MÁRCIA BEATRIZ PEREIRA DOMINGUES

UM NOVO PROCESSO PARA REFATORAÇÃO DE BANCOS DE DADOS

São Paulo
2014

MÁRCIA BEATRIZ PEREIRA DOMINGUES

UM NOVO PROCESSO PARA REFATORAÇÃO DE BANCOS DE DADOS

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção
do Título de Doutor em Ciências.

São Paulo
2014

MÁRCIA BEATRIZ PEREIRA DOMINGUES

UM NOVO PROCESSO PARA REFATORAÇÃO DE BANCOS DE DADOS

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção
do Título de Doutor em Ciências.

Área de Concentração:
Engenharia da Computação

Orientador:
Prof. Dr. Jorge Rady de Almeida Junior

São Paulo
2014

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 06 de Junho de 2014.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Domingues, Márcia Beatriz Pereira

**Um novo processo para refatoração de banco de dados /
M.B.P. Domingues. -- versão corr. -- São Paulo, 2014.
112 p.**

**Tese (Doutorado) - Escola Politécnica da Universidade de
São Paulo. Departamento de Engenharia de Computação e Sis-
temas Digitais.**

**1.Banco de dados (Evolução; Refatoração) 2.Esquemas
3.Performance de consultas 4.Workflow I.Universidade de São
Paulo. Escola Politécnica. Departamento de Engenharia de
Computação e Sistemas Digitais II.t.**

Dedico este trabalho à minha família, especialmente para meu marido Helves e meu
sobrinho Gabriel

AGRADECIMENTOS

Agradeço primeiramente a Deus, que nos momentos difíceis me deu forças para superar os obstáculos e continuar a caminhar.

Ao meu orientador Jorge Rady de Almeida Júnior que soube com paciência e sabedoria me conduzir durante esse trabalho. Agradeço de coração tudo o que me ensinou, as oportunidades que me deu e a atenção que teve comigo.

Aos meus pais, Márcia e Davi que tanto amo, que muito se empenharam para me proporcionar mais essa conquista profissional.

Aos meus queridos irmãos, Fábio e Naíra, que são tão especiais na minha vida.

Ao meu marido Helves, pelo apoio, compreensão e motivação para realização deste trabalho.

Ao professor Antônio Mauro Saraiva, pela confiança e oportunidade de contribuir com os trabalhos do LAA.

Aos colegas do Laboratório de Automação Agrícola, Wilian e Allan, pela parceria em todos os trabalhos que realizamos juntos.

Aos colegas do Grupo de Análise de Segurança, Andréia, Lúcio, Ricardo, Antônio e Daniel, pela disposição em me ajudar quando precisei.

Ao meu amigo e professor de inglês Claudio Maia, pela ajuda e apoio nos textos e trabalhos que apresentei.

Aos meus amigos Laura Lourenço, Rafael Will, pelo apoio e torcida de sempre.

Agradeço a todas as pessoas que direta ou indiretamente contribuíram de alguma forma para a conclusão deste trabalho.

*A gente pode morar numa casa mais ou menos, numa rua mais ou menos,
numa cidade mais ou menos, e até ter um governo mais ou menos.
A gente pode dormir numa cama mais ou menos, comer um feijão mais ou
menos, ter um transporte mais ou menos, e até ser obrigado a acreditar
mais ou menos no futuro.
A gente pode olhar em volta e sentir que tudo está mais ou menos...*

TUDO BEM!

*O que a gente não pode mesmo, nunca, de jeito nenhum...
é amar mais ou menos, sonhar mais ou menos, ser amigo mais ou menos,
namorar mais ou menos, ter fé mais ou menos, e acreditar mais ou menos.*

Senão a gente corre o risco de se tornar uma pessoa mais ou menos.

(Chico Xavier)

RESUMO

O projeto e manutenção de bancos de dados é um importante desafio, tendo em vista as frequentes mudanças de requisitos solicitados pelos usuários. Para acompanhar essas mudanças o esquema do banco de dados deve passar por alterações estruturais que muitas vezes prejudicam o desempenho e o projeto das consultas, tais como: relacionamentos desnecessários, chaves primárias ou estrangeiras criadas fortemente acopladas ao domínio, atributos obsoletos e tipos de atributos inadequados. A literatura sobre Métodos Ágeis para desenvolvimento de software propõe o uso de refatorações para evolução do esquema do banco de dados quando há mudanças de requisitos. Uma refatoração é uma alteração simples que melhora o *design*, mas não altera a semântica do modelo de dados, nem adiciona novas funcionalidades. Esta Tese apresenta um novo processo para aplicar refatorações ao esquema do banco de dados. Este processo é definido por um conjunto de tarefas com o objetivo de executar as refatorações de uma forma controlada e segura, permitindo saber o impacto no desempenho do banco de dados para cada refatoração executada. A notação BPMN foi utilizada para representar e executar as tarefas do processo. Como estudo de caso foi utilizado um banco de dados relacional, o qual é usado por um sistema de informação para agricultura de precisão. Esse sistema, baseado na Web, necessita fazer grandes consultas para plotagem de gráficos com informações georreferenciadas.

Palavras Chave: Evolução de banco de dados. Refatoração de banco de dados. Esquemas. Performance de consultas. Processo.

ABSTRACT

The development and maintenance of a database is an important challenge, due to frequent changes and requirements from users. To follow these changes, the database schema suffers structural modifications that, many times, negatively affect its performance and the result of the queries, such as: unnecessary relationships, primary and foreign keys, created strongly attached to the domain, with obsolete attributes or inadequate types of attributes. The literature about Agile Methods for software development suggests the use of refactoring for the evolution of database schemas when there are requirement changes. A refactoring is a simple change that improves the design, but it does not alter the semantics of the data model neither adds new functionalities. This thesis aims at proposing a new process to apply many refactoring to the database schema. This process is defined by a set of refactoring tasks, which is executed in a controlled, secure and automatized form, aiming at improving the design of the schema and allowing the DBA to know exactly the impact on the performance of the database for each refactoring performed. A notation BPMN has been used to represent and execute the tasks of the workflow. As a case study, a relational database, which is used by an information system for precision agriculture was used. This system is web based, and needs to perform large consultations to transfer graphics with geo-referential information.

Keywords: Evolutionary Databases. Database Refactoring. Database Schema. Query Performance. Workflow.

LISTA DE FIGURAS

Figura 1 – Esquema para o desenvolvimento de um método	18
Figura 2 – Método de pesquisa aplicado à Tese	20
Figura 3 - Ciclo de vida de uma refatoração.....	31
Figura 4 - Replicação de dados Assíncrona: etapa de coleta de dados	34
Figura 5 - Replicação de dados Assíncrona: Etapa de mapeamento	34
Figura 6 - Replicação de dados Assíncrona: Etapa de execução.....	35
Figura 7 – Diferença conceitual entre processo, atividades e tarefas	37
Figura 8 – Ciclo PDCA (<i>Plan, Do, Check, Act</i>)	40
Figura 9 – Elementos Básicos da notação BPMN	42
Figura 10 – Tipos de Atividade da notação BPMN	43
Figura 11 – Tipos de Tarefas da notação BPMN.....	43
Figura 12 – Tipos de Eventos da notação BPMN.....	43
Figura 13 – Tipos de fluxo de sequência da notação BPMN	45
Figura 14 – Tipos de <i>Gateways</i> da notação BPMN	46
Figura 15 – Decisão Exclusiva (XOR) – Baseada em Dados	47
Figura 16 – Decisão Exclusiva (XOR) – Baseada em Eventos.....	47
Figura 17 – Decisão Inclusiva (OR).....	48
Figura 18 – Execução de Fluxos em paralelo	48
Figura 19 – Exemplo de Piscina e Raia.....	49
Figura 20 – Conectores para fluxo de mensagem e associação de elementos	50
Figura 21 – Processo com uso de Piscinas, Raias e Fluxo de Mensagem.....	51
Figura 22 – Uso de Artefatos	52
Figura 23 – Contexto do processo da refatoração.....	54
Figura 24– Um novo processo para refatoração de banco de dados	55

