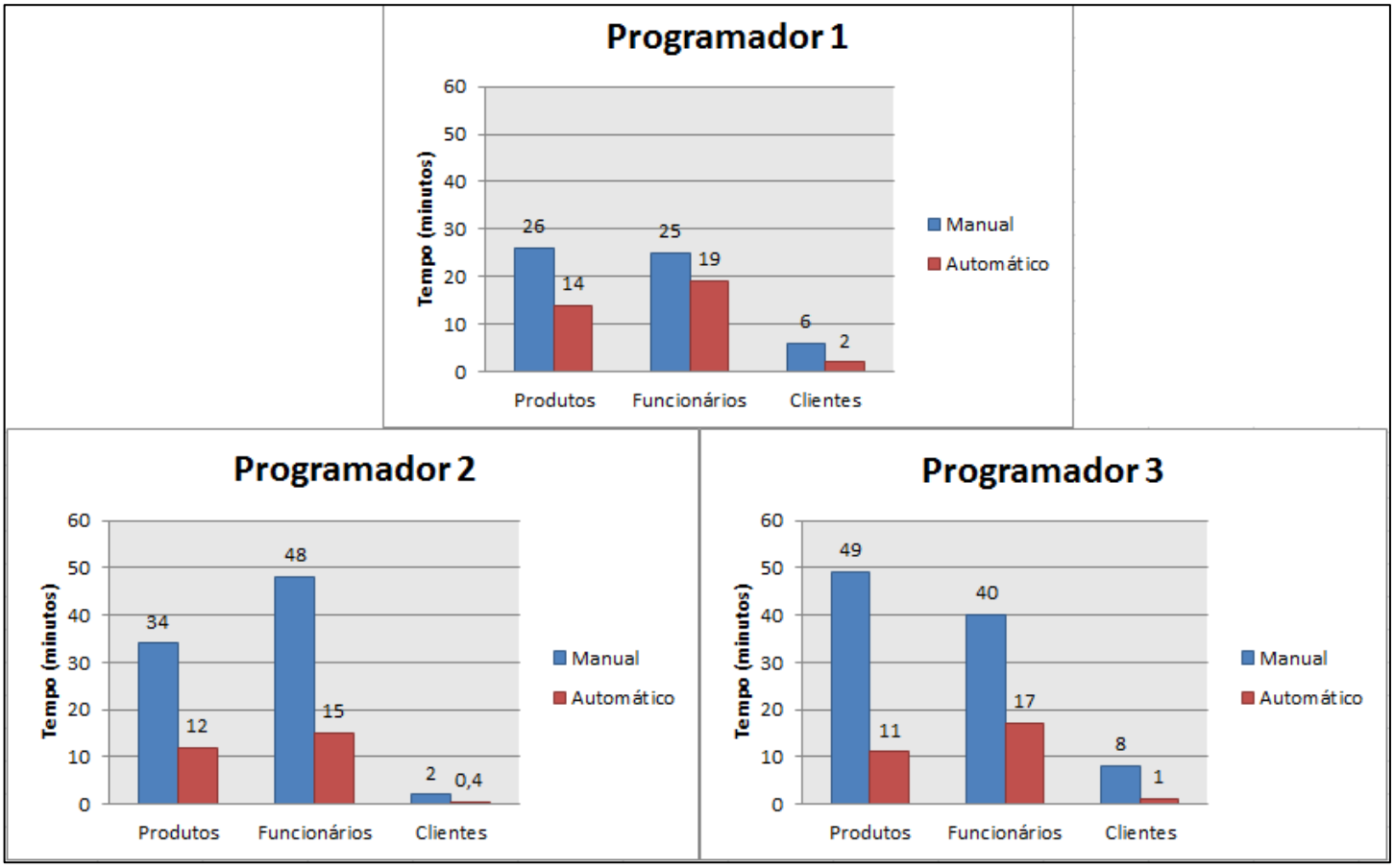


Figura 46: *Tempos medidos (em minutos) para a codificação dos Casos de Uso Produtos, Funcionários e Clientes.*

A figura 46 apresenta os tempos de desenvolvimento medidos e coletados para os quatro programadores envolvidos nos experimentos. Para cada programador são apresentados os tempos da codificação manual e da codificação automática (utilização da ferramenta) para os casos de uso: Produtos, Funcionários e Clientes.