Figura 25 – Diagrama original para o processo de uma refatoração proposto por Ambler	61
Figura 26 – Processo de uma refatoração proposto por Ambler escrito em BPMN ...	62
Figura 27– Exemplo de Fazenda e Talhão.....	69
Figura 28– Modelo de dados Original	70
Figura 29– Tabelas do modelo original, antes das refatorações.	73
Figura 30– Tabelas após as refatorações da Atividade 8.....	84
Figura 31– Tabela productivity_raw após a refatoração “Unir Colunas”.	85
Figura 32 - Modelo final após todas as refatorações	91
Figura 33– Tempos coletados para selecionar dados antes e depois das refatorações	92
Figura 34– Tempos coletados para inserir dados antes e depois das refatorações .	93
Figura 35– Tempos coletados para selecionar dados antes e depois da refatoração de “Unir colunas”	94
Figura 36– Tempos coletados para selecionar dados antes e depois de adicionar o índice	95

LISTA DE TABELAS

Tabela 1 – Diferentes tipos de eventos e suas ocorrências.....	44
Tabela 2 – Conjunto de refatorações utilizadas no estudo de caso.....	75
Tabela 3 – Tempos gastos para inclusão de dados antes da refatoração.	80
Tabela 4 – Tempos gastos para seleção de dados antes da refatoração.	81
Tabela 5 – Tempos gastos para inclusão de dados depois da refatoração.	87
Tabela 6 – Tempos gastos para seleção de dados depois da refatoração.	88
Tabela 7 – Tempos gastos para seleção de dados após a refatoração “Unir colunas”.	89
Tabela 8 – Tempos gastos para seleção de dados após a refatoração introduzir o índice.	90

LISTA DE SIGLAS

ABPMP	<i>Association of Business Process Management Professionals</i>
BEPEL4WS	<i>Business Process Execution Language for Web Services</i>
BPMI	<i>Business Process Management Initiative</i>
BPMN	<i>Business Process Modeling Notation</i>
BPMS	<i>Business Process Management Systems</i>
DBA	<i>Database Administrator</i>
BPM	<i>Business Process Management</i>
IDEF	<i>Integrated Definition</i>
PDCA	<i>Plan - Do - Check- Act</i>
OMG	<i>Object Management Group</i>
SOAP	<i>Simple Object Access Protocol</i>
SQL	<i>Structured Query Language</i>
TI	<i>Tecnologia da Informação</i>
UML	<i>Unified Modeling Language</i>
XML	<i>eXtensible Markup Language</i>
XPDL	<i>XML Process Definition Language</i>
WfMC	<i>Workflow Management Coalition</i>

SUMÁRIO

1. INTRODUÇÃO	14
1.1. JUSTIFICATIVA	15
1.2. OBJETIVOS	15
1.3. ORGANIZAÇÃO DO TRABALHO.....	16
2. METODOLOGIA DA PESQUISA	18
2.1. PLANEJAMENTO.....	20
2.2. PREPARAÇÃO	21
2.3. PROJETO	21
2.4. ESCOLHA DO CASO.....	23
2.5. EXECUÇÃO	23
2.6. ANÁLISE	23
2.7. COMPARTILHAMENTO.....	24
2.8. CONSIDERAÇÕES FINAIS DESTE CAPÍTULO	24
3. TRABALHOS RELACIONADOS	25
3.1. EVOLUÇÃO DE BANCO DE DADOS	25
3.2. FERRAMENTAS PARA EVOLUÇÃO DE BANCO DE DADOS	27
3.3. REFATORAÇÃO DE BANCO DE DADOS	30
3.4. CONSIDERAÇÕES FINAIS DESTE CAPÍTULO	36
4. PROCESSOS DE NEGÓCIO	37
4.1. PROCESSO DE NEGÓCIO	38
4.2. MODELAGEM DE PROCESSOS.....	39
4.3. TÉCNICAS PARA MODELAGEM DE PROCESSOS	40
4.4. DESCRIÇÃO DO BPMN.....	42
4.5. CONSIDERAÇÕES FINAIS DESTE CAPÍTULO	52

5.	UM NOVO PROCESSO PARA REFATORAÇÃO DE BANCOS DE DADOS	54
5.1.	O NOVO PROCESSO	54
5.1.1.	Backup do Esquema Original	56
5.1.2.	Identificar o Conjunto de Refatorações	56
5.1.3.	Selecionar uma Refatoração	56
5.1.4.	Identificar as Entidades Envolvidas	57
5.1.5.	Escolher a Melhor Ordem das Entidades	57
5.1.6.	Executar um Conjunto de Consultas Antes da Refatoração	57
5.1.7.	Guardar o Tempo de Execução das Consultas Antes da Refatoração	57
5.1.8.	Executar uma Refatoração	57
5.1.9.	Recuperar o Esquema Original	58
5.1.10.	Salvar o Que Foi Executado No Banco de Dados	58
5.1.11.	Executar um Conjunto de Consultas Depois da Refatoração	58
5.1.12.	Guardar o Tempo de Execução das Consultas Depois da Refatoração	59
5.1.13.	Gerar o Relatório das Refatorações	59
5.2.	O PROCESSO PROPOSTO POR AMBLER	60
5.2.1.	Problemas do processo proposto por Ambler	63
5.3.	CONSIDERAÇÕES FINAIS DESTE CAPÍTULO	65
6.	ESTUDO DE CASO	68
6.1.	O PROCESSO DE REFATORAÇÃO	73
6.1.1.	Atividade 1 - Backup do esquema original	74
6.1.2.	Atividade 2 - Identificar o conjunto de refatorações	74
6.1.3.	Atividade 3 - Selecionar uma refatoração	75
6.1.4.	Atividade 4 - Identificar as entidades envolvidas	77
6.1.5.	Atividade 5 - Escolher a melhor ordem das entidades	78

6.1.6.	Atividade 6 - Executar um conjunto de consultas antes da refatoração	79
6.1.7.	Atividade 7 - Guardar o tempo de execução das consultas antes da refatoração	80
6.1.8.	Atividade 8 - Executar uma refatoração	81
6.1.9.	Atividade 9 - Salvar o que foi executado no banco de dados	85
6.1.10.	Atividade 10 - Executar um conjunto de consultas depois da refatoração	86
6.1.11.	Atividade 11 - Guardar o tempo de execução das consultas depois da refatoração	86
6.1.12.	Atividade 12 - Mostrar o relatório das refatorações	91
6.2.	CONSIDERAÇÕES FINAIS DESTE CAPÍTULO	95
7.	CONCLUSÕES	97
7.1.	EVOLUÇÃO DO TRABALHO	97
7.2.	CONTRIBUIÇÕES DO TRABALHO	98
7.3.	SUGESTÕES PARA TRABALHOS FUTUROS	99
7.4.	CONSIDERAÇÕES FINAIS	99
	REFERÊNCIAS BIBLIOGRÁFICAS	100
	APÊNDICE A – SCRIPTS PARA SELECIONAR E INSERIR DADOS (ATIVIDADE 6 - SEÇÃO 6.1.6)	109
	APÊNDICE B – SCRIPTS PARA SELECIONAR E INSERIR DADOS (ATIVIDADE 10 - SEÇÃO 6.1.10)	110

1. INTRODUÇÃO

A evolução de esquemas de bancos de dados tem sido reconhecida como um dos principais obstáculos que dificultam as atualizações em sistemas de informação (AMBLER; SADALAGE, 2006). Estudos mostram que essas dificuldades aumentaram com o uso de sistemas Web, nos quais a frequência de mudanças é muito grande e praticamente não há tolerância para erros ou indisponibilidade dos serviços.

Com a chegada das Metodologias Ágeis para Desenvolvimento de *Software* (BECK et al., 2014), alguns conceitos do modelo tradicional foram substituídos por modelos iterativos e incrementais, que não precisam ter todo o modelo lógico e físico do banco de dados para o início do desenvolvimento do software. Isso significa organizar o desenvolvimento do sistema em várias iterações de desenvolvimento e entregas ao cliente (LARMAN; BASILI, 2003). Cada iteração contempla todas as fases já conhecidas nos modelos tradicionais como planejamento, especificação de requisitos, projeto, codificação e testes (PRESSMAN, 2010). Essas iterações que normalmente duram em torno de duas semanas, têm como principais vantagens a detecção antecipada de erros de requisitos e de implementação e também a flexibilidade de mudar os requisitos no decorrer do projeto. Para atender a essas mudanças de requisitos é necessário que o banco de dados também tenha um desenvolvimento iterativo, ou seja, à medida que o *software* é desenvolvido, o banco de dados precisa ser alterado para acompanhar os novos requisitos (RODDICK, 1995). As alterações no esquema de banco de dados são chamadas de refatorações de banco de dados.

Refatorar um banco de dados é mais trabalhoso do que refatorar um código-fonte de uma aplicação, embora, segundo Martin Flower (1999) possa seguir as mesmas premissas. Tal afirmação baseia-se no fato de que é preciso se preocupar com todas as instâncias das aplicações que utilizam o banco de dados; realizar as alterações respeitando os requisitos dos usuários; preservar os dados existentes; não modificar a semântica do esquema atual e manter a integridade do banco de dados.

As técnicas de refatoração de código de Martin Flower (1999) foram adaptadas para banco de dados por Scott Ambler e inseridas na literatura de Métodos Ágeis para Desenvolvimento de Software (AMBLER, 2002). Ambler definiu uma refatoração de banco de dados como uma pequena mudança no esquema que melhora o projeto, não adiciona funcionalidades e não altera a semântica informacional ou comportamental do esquema (AMBLER; SADALAGE, 2006).

Entretanto, nem todas as alterações se encaixam nessa definição de refatoração. É muito comum a necessidade de implementação de grandes alterações em bancos de dados. Isso implica escolher um conjunto de refatorações que represente o estado final desejado do esquema do banco de dados. Para a escolha desse conjunto de refatorações é necessário o conhecimento prévio do esquema atual do banco de dados, bem como suas limitações e objetivos. Também é necessária a escolha de um conjunto de tarefas que modifique o esquema sem negligenciar a integridade do banco de dados.

1.1. JUSTIFICATIVA

A evolução de bancos de dados baseada em refatorações foi proposta por Ambler e inspirou diversos trabalhos aqui relatados, mas possui diversas limitações, tais como: realizar diversas refatorações em um mesmo processo; salvar o histórico dos passos dessas refatorações e sugerir refatorações importantes para o desempenho do modelo do banco de dados.

Ambler apresentou um catálogo de refatorações que incluem os passos para execução de cada uma. No entanto, algumas alterações no esquema exigem que várias refatorações sejam feitas em um mesmo processo, ou seja elas podem ser combinadas e muitos passos podem ser suprimidos.

Na pesquisa bibliográfica, apresentada no Capítulo 3, não foi encontrado nenhum processo que trate uma grande refatoração.

1.2. OBJETIVOS

O objetivo principal deste trabalho é contribuir para a literatura de evolução de banco de dados, apresentando um novo processo de refatoração de banco de dados,

definindo um conjunto de atividades, a ordem destas atividades e o resultado de cada uma. Este processo tem como meta possibilitar a execução de uma grande alteração de banco de dados que inclua diversas refatorações para atender uma necessidade do usuário deste banco de dados.

Utilizou-se a notação BPMN para representar os passos desse novo processo de refatoração. A representação em BPMN abrange não somente os aspectos físicos do modelo do banco de dados, mas também as interações e decisões do usuário, a documentação de todo o processo e a possibilidade de retornar ao estado antes das refatorações caso haja erros ou os resultados não sejam satisfatórios.

O processo proposto foi comparado com um processo amplamente utilizado na literatura de banco de dados para verificar o que foi melhorado e quais problemas foram resolvidos.

Por último, foi feito um estudo de caso em um banco de dados relevante e com necessidade de uma grande refatoração. O banco de dados escolhido foi de um sistema de informações geográficas, que possui características de modelagem que prejudicam o desempenho de suas consultas. Assim, o novo processo deverá ser executado para validar a sua eficiência e analisar as possíveis melhorias das consultas após o processo de refatoração.

1.3. ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado de acordo com a seguinte estrutura:

No Capítulo 2 é descrita a metodologia utilizada na pesquisa desta Tese. Os trabalhos relacionados à evolução e refatoração de banco de dados são apresentados no Capítulo 3. O Capítulo 4 é dedicado para apresentar os conceitos relevantes sobre processos de negócio e BPMN.

A proposta central desta Tese é apresentada no Capítulo 5, onde é discutido passo-a-passo o desenvolvimento do novo processo para refatoração de bancos e dados. No Capítulo 5, também é apresentada a comparação das tarefas do novo processo com um processo existente na literatura de banco de dados. No Capítulo 6 é apresentado um estudo de caso com a execução completa do processo

desenvolvido. Finalmente, no Capítulo 7 são apresentadas as conclusões sobre os resultados obtidos e as possibilidades para trabalhos futuros.

2. METODOLOGIA DA PESQUISA

A metodologia oferece meios para que a realidade observada seja abstraída, com o objetivo de orientar o caminho para atingir os objetivos de uma pesquisa. Apresenta a definição dos métodos e a forma de pesquisa utilizada no desenvolvimento do trabalho.

Segundo Oliveira (2012), método é uma forma de pensar para se chegar à natureza de um problema, quer seja para estudá-lo, quer seja para explicá-lo.

O método utilizado para atingir o objetivo deste trabalho foi organizado segundo o esquema de Yin (2010) e está apresentado na Figura 1.

Figura 1 – Esquema para o desenvolvimento de um método

Fonte: adaptado de YIN (2010)

Yin define os elementos da Figura 1 como:

- **Plano:** identifica quais são as questões da pesquisa ou outras questões que justifiquem o estudo. Nessa fase é necessário decidir se o estudo de caso é o melhor método a ser aplicado, comparando-o a outros métodos de pesquisa.
- **Projeto:** O projeto orienta o pesquisador no processo de coleta, análise e interpretação das observações. Para estudos de caso, cinco componentes de um projeto de pesquisa são especialmente importantes:

- a) as questões do estudo;
 - b) as proposições, se houver;
 - c) a(s) unidade(s) de análise (definir qual será o estudo de caso);
 - d) a lógica que une os dados às proposições; e
 - e) e os critérios para interpretar as constatações.
- **Preparação:** Nessa fase o pesquisador deve desenvolver habilidades para conduzir o estudo de caso, tais como:
 - a) formular boas perguntas;
 - b) interpretar as respostas;
 - c) deve ser imparcial, para não ser atrapalhado por suas próprias ideologias o preconceitos;
 - d) ser adaptável e flexível para que situações novas possam ser aproveitadas como oportunidades e não serem vistas como ameaças; e
 - e) deve ter noção clara sobre o assunto pesquisado.
 - **Coleta:** A evidência do estudo de caso pode vir de várias fontes, tais como: documentação, registros em arquivos, entrevistas, observação direta, observação participante e artefatos físicos. Deve-se ter bons livros-texto de apoio, criar um banco de dados do estudo de caso e manter atualizado o encadeamento das evidências.
 - **Análise:** Para a análise utilizam-se técnicas para dados quantitativos, qualitativos ou ambos. Tabulação dos resultados, elaboração de séries temporais, categorização, testes ou simulações são usadas para formular conclusões acerca do estudo realizado.
 - **Compartilhamento:** Relatar um estudo de caso significa trazer seus resultados e constatações ao encerramento. Deve-se escolher um público alvo para apresentar as evidências. O processo de compartilhamento das informações, auxilia no processo de revisão para que se alcance um alto nível

de apresentação e que outros pesquisadores tirem suas próprias conclusões a respeito do assunto.