A análise dos tempos apresenta fortes indícios que existe uma relação entre o tipo de tratamento (manual ou automático) e o tempo necessário para o desenvolvimento das rotinas de povoamento. A utilização da ferramenta de geração automática de código, para todos os casos e programadores, obteve sempre menores tempos de desenvolvimento. Essa relação se

mantém mesmo quando existem diferenças de produtividade entre os programadores. Logo, podemos afirmar que existem fortes evidências que a utilização da ferramenta de geração automática de código aumenta a produtividade dos programadores durante a criação das rotinas de povoamento.

5.3. Número de Erros de Codificação

A variável *Número de Erros de Codificação* foi extraída da hipótese: que a geração automática de código pode reduzir ou eliminar erros de codificação introduzidos durante as fases de construção e manutenção das rotinas de povoamento. No contexto desse trabalho, erros de codificação representam instruções que causam ou podem causar erros na lógica de povoamento e, consequentemente, inserir ou atualizar registros de forma incorreta nas bases de dados. Os experimentos buscam identificar se existe uma relação entre o uso da ferramenta e os erros de codificação encontrados nos procedimentos.

Após o término dos experimentos, os códigos gerados manualmente e de forma automática foram analisados e testados para verificar a validade, os possíveis erros de codificação e se atendiam aos requisitos dos Casos de Uso.

Todos os códigos gerados pela ferramenta foram considerados válidos, atendiam aos requisitos dos Casos de Uso e não apresentaram erros de codificação.

Os códigos gerados manualmente apresentaram alguns problemas em relação à validade e erros de codificação:

1) Dois procedimentos manuais apresentaram problemas. O primeiro no tratamento de histórico para atributos com comportamento tipo 2. E o segundo para carregamento de hierarquias.
2) Todos os procedimentos manuais apresentaram problemas no tratamento de atributos cujos valores são cadeias de caracteres. As comparações de igualdade efetuadas não levaram em conta as diferenças entre maiúsculas e minúsculas. Essa falta de tratamento pode incluir registros desnecessários em uma tabela de dimensão.

3) Três procedimentos apresentaram problemas na codificação de restrições envolvendo a chave natural da dimensão. A identificação incorreta da chave natural pode gerar inclusões desnecessárias e atualizações incorretas.

Em função dos dados observados, pode-se afirmar, também, que existem fortes indícios de que a utilização de uma ferramenta de geração automática de código pode reduzir ou eliminar erros de codificação durante a fase de construção de rotinas de povoamento. Como não foram realizados experimentos com Casos de Uso que especificam manutenções, os dados coletados não podem expressar alguma relação entre o número de erros de codificação e a manutenção de rotinas.

6. CONCLUSÕES

Identificar e tentar solucionar os problemas de qualidade de dados em um ambiente de *Data Warehouse* representa um dos principais obstáculos enfrentados pelas grandes empresas no processo de utilização de Sistemas de Apoio à Decisão. Dentre os vários fatores que contribuem para a má qualidade dos dados, está a codificação manual de rotinas de povoamento de dados. Nesse trabalho, levantamos a hipótese de que a geração automática de código pode substituir a codificação manual e contribuir para melhoria da qualidade através do impacto em variáveis como produtividade e erros de codificação.

Para verificar essas hipóteses, foi proposta a criação de uma ferramenta para geração automática de código para rotinas de povoamento em um ambiente de *Data Warehouse*.

Dentre os principais desafios para a geração automática de código, está a coleta e definição de metadados necessários para o processo de povoamento. O conjunto de metadados precisa conter informações que possam contemplar as várias características das rotinas de povoamento, assim como as variações que possam ocorrer em diferentes ambientes e aplicações. Através de revisão da literatura, utilização de rotinas já empregadas em grandes empresas e muitas discussões, chegamos a um conjunto satisfatório de metadados que se mostraram suficientes para a realização dos experimentos. Esse conjunto de metadados também se mostrou satisfatório nos testes realizados com a ferramenta desenvolvida.

Outros trabalhos moderadamente relacionados também buscam soluções para a geração automática de código dos processos ETL. Em (MUNOS; MAZÓN; TRUJILLO, 2009) é apresentada uma abordagem dirigida a modelos para a geração automática de processos ETL. Essa abordagem difere da nossa uma vez que o objetivo é gerar processos baseados em modelos de arquitetura de ferramentas ETL já existentes. Nesse sentido, o tratamento dado a dimensões é limitado às ferramentas que se integram ao framework. Em (THOMSEN; PEDERSEN, 2009) é apresentado um framework para programação de rotinas ETL em substituição a ferramentas gráficas. Embora o framework tenha alcançado bons resultados nos primeiros experimentos, a abordagem para o tratamento de dimensões é

limitado aos Tipos 1 e 2. A falta de definição de metadados também difere da abordagem desse trabalho.

Os primeiros experimentos realizados com a ferramenta confirmaram a hipótese que a codificação manual de rotinas de povoamento pode ser substituída pela geração automática de código. Os experimentos também apresentaram fortes indícios de que as variáveis Tempo de Desenvolvimento e Número de Erros de Codificação possuem relação com o tipo de tratamento empregado. Desse modo, existem fortes evidências que a ferramenta de geração de código para rotinas de povoamento pode contribuir para o aumento da produtividade e redução ou eliminação dos erros de codificação durante a fase de construção de rotinas de povoamento para ambientes de suporte à decisão. Ainda não foram realizados experimentos com o objetivo de encontrar evidências que a utilização da ferramenta também pode melhorar a produtividade e reduzir erros de codificação durante a fase de manutenção das rotinas de povoamento.

O presente trabalho pode servir de base para trabalhos futuros. A avaliação da ferramenta durante a fase de manutenção das rotinas de povoamento é uma extensão importante que pode confirmar os benefícios da utilização de uma ferramenta de geração automática de código para ambientes de *Data Warehouse.* Outra extensão possível seria a criação de novos módulos para contemplar outros subsistemas como: Tabelas de Fatos, Procedimentos de Recuperação e Controle de Cargas.

REFERÊNCIAS

ADAMSON, C. **Mastering Data Warehouse Aggregates: Solutions for Star Schema Performance.** Indianapolis: Wiley Publish, Inc., 2006.

BECKER, B.**Kimball University: The Subsystems of ETL Revisited.**2007.Disponível em:<http://www.kimballgroup.com/2007/10/21/subsystems-of-etl-revisited/> Acesso em Outubro, 2012.

COLAÇO JÚNIOR, M. **Projetando sistemas de apoio à decisão baseados em data warehouse**. Rio de Janeiro: Axcel Books, 2004.

IMHOFF, C.; GALEMMO, N.; GEIGER. J. G. **Mastering Data Warehouse Design: Relational and Dimensional Techniques**. Indianapolis: Wiley Publish, Inc., 2003.

INMON, W.H. **Como construir o DataWarehouse**. Rio de Janeiro: Editora Campus, 1997.

INMON, W. H. **Building the data warehouse, Fourth Edition.** 4. ed. Indianapolis, Indiana: Wiley Publishing Inc., 2005.

KIMBALL, R; ROSS, M.**The data warehouse toolkit: The complete Guide to Dimensional Modeling.** 2. ed. John Wiley and Sons, Inc., 2002.

KIMBALL, R. **The Data Warehouse ETL Toolkit.**1 ed. Wiley India (P) Ltd., 2004.

KIMBALL, R; ROSS, M.; THORNTHWAITE, W.**The data warehouse lifecycle toolkit**. 2. ed. Indianapolis, Indiana: Wiley Publishing Inc., 2008.

MUNOZ L.; MAZÓN J.; TRUJILLO J. **Automatic generation of ETL processes from conceptual models**.Proceedings of the ACM international workshop on Data warehousing and OLAP. Hong Kong, China, 2009.

OBJECT MANAGEMENT GROUP.**Model Driven Architecture (MDA)**, version 1.0.1. 2003. Disponível em:<http://www.omg.org/mda/specs.htm>Acesso em Outubro, 2012.