O esquema da Figura 1, foi modificado para adaptar-se ao contexto desta Tese como mostrado na Figura 2 e detalhado nas subseções a seguir.

Figura 2 – Método de pesquisa aplicado à Tese

Fonte: Autor

2.1. PLANEJAMENTO

Nesta etapa foi definido o problema a ser estudado: “Refatoração de Banco de Dados”. A principal motivação para este estudo foi a necessidade definir um processo para realizar uma grande alteração em um banco de dados. Essas alterações podem ser requisitos dos usuários, avanços nas linguagens de programação e/ou adaptação a diferentes ambientes como dispositivos móveis.

Segundo Yin (2010), o estudo de caso é uma investigação empírica que: Investiga um fenômeno contemporâneo em profundidade e em seu contexto de vida real, especialmente quando os limites entre o fenômeno e o contexto não são claramente evidentes. É necessário ainda obter outras características técnicas, incluindo a coleta de dados e as estratégias de análise de dados. Assim a investigação do estudo de caso enfrenta a situação tecnicamente diferenciada em que existirão muito mais variáveis de interesse do que pontos de dados. Isso leva como resultado, múltiplas fontes de evidências que precisam de alguma forma convergir.

2.2. PREPARAÇÃO

O estudo começa com uma revisão minuciosa da literatura e com a proposição cuidadosa que atenda as questões ou objetivos da pesquisa. Igualmente importante é a dedicação aos procedimentos formais e explícitos ao realizar a pesquisa (YIN, 2010). Essa revisão de literatura, segundo Gil (2008) pode ser elaborada a partir de materiais já publicados, como livros, artigos, periódicos e matérias disponíveis na internet.

O Capítulo 3 apresenta a revisão literária apresentada para esta Tese. Este estudo ressalta a necessidade de uma orientação para a realização de uma grande alteração em um banco de dados, no qual, serão realizados diversos tipos de refatorações. Os tipos de refatorações foram definidos por Ambler e Sadalage (2006) e amplamente discutidos por Domingues (2011). Um dos problemas em aberto e explorado nesta Tese é a necessidade de executar diversas refatorações em um mesmo processo. Ambler e Sadalage (2006), Curino (2009), Chang (2007) e D'Souza (2009) são exemplos de autores que evidenciaram isso ao apresentar soluções, ferramentas ou mesmo processos para a execução de uma refatoração por vez em um banco de dados.

2.3. PROJETO

Nesta fase foram testadas algumas opções para representação de processos, como UML, Redes de Petri, Planejamento Automático e BPMN.

Os diagramas da UML foram de muita importância para compreender as tarefas de uma refatoração e a separação do que é requisito de usuário e requisito de sistema. Foram estudados os diagramas de Casos de Uso e de Estado (GUEDES, 2009) para representar as refatorações do catálogo do Ambler. Uma das limitações para que a UML não fosse adotada como representação padrão para esta Tese é o fato de não ser possível automatizar as suas tarefas.

As Redes de Petri foram estudadas com o objetivo de investigar a possibilidade de sequenciar diversas refatorações e paralelizar tarefas de refatorações diferentes. Isso foi comprovado no artigo **“Evolution of Databases Using Petri Nets”** apresentado no CBA (Congresso Brasileiro de Automática) de 2012 (DOMINGUES

et al., 2012). Para esse artigo foram usados exemplos pilotos elaborados especificamente para demonstrar o uso das Redes de Petri para refatoração de banco de dados. Foram pesquisados vários autores como Silva (2004) e Vaquero (2007) que usou Redes de Petri para análise e documentação de requisitos de usuários, convertendo diagramas UML em Redes de Petri. Poucos trabalhos direcionados para banco dados foram encontrados. O trabalho que mais destacou-se nesse sentido foi o de Heuser (1993) que fez uma representação do modelo E-R (Entidade-Relacionamento) utilizando Redes de Petri. Embora seja uma ferramenta completa em relação à formalização matemática e a estados dos sistemas, as Redes de Petri ainda possuem carência de ferramentas mais intuitivas e mais próximas dos usuários.

O estudo dos estados de uma Rede de Petri e o objetivo de execução automática de atividades motivaram a pesquisa sobre Planejamento Automático. As técnicas de planejamento podem ser usadas em conjunto com linguagens de análise e modelagem de requisitos, como a UML, e ferramentas para análise e simulação do modelo, como Redes de Petri, para o planejamento de tarefas em diferentes domínios de problemas (GHALLAB et al., 2004). Tais modelos possuem, de forma geral, suas ações com precondições e efeitos de suas transições e estados, restrições, recursos disponíveis e objetivos a serem atingidos (VAQUERO et al., 2007). Esse cenário representa uma estrutura semelhante ao de uma grande alteração em um banco de dados, na qual é necessário avaliar pré e pós-condições após cada alteração no esquema. São características bem semelhantes ao de um processo de uma grande refatoração de banco de dados. Porém, ainda faltava ter uma ferramenta que pudesse executar o processo completo, sem ter que passar por traduções como de UML para Redes de Petri. Além disso a integração desse trabalho com um futuro ambiente visual e iterativo com o usuário final ficaria bastante prejudicada.

O uso de Processos de Negócios (*business process*) na área de TI (Tecnologia da informação) em conjunto com as já conhecidas técnicas para desenvolvimento de software (PRESSMAN, 2010), vem crescendo, por conta da necessidade de se detalhar e acompanhar etapas do processo de desenvolvimento de *software*. Os diagramas UML como já foi citado, embora ainda sejam amplamente usados, têm

aberto oportunidade para o surgimento de novas ferramentas e notações para processos de *software*.

Acompanhando essa demanda a OMG (*Object Management Group*), órgão responsável por manter a especificação da UML, passou a manter também a especificação da BPMN (*Business Process Model and Notation*), notação criada para representação gráfica de processos de negócio. A BPMN possui um conjunto de simbologia abrangente, é intuitiva e bem formalizada. Tem suporte a uma extensa gama de ferramentas, desde plataformas livres até sofisticadas suítes de BPM (Vide Capítulo 4). Conta com extensa bibliografia, além de outras fontes de pesquisa disponíveis na internet.

Diante desta análise foi escolhida a notação BPMN como uma solução mais adequada para representação de processos de refatoração de bancos de dados.

2.4. ESCOLHA DO CASO

Para estudo de caso foi utilizado o banco de dados geográfico do sistema desenvolvido no projeto FINEP-ESALQ do Laboratório de Automação Agrícola da Escola Politécnica da Universidade de São Paulo (USP) (<http://www.laa.pcs.usp.br/html/pagina.php?p=projetos>). Diversos aspectos do banco de dados foram analisados para a conclusão que ele seria adequado para o estudo de caso, tais como: problemas de modelagem, consultas lentas e necessidade de diminuir a complexidade de *scripts* de seleção e inserção de dados.

2.5. EXECUÇÃO

O processo foi executado passo-a-passo no banco de dados escolhido para o estudo de caso. Todas as etapas foram apresentadas e discutidas no Capítulo 6.

2.6. ANÁLISE

A cada execução do processo eram coletadas informações de sucesso ou falha. A cada falha foram feitos ajustes no processo e uma nova execução era realizada. Isso repetiu-se até que os resultados fossem satisfatórios para a definição de um processo consistente para refatoração de banco de dados.

2.7. COMPARTILHAMENTO

Conforme Yin o compartilhamento das informações de um estudo de caso deve ser direcionado a um determinado público. Espera-se que esse público elabore conclusões a respeito do trabalho que possam contribuir com a pesquisa. No caso do esquema de Yin da Figura 1, ele não considera que, após o compartilhamento, seja possível retomar o fluxo das atividades. No caso da Figura 2, o retorno das informações após o compartilhamento, é usado como entrada para uma nova elaboração e execução do processo. As publicações em congressos científicos na área de Tecnologia foram de muita importância para o crescimento do trabalho como em (DOMINGUES et al., 2013a, 2013b).

2.8. CONSIDERAÇÕES FINAIS DESTE CAPÍTULO

Neste Capítulo foi apresentado o esquema para o desenvolvimento do método científico aplicado nesta Tese. Foi apresentada, passo-a-passo, cada fase do método, discutindo a importância de cada uma delas e os conhecimentos adquiridos.

Foi apresentado o caminho que foi seguido para construir o novo processo de refatoração e quais as técnicas e ferramentas que foram estudadas e utilizadas para o desenvolvimento do trabalho.

3. TRABALHOS RELACIONADOS

Neste Capítulo são apresentados os principais trabalhos relacionados à evolução e refatoração de banco de dados.

3.1. EVOLUÇÃO DE BANCO DE DADOS

A eliciação de requisitos é a fase mais importante e problemática do projeto de um sistema (SILVA, 2004). Ela envolve desenvolvedores, gerentes e usuários do sistema para discutir, coletar e disseminar o conhecimento sobre o domínio do problema do sistema. O custo dessa fase é justificável ao longo de todo o projeto, visto que a detecção e correção de erros nessa fase tornam-se mais fáceis e exigem menos recursos para serem corrigidos, quando comparados a erros detectados em outras fases do desenvolvimento. A realização de uma mudança no esquema de um banco de dados visa atender a alterações nos requisitos dos usuários e nas especificações técnicas dos bancos de dados, após a fase de projeto de um sistema.

A evolução de aplicações com processamento de grande volume de dados é conhecida como um processo altamente complexo, dispendioso e arriscado quando a evolução envolve mudanças no esquema de banco de dados (CLEVE et al., 2010). Uma mudança de esquema pode afetar muitos aspectos de um sistema. Dois problemas fundamentais devem ser considerados:

- a semântica de mudança que se refere aos efeitos da mudança de esquema, na forma geral em que o sistema organiza a informação (ou seja, os efeitos no esquema); e
- a propagação da alteração da mudança para as instâncias adjacentes, inclusive as aplicações que utilizam a base de dados (OERTLY; SCHILLER, 1989; PETERS; ÖZSU, 1997).

A preocupação de como o administrador do banco de dados pode acompanhar as mudanças foi citada por Roddick (1995) como uma importante etapa a ser considerada no processo de evolução do esquema de dados.

Liu estudou a evolução de bancos baseada na evolução de esquemas (LIU et al., 1994). A derivação do esquema atual para o novo esquema utiliza uma abordagem que Liu chamou de *Evolutionary ER Diagram* (EVER Diagram) que se baseia em derivação e versionamento do esquema. Esse diagrama especifica as relações dos atributos do esquema antigo e novo para garantir a consistência e a manutenção das aplicações que utilizam o banco de dados.

A manutenção de bancos de dados relacionais, bem como a representação de diagramas ER (Entidade – Relacionamento) foram amplamente estudadas e formalizadas por Casanova (CASANOVA et al., 1989, 1990, 1993). A otimização para diagramas ER também foi discutida por Silva em (SILVA et al., 1996). Os aspectos de integridade referencial aplicados à manutenção de esquemas foram discutidos por Laender em (LAENDER et al., 1994).

Reiter (1992 e 1995) tratou a evolução de um banco de dados sob o efeito de uma sequência arbitrária de transações de atualização. Reiter utilizou um formalismo muito parecido com a técnica de Planejamento da área de Inteligência Artificial (GHALLAB et al., 2004). Segundo o autor, foi utilizada uma linguagem de representação de primeira ordem, na qual a situação de cálculo representa uma abordagem padrão em inteligência artificial para a formalização de problemas de planejamento. Em seus trabalhos, Reiter formalizou as operações de bancos de dados exatamente da mesma forma como as ações no domínio da inteligência artificial de planejamento, amplamente exploradas em (MCCARTHY, 1968).

Domínguez et al. (2003) desenvolveu uma metodologia para evolução de bancos de dados denominada de “*Foward Strategy*”. Essa metodologia faz com que alterações realizadas no banco de dados sejam aplicadas ao esquema conceitual e propagadas automaticamente, até o esquema lógico e físico. A ferramenta desenvolvida por Domínguez captura o conhecimento sobre os mapeamentos realizados para obter o esquema lógico de um esquema conceitual. Esta informação é usada posteriormente, a fim de obter o novo esquema lógico associado ao esquema conceitual alterado.

Em cada iteração da modelagem existem dados importantes em produção que devem ser preservados. A evolução do banco de dados deve levar em conta essa

herança e permitir que alterações controladas e organizadas no modelo de dados sejam possíveis (DOMINGUES, 2011).

3.2. FERRAMENTAS PARA EVOLUÇÃO DE BANCO DE DADOS

A ferramenta chamada Squash - *SQL Query Analyzer and Schema Enhancer* foi desenvolvida para visualizar o esquema atual, analisar possíveis falhas no projeto que podem ser resolvidas com a ferramenta. A ferramenta propõe alterações no esquema do banco de dados, como criação de índices, particionamentos e normalização. Toda representação do esquema foi baseada em XML - *eXtensible Markup Language*. A ferramenta mede o tempo de execução das consultas após refatorar o esquema, adicionando índices. O trabalho não leva em conta outros tipos de refatorações, mas a ferramenta desenvolvida é uma boa opção para análise do esquema de dados (BOEHM et al., 2009).

Curino et al. (2009) apresenta a ferramenta PRISMA que auxilia os DBAs (*Database Administrators*) a prever e avaliar os efeitos das alterações nos esquemas de bancos de dados para as aplicações, executa as alterações selecionadas pelos DBAs e salva os registros de todas as alterações. Curino ressalta os benefícios da refatoração de banco de dados para o desempenho das consultas e também a necessidade de se fazer a evolução do esquema do banco de dados utilizando *workflows*.

Comyn-Wattiau et al. (2003) desenvolveu uma ferramenta para evolução de bancos de dados que orienta o desenvolvedor a fazer as alterações baseado em três premissas: a natureza da mudança, pois ele precisa se preocupar com as especificações de requisitos, regras de negócios e armazenamento físico; o tamanho da mudança: pois pequenas ou grandes mudanças afetam de maneira diferente o sistema e precisam de técnicas diferentes para serem aplicadas; e o prazo das mudanças, pois quanto antes forem feitas, mais fácil de serem aplicadas e propagadas.

Chang et al. (2007) propôs um *framework* para refatoração de bancos de dados que analisa as alterações no esquema do banco de dados e também o impacto dessas alterações nas consultas. O *framework* define um modelo lógico de alterações e

tenta descobrir e resolver inconsistências e anomalias no modelo. O trabalho é um passo sugerido para a automatização do processo de refatoração de bancos de dados. Chang formaliza o problema da refatoração de bancos de dados utilizando *Espitemic Logic* e ressalta a necessidade de ampliar o processo de refatoração para que seja capaz de tomar decisões mais independentes e também de utilizar o processo proposto em uma linguagem visual.

A ferramenta desenvolvida por D'Sousa (2009) adota um Modelo de Padrão de Projeto - *Template Design Pattern* para tornar bancos de dados independente do método. Nesse modelo D'Sousa desenvolveu uma ferramenta chamada "*The Metadata Manipulation Tool*" que utiliza metadados para adquirir o esquema de dados atual, incluindo suas tabelas, chaves primárias e chaves estrangeiras. Para cada refatoração definida por Ambler, D'Sousa propõe modelos em metadados. D'Sousa ressalta, em seu trabalho, a limitação de trabalhar com bancos de dados genéricos e bancos de dados que atentem múltiplas aplicações.

Em algumas situações as mudanças são tão significativas que não é possível utilizar a refatoração e nem a migração. Nesses casos a integração dos bancos de dados é o caminho a ser seguido.