ROSS, M.**Kimball University: Slowly Changing Dimensions Are Not Always as Easy as 1, 2, 3. 2005.** Disponível em:<http://www.kimballgroup.com/2005/03/10/slowly-changing-dimensions-are-not-always-as-easy-as-1-2-3/>Acesso em Outubro, 2012.

RUDRA, A.; YEO, E. **Key Issues in Achieving Data Quality and Consistency in Data Warehousing among Large Organizations in Australia**. In: Proceedings of the 32nd Hawaii International Conference on System Sciences. Hawaii: IEEE, 1999. v. 7.

SANTOS, V; BELO, O.**No Need to Type Slowly Changing Dimensions.**IADIS International Conference Information Systems, 2011a.

SANTOS, V; BELO, O. **Slowly Changing Dimensions Specification a Relational Algebra Approach**.Proc. of Int. Conf. on Advances in Communication and Information Technology, 2011b.

SINGH, R.; SINGH, K.**A Descriptive Classification of Causes of Data Quality Problems in Data Warehouse.**IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 2, May 2010.

THOMSEN C.; PEDERSEN T. B. **Pygrametl: a powerful programming framework for extract-transform-load programmers**. Proceedings of the ACM international workshop on Data warehousing and OLAP. Hong Kong, China, 2009.

APÊNDICE I

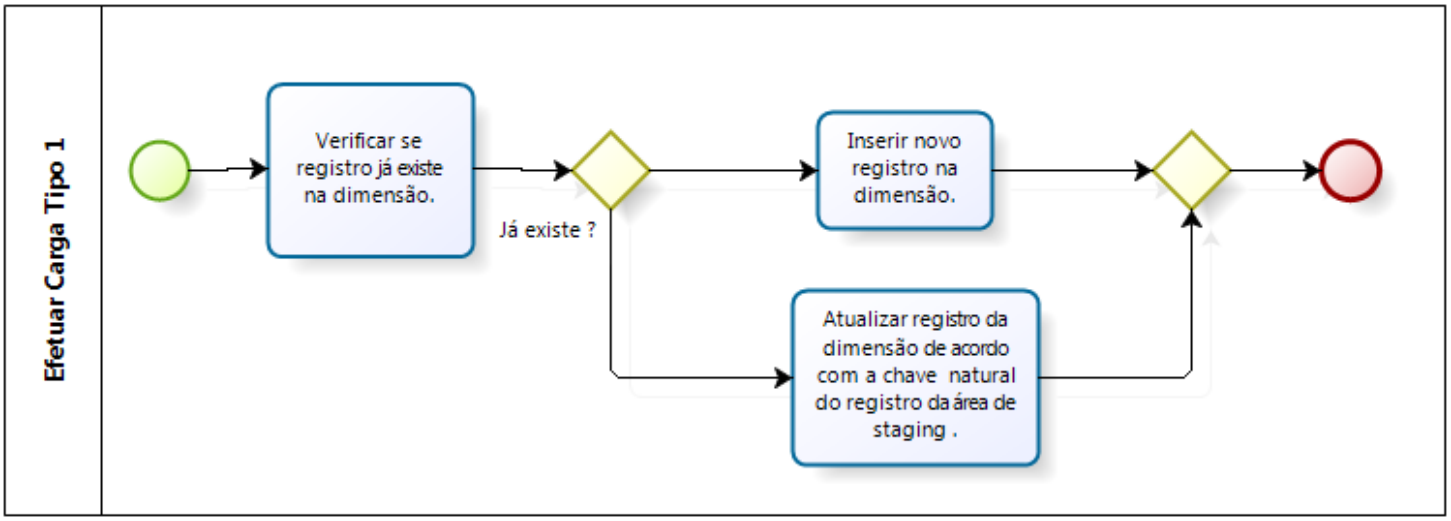


Figura 1: *Fluxograma do algoritmo para efetuar povoamento de atributo do tipo 1.*