A ferramenta ALADIM (LESER; NAUMANN, 2005) direciona esforços para a integração de dados de forma semiautomática, facilitando a adição de novas fontes de dados. A ferramenta foi especialmente desenvolvida para atender cientistas da área de ciências biológicas. Essa ferramenta utiliza referências cruzadas para descobrir relacionamentos entre atributos e entidades. ALADIM sugere um método de mineração de dados para a busca de informações duplicadas a fim de aumentar a qualidade dos dados resultantes de uma consulta e minimizar o custo da integração.

A ferramenta SEMEX (LIU et al., 2006; DONG et al., 2009) oferece uma interface para perguntas do usuário em fontes de dados estruturadas e não estruturadas, por meio de classificação de palavras chaves, extraíndo associações nos modelos e criando instâncias para referências dos reais valores dos objetos no domínio do modelo. Essa referência de conciliação pode ser baseada em palavras chaves, na similaridade de *strings*, aplicando o conhecimento global. Podem-se mapear também

fontes de dados externos utilizando a correspondência de atributos, e a descoberta de classes correspondentes no domínio.

A ferramenta iMeMeX apresentada por Dittrich e Salles (2006), gerencia dados de diferentes tipos de fontes provenientes, por exemplo, de e-mails, XMLS, *data streams*, RSS *feeds*, textos, figuras, conteúdo de pastas, entre outros. Essa ferramenta responde perguntas genéricas como buscar arquivos que tenham como parâmetro a descrição de uma figura, uma data, ou mesmo procurar palavras chaves em arquivos pré-formatados como os do LATEX. Esse contexto de busca heterogênea de formas de dados diferentes também é abordado na ferramenta, onde são usadas heurísticas complexas para o resultado da consulta, mas sem oferecer suporte automático para realização das tarefas.

Sarma (2008) e Dong (2009) apresentaram a ferramenta UDI que integra dados com alto nível de incerteza tanto na fonte de dados quanto no mapeamento. A ferramenta UDI usa esquemas de busca probabilísticos para encontrar resultados em diferentes estruturas de consultas compreendendo tabelas, tuplas utilizando GLAV *mappings* (*global-and-local-as-view*).

A ferramenta Query (HOWE et al., 2008) é uma API para consultar metadados e refinar os resultados em pequenas granularidades. Cria tabelas de assinatura (*signature tables*), que contêm as propriedades de cada fonte, e sugere que as tabelas de assinatura com propriedades semelhantes poderão ser combinadas. As perguntas são avaliadas, construindo indicações de SQL sobre as tabelas da assinatura. Somente os atributos das tabelas da assinatura envolvidos na pergunta, devem ser alcançados durante a avaliação da mesma; esta limitação é a chave do bom desempenho em muitas de séries de dados.

Dentre as ferramentas livres e comerciais destaca-se o Liquibase, escrito em JAVA que funciona em qualquer banco de dados para rastrear, gerenciar e aplicar as alterações do modelo de dados (LIQUIBASE, 2014). Nele todas as alterações do banco de dados são armazenadas em uma forma legível e rastreável em um controlador de versão. A ferramenta é constantemente atualizada pelas comunidades de software livre para atender a novas versões de banco de dados,

porém ainda há limitações dessa ferramenta para uso de banco de dados geográficos.

O Dbdeploy é uma ferramenta para desenvolvedores ou DBAs (*Database Administrators*) utilizada para evoluir ou refatorar o projeto de bancos de dados de uma forma simples, controlada e flexível (DBDEPLOY, 2014). Dbdeploy facilita o controle dos scripts em SQL para alterações no banco de dados utilizando um controlador de versões. A ferramenta permite desfazer os scripts e voltar o esquema ao estado anterior à da alteração.

3.3. REFATORAÇÃO DE BANCO DE DADOS

Segundo Ambler e Sadalage (2006) uma refatoração em um banco de dados pode ser definida como uma alteração simples em seu esquema, com o objetivo de melhorar seu projeto, preservando sua semântica informacional e sua semântica comportamental. Assim, para se realizar uma refatoração não é possível adicionar novas funcionalidades ou modificar alguma existente, não se pode alterar um dado e nem alterar o significado de um dado existente.

Em outras palavras, após a execução de uma refatoração ou um conjunto de refatorações no esquema de dados, o banco de dados deve responder às mesmas consultas que eram realizadas antes deste processo de refatoração.

A estratégia de refatoração definida por Ambler é aplicada em ambientes complexos, nos quais existem várias aplicações acessando simultaneamente o mesmo banco de dados. Quando esse banco precisa ser alterado, praticamente todas as aplicações precisam sofrer algum tipo de adaptação, sendo que não é possível supor que todas essas aplicações serão alteradas e implantadas no mesmo momento (DOMINGUES et al., 2011). Por esse motivo, Ambler propôs que se tenha um período de transição entre o início e o fim de uma refatoração, no qual o esquema antigo e o esquema novo coexistam, como mostrado na Figura 3.

Figura 3 - Ciclo de vida de uma refatoração

Fonte: adaptado de AMBLER, SADALAGE (2006)

Ambler destaca alguns problemas no banco de dados que podem exigir refatorações como: tabelas ou colunas com múltiplas funções; tabelas com muitas colunas; e tabelas com colunas ou associações obsoletas.

Para abranger todos esses problemas e outros que podem interferir no bom desempenho do modelo de dados, e também para atender mudanças de requisitos, Ambler e Sadalage (2006) escreveu um catálogo para as refatorações que foi amplamente discutido por Domingues (2011).

Essas refatorações serão apresentadas a seguir divididas em 4 grupos: estrutural, qualidade de dados, integridade referencial e arquitetural.

i. Estrutural

As refatorações estruturais têm como objetivo melhorar o esquema do banco de dados. Mudanças de requisitos são as principais causas para se alterar a estrutura do esquema do banco de dados. As refatorações estruturais podem incluir alterações no esquema do banco de dados, tais como:

- Remover tabelas, colunas ou visões;
- Dividir ou unir tabelas ou coluna; e
- Renomear tabelas, colunas ou visões.

ii. Qualidade de Dados

As refatorações de qualidade de dados têm como objetivo aumentar as validações e diminuir as inconsistências do banco. As refatorações de qualidade de dados podem incluir ações como:

- Introduzir restrição de coluna, valor padrão ou formato comum;
- Aplicar código ou tipo padrão;
- Consolidar estratégias de chaves;
- Remover restrição de coluna, valor padrão ou restrição de não nulo; e
- Alterar colunas para receber valores nulos.

iii. Integridade Referencial

As refatorações de integridade referencial têm como objetivo manter consistentes as referências entre as tabelas, incluindo ou removendo regras, armazenando históricos ou implementando lógicas da aplicação no banco de dados. As refatorações de qualidade de integridade referencial podem incluir ações como:

- Adicionar ou remover restrição de chave estrangeira; e
- Adicionar ou remover a exclusão em cascata.

iv. Arquitetural

O objetivo principal das refatorações arquiteturais é mover uma parte do código dos aplicativos para o banco de dados (lógica no banco de dados). Os objetivos secundários são organizar, padronizar e melhorar o desempenho do banco de dados. As refatorações arquiteturais podem incluir ações como:

- Introduzir índice, tabela espelho, tabela somente para leitura; e
- Adicionar métodos (*Stored Procedures*).

Nem todas as alterações no esquema do banco de dados podem ser definidas como refatorações. Algumas alterações inserem novos elementos ao esquema do banco de dados. A essas alterações é dado o nome de transformações (AMBLER; SADALAGE, 2006). As transformações estruturais podem incluir, no esquema do banco de dados, tabelas, colunas e visões. Neste trabalho não foram abordadas alterações desse tipo.

Algumas refatorações propostas por Ambler e Sadalage (2006) necessitam de replicação de dados para manter os dados atualizados durante o período de transição, tais como: renomear tabelas ou colunas; consolidar estratégias de chaves e dividir ou unir tabelas ou colunas. Para esses casos Ambler propõe 2 estratégias que necessitam de sincronização dos dados: o uso de *triggers* síncronos e processo em lote (*batch*).

O código do *trigger* é acoplado a uma tabela e é disparado a cada transação executada. O banco de dados executa o código de apoio, passando para o *trigger* informações como qual operação foi executada, a tabela, coluna ou linha onde os dados serão atualizados. A sincronização feita em lote, baseia-se em um código de apoio que percorre toda a tabela origem para descobrir o que foi alterado e acrescentado, comparando com a tabela destino para manter a sincronização dos dados.

Ambler e Sadalage (2006) enfatiza que o uso de *triggers* síncronos é a melhor solução para replicação de dados durante a refatoração, mas segundo Domingues et. al. (2009 e 2011) a utilização de *triggers* síncronos como código de apoio, apresenta limitações, como a necessidade de uma codificação específica para realizar a sincronização e a necessidade de tratar possíveis ciclos dos gatilhos disparados. Essas dificuldades apresentadas por Domingues, podem causar lentidão nas transações, pois elas só serão finalizadas ao término do último gatilho da sequência.

Para superar essas dificuldades, uma abordagem alternativa consiste em minimizar os efeitos negativos do processo síncrono, substituindo-o por uma abordagem assíncrona para os casos de refatorações. Domingues definiu três etapas para o

processo de replicação de dados durante uma refatoração: Coleta da transação, Mapeamento e Execução. Tais etapas serão descritas a seguir.

i. Coleta da transação

Utiliza-se um *trigger* denominado “*trigger* coletor”, que é utilizado para todos os tipos de refatorações existentes, com o objetivo de capturar todas as informações da transação, qual foi a operação (inserção, exclusão ou atualização), quais os valores anteriores existentes na tabela e quais os novos valores usados para a atualização.

Figura 4 - Replicação de dados Assíncrona: etapa de coleta de dados

Fonte: Adaptado de DOMINGUES (2011)

ii. Mapeamento

Nessa etapa são definidos os mapeamentos de origem e destino das transações.

Figura 5 - Replicação de dados Assíncrona: Etapa de mapeamento

Fonte: Adaptado de DOMINGUES (2011)

iii. Execução

Nesta etapa é disparado um processo que sincroniza o esquema antigo com o novo. Assim, as informações da coleta são processadas e são realizados os mapeamentos e gravados os logs de execução (DOMINGUES, 2011).

Figura 6 - Replicação de dados Assíncrona: Etapa de execução

Fonte: Adaptado de DOMINGUES (2011)

Segundo Domingues; Kon e Ferreira (2009), essas três etapas fazem todo o trabalho esperado do código de apoio. Quando elas estão dentro dos *triggers*, no formato proposto por Ambler, a atualização do esquema é feita de forma síncrona. Uma atualização assíncrona é feita quando somente a etapa de coleta é executada junto com a transação de aplicação e a etapa de execução é realizada posteriormente.

Essa divisão em etapas auxilia no desenvolvimento de ferramentas automáticas para realizar as refatorações, permitindo aos desenvolvedores definir mapeamentos distintos para cada tipo de refatoração, pois apenas será codificado um *trigger* para todas as refatorações. Esta solução detecta circularidade dos *triggers* na etapa de execução, suas transações são rápidas com disparo de apenas um *trigger*, gera *logs* com possíveis erros de execução, possibilita a construção de aplicativos

automatizados para refatoração e não exige que as consultas SQL sejam reescritas dinamicamente para as diversas versões existentes do banco de dados.

As abordagens síncrona e assíncrona têm como limitação não proporcionar uma organização das refatorações em conjuntos, não permitindo a realização de uma grande alteração no banco de dados. Esses conjuntos de refatorações devem ser planejados para modificar o esquema de um banco de dados já existente, sem causar perda de dados.

Na organização de uma sequência de refatorações, deve-se pensar nas possibilidades de realizar uniões (*merges*), aninhamentos e intercalações de refatorações. Uma organização bem feita permitirá minimizar a duração de uma grande alteração de banco de dados (DOMINGUES, 2011).

3.4. CONSIDERAÇÕES FINAIS DESTE CAPÍTULO

Na literatura sobre evolução de bancos de dados há diversos trabalhos que se baseiam nas refatorações de Ambler para melhorar o projeto, resolver anomalias ou melhorar o desempenho das consultas (CHANG et al., 2007; D'SOUSA; BHATIA, 2009; DOMINGUES et al., 2011) . Muitas dificuldades são identificadas no que se refere a aplicar técnicas que funcionem em qualquer esquema de banco de dados, tais como, a replicação de dados necessária em algumas refatorações; melhorar o desempenho do banco de dados e também na automatização das tarefas de refatoração.

As refatorações de bancos de dados propostas por Ambler (2006) compõem um conjunto muito importante de técnicas e boas práticas já existentes. Esta técnica foi aqui apresentada como uma importante ferramenta para evolução de bancos de dados.

Entretanto em nenhum dos trabalhos relacionados apresentados neste Capítulo há uma definição de um processo para refatoração de banco de dados que trate um conjunto de refatorações e represente uma grande alteração.

4. PROCESSOS DE NEGÓCIO

Davenport (1993) define processo como uma ordenação específica das atividades de trabalho no tempo e no espaço, com um começo, um fim e entradas e saídas claramente identificados, ou seja, uma estrutura para a ação (DAVENPORT, 1993).

A ISO 9000 (ABNT, 2005) define processo como “um conjunto de atividades inter-relacionadas ou iterativas, que transformam entradas em saídas”.

Segundo ABPMP - *Association of Business Process Management Professionals* (2009), um processo é um conjunto definido de atividades ou comportamentos executados por humanos ou máquinas para alcançar uma ou mais metas. Os processos são disparados por eventos específicos e apresentam um ou mais resultados que podem conduzir ao término do processo ou a transferência de controle para outro processo. Assim, processos são compostos por várias tarefas ou atividades inter-relacionadas que solucionam uma questão específica (Figura 7).

Pavanini Júnior e Scuglia (2011) compartilham com a ABPMP o conceito de que processos são formados por atividades e tarefas, mas assumem um conceito próprio para cada uma delas. Para eles, atividade representa algo que tenha conexão com “o que fazer” e tarefa é algo capaz de detalhar uma atividade no sentido de explicar “como fazer”. Assim para os autores um “o que fazer” (atividade) será composto por diversos “como fazer” (tarefas). A Figura 7 apresenta essa diferença conceitual.

Figura 7 – Diferença conceitual entre processo, atividades e tarefas

Fonte: adaptado de PAVANINI JÚNIOR, SCUCUGLIA (2011)

4.1. PROCESSO DE NEGÓCIO

Um processo de negócio (*Business Process*) é definido por um conjunto de atividades que juntas realizam um objetivo de negócio, normalmente dentro do contexto de uma ou mais organizações (DUMAS et al., 2005). As atividades podem ser ou não automatizadas, mas precisam ter bem definidos seu início e término (WFMC, 1999).

A gestão de processos de negócios (BPM – *Business Process Management*) tem por objetivo contribuir para a sistematização da estrutura de qualquer organização utilizando sistemas ou *softwares* que apoiem sua atuação sobre o modelo funcional proposto (VALLE; OLIVEIRA, 2013). O gerenciamento de processos de negócios possui um ciclo a ser seguido, para que se tenha sucesso na implantação. Segundo Valle e Oliveira o ciclo de BPM inclui: planejamento; modelagem e otimização de processos; implantação de processos e controle e análise de processos.

- Planejamento: Tem o propósito de definir as atividades de BPM que contribuirão para o alcance das metas organizacionais.
- Modelagem e Otimização de processos: Essa fase engloba atividades que permitem gerar informações sobre o processo atual conhecido como conceito “As Is” e a proposta de processo futuro, representado pelo conceito “To be” (SHARP; MCDERMOTT, 2001). Modelar o processo na situação atual inclui: compreender os processos atuais, seu modo de atuação, falhas e necessidades; documentação; integração entre processos e priorizar soluções. A modelagem de processos para situações futuras inclui: otimizar processos; simulações; mudanças nos novos processos e inovações (VALLE; OLIVEIRA, 2013).
- Implantar processos: Engloba atividades que garantirão o suporte à implantação e à execução dos processos, tais como: coordenar ajustes de equipamentos e *softwares*; coordenar testes; treinamentos e realizar mudanças ou ajustes de curto prazo.