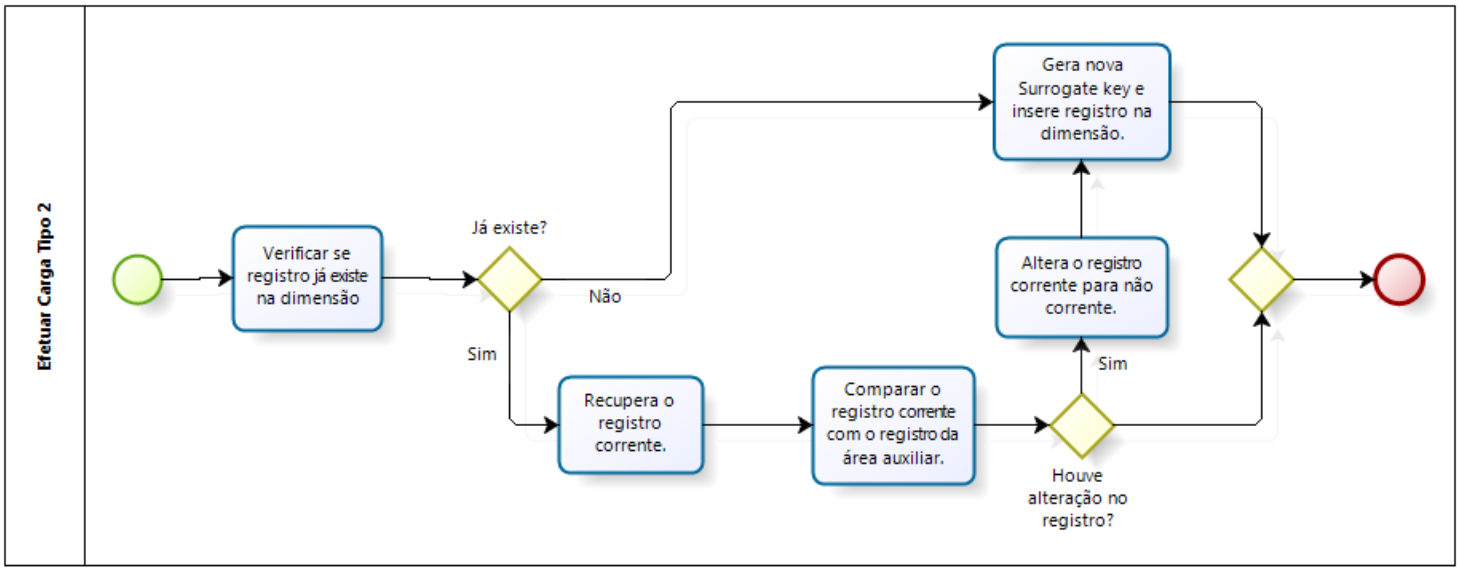


Figura 2: *Fluxograma do algoritmo para efetuar povoamento de atributo do tipo 2.*

APÊNDICE II

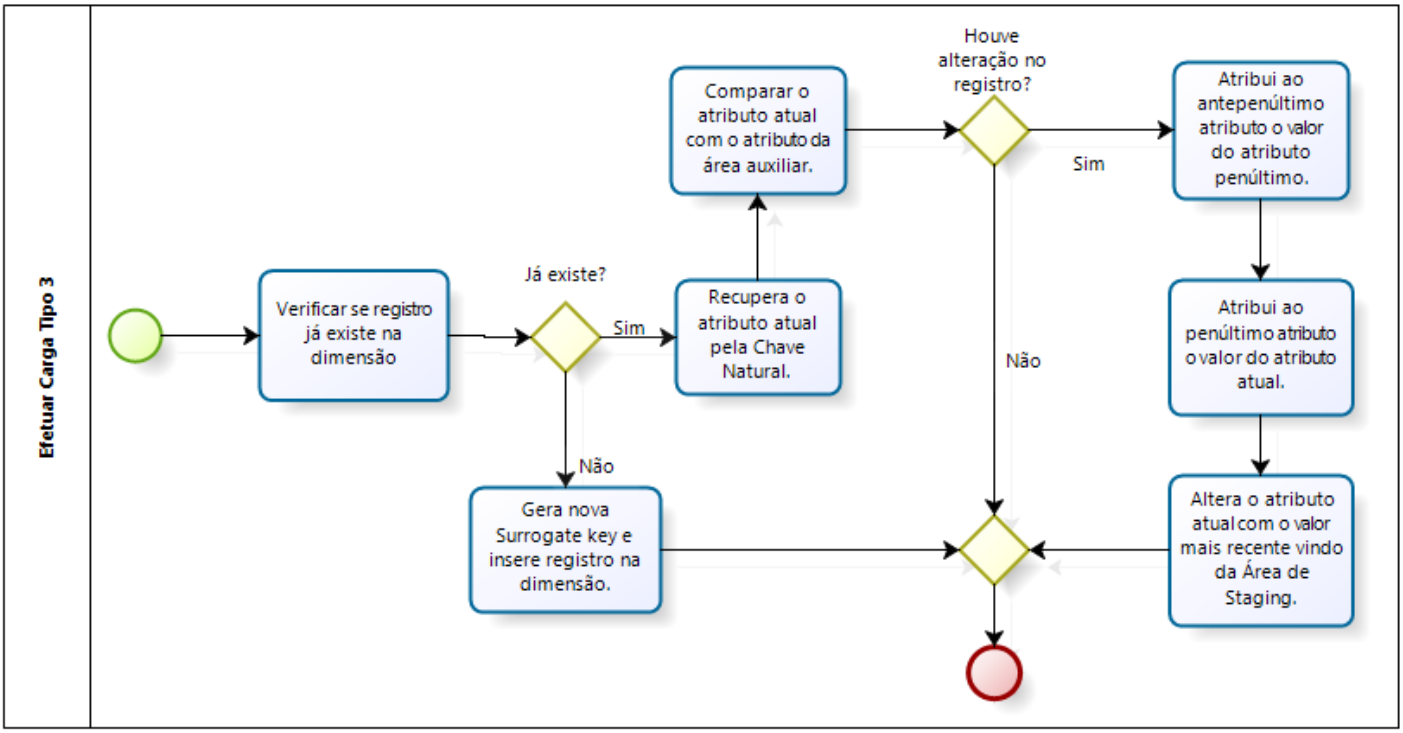


Figura 1: *Fluxograma do algoritmo para efetuar povoamento de atributo do tipo 3.*

ANEXO I

XII Escola Regional de Computação Bahia Alagoas Sergipe
Complexo Multieventos da UNIVASF - Campus Juazeiro/BA
24 a 27/abr/2012

CERTIFICADO

Certificamos que o trabalho **Desenvolvimento e Avaliação de uma Ferramenta de Geração Automática de Código para Ambientes de Apoio à Decisão** foi publicado por **Igor Santos, Juli Costa, André Nascimento e Methanias Colaço Jr.** no XII Workshop de Trabalhos de Iniciação Científica e de Graduação Bahia-Alagoas-Sergipe (XII WTICG) da XII Escola Regional de Computação Bahia Alagoas Sergipe - ERBASE 2012, conforme os Anais Digitais do evento com o ISSN: 2177-4692.

Juazeiro-BA, 27 de abril de 2012.

Prof. MSc. Jorge Luis Cavalcanti Ramos
Coordenador Geral da ERBASE 2012

Responsabilidade Ambiental. Este certificado foi gerado digitalmente, evitando impressões adicionais. Para informações e verificação, erbase2012@univasf.edu.br.

Realização

Promoção

Apoio

ANEXO II

22º ENCONTRO DE INICIAÇÃO CIENTÍFICA

Certificado

Igor Peterson Oliveira Santos

apresentou o trabalho

DESENVOLVIMENTO E AVALIAÇÃO DE UMA FERRAMENTA DE GERAÇÃO AUTOMÁTICA DE CÓDIGO PARA AMBIENTES DE APOIO À DECISÃO.

de autoria de

Igor Peterson Oliveira Santos; Juli Kelle Gois Costa; Methanias Colaço Rodrigues Júnior; Andre Vinicius Rodrigues Passos Nascimento

Cidade Universitária "Prof. José Aloísio de Campos", 01 de novembro 2012.

Anne Michelle Garrido Pedrosa de Souza
Coordenadora de Pesquisa

Cláudio Andrade Macêdo
Pró-Reitor de Pós-Graduação e Pesquisa

Realização: UFS / PROPAP/CNPEB

Apoio: FAPITEC/SE, CNPq