- Controlar e Analisar Processos: Engloba atividades relacionadas ao controle geral do processo com o uso de indicadores como: métodos estatísticos e diagramas de causa e efeito. Essas atividades servirão de entrada para as atividades de otimização e planejamento. Nesta etapa é possível registrar o desempenho de processos ao longo do tempo e auditorias do processo em uso.

4.2. MODELAGEM DE PROCESSOS

A ideia básica de processos de negócio é identificar o conjunto de atividades que tem de ser realizada para alcançar um objetivo específico e definir uma ordem em que essas tarefas devem ser realizadas (DECKER et al., 2010).

A modelagem visa criar um modelo de processos por meio da construção de desenhos e diagramas operacionais sobre seu comportamento. A modelagem valida o projeto, testando suas reações sobre diversas condições para certificar que seu funcionamento atenderá aos requisitos globais estabelecidos, além de medir seu desempenho no ponto de vista de qualidade, performance e custo (ABPMP, 2009; VALLE; OLIVEIRA, 2013).

Um modelo de processo pode conter: um ou mais diagramas; informações de objetos; informações de relacionamentos entre os objetos; informações de iteração das atividades do processo com pessoas, com sistemas de informações ou funções automatizadas (ABPMP, 2009).

Segundo Valle e Oliveira (2013) a modelagem deve atingir os seguintes objetivos: entendimento, aprendizado, documentação e melhoria, aos quais pode-se aplicar o ciclo PDCA (*Plan, Do, Check, Act*) visando a busca da melhoria contínua, conforme a Figura 8.

Figura 8 – Ciclo PDCA (*Plan, Do, Check, Act*)

Fonte: adaptado de VALLE, OLIVEIRA (2013)

A cada rodada do ciclo PDCA, são conseguidos alguns avanços em termos de melhoria, sendo que o final de um ciclo produz desafios que deverão ser superados pelo próximo ciclo, como contradições sobre a maneira correta de executarem as atividades dos processos ou conflitos de interesses e requisitos (CAMPOS, 2013; VALLE; OLIVEIRA, 2013)

4.3. TÉCNICAS PARA MODELAGEM DE PROCESSOS

Dentre as técnicas para modelagem de processos mais difundidas atualmente estão: Fluxogramas, BPMN (*Business Process Modeling Notation*), UML (*Unified Modeling Language*), IDEF0 e IDEF3 (*Integrated Definition*) e Redes de Petri (GIAGLIS, 2001). Entretanto BPMN é uma das notações mais amplamente utilizada para modelar processos (VALLE; OLIVEIRA, 2013). Essa notação é mantida pela OMG (*Object Management Group*) com suporte de empresas como IBM, Microsoft, Unisys e outras. A OMG também é responsável por manter as especificações da UML.

Em 2005, um grupo conhecido por *Business Process Management Initiative* (BPMI) iniciou o desenvolvimento de uma notação gráfica para representação de processos de negócio chamada BPMN. Neste mesmo ano, o *Workflow Management Coalition* (WfMC), também começou a construir um padrão formal, escrito para a representação de processos, chamado *XML Process Definition Language* (XPDL),

porém sem representação gráfica (CAMPOS, 2013). A XPDL 2.0 é uma linguagem para descrição de *workflows* (VALLE; OLIVEIRA, 2013).

A notação BPMN passou a ser mantida pela OMG em 2006 que uniu a notação BPMN com a notação XPDL e hoje a notação BPMN é considerada uma linguagem formal de representação. Essa integração facilitou o desenvolvimento e utilização em ferramentas diferentes, desde que essas ferramentas de modelagem sejam construídas conforme os padrões publicados pela OMG.

O Diagrama de Processos de Negócio (DPN) proposto pelo BPMN visa o mapeamento do processo, sem se estender aos demais aspectos de arquitetura do processo, como organização, tecnologia e informações, por exemplo.

Segundo Valle e Oliveira (2013) um dos pontos fortes do BPMN é o fato de que o DPN pode ser integrado a um ambiente operacional e automatizado. Outro ponto forte é assegurar que as linguagens XML desenhadas para execução de processos de negócio, como BEPEL4WS (*Business Process Execution Language for Web Services*), possam ser visualizadas dentro de uma notação orientada a processos.

A sequência de passos necessários para que se possa atingir a automação de processos de negócio, de acordo com um conjunto de regras definidas é chamada de *workflow* (ABPMP, 2009). Esta automação pode ser no todo ou em parte do processo de negócio, durante a qual documentos, informações, ou tarefas são passadas de um participante para outro (AALST, 2004).

A automatização de processos é feita por meio de uma categoria de *software* chamada de BPMS (*Business Process Management Systems*) que automatiza a gestão de processos de negócio.

Há no mercado diversas ferramentas que dão suporte a construção de *workflows* como: BizAgi BPM Suite, Bonita, AgilePoint BPMS, JBoss jBPM e Oracle BPM Suite. A maioria dessas ferramentas disponibiliza, gratuitamente, o módulo para modelagem de processos e cobra para utilizar módulos que dão suporte a automatização, integração com bancos de dados ou monitoramento em tempo real.

4.4. DESCRIÇÃO DO BPMN

As atividades de um processo podem ser executadas por diferentes atores: usuários e sistemas, portanto, um processo é uma forma de trabalho colaborativo. Um processo de negócio pode incluir a execução de atividades em paralelo e iterações entre subprocessos (DECKER et al., 2010).

Embora a notação BPMN seja rica em elementos de modelagem, os elementos mais utilizados na modelagem de processos de negócio são somente 4: atividades (*tasks*), eventos (*events*), decisões (*gateways*) e sequências de fluxos (*sequences flows*), como mostra a Figura 9. Com apenas esses quatro elementos é possível construir modelos de processos bastante intuitivos e de fácil leitura (VALLE; OLIVEIRA, 2013).

Figura 9 – Elementos Básicos da notação BPMN

Fonte: Adaptado de OMG (2011)

Atividade (*Activity*): Uma Atividade é um termo genérico do trabalho realizado em um processo. Uma atividade pode ser atômica ou composta. Se ela for atômica não possui outras atividades dentro dela, então ela é definida também uma Tarefa (*task*). Se ela for composta, ou seja, contém outras atividades dentro dela, ela é classificada como um processo ou subprocesso (Figura 10).

Figura 10 – Tipos de Atividade da notação BPMN

Fonte: Adaptado de OMG (2011)

A notação BPMN sugere alguns símbolos que podem ser adicionados à tarefa para representar visualmente sua utilização (Figura 11):

Figura 11 – Tipos de Tarefas da notação BPMN

Fonte: adaptado de SILVA, 2013

Evento (*Event*): Um evento é algo que acontece durante o curso de um processo e afeta o seu fluxo. Há três tipos de eventos, com base em quando e como eles afetam o fluxo: de início, intermediário e de fim, como mostra a Figura 12.

Figura 12 – Tipos de Eventos da notação BPMN

Fonte: Adaptado de OMG (2011)

A BPMN permite que se especifique o tipo de disparo do evento, e representá-lo com um ícone representativo. A Tabela 1 apresenta os principais disparos para eventos. A referência completa pode ser vista em (OMG, 2011; CAMPOS, 2013; VALLE; OLIVEIRA, 2013):

Tabela 1 – Diferentes tipos de eventos e suas ocorrências

Identificação	Descrição	Tipos de Eventos
		Início	Intermediário	Final
Mensagem	Envio e recebimento de mensagens.
Marca de tempo	Limite de tempo de espera.
Regras	Reação a alterações nas condições de negócio ou a regras de negócio.
Erro	Captura ou inserção de erros pré-identificados.
Link	Dois eventos de conexão equivalem a um fluxo de sequência. Em um diagrama extenso é preferível que se divida o fluxo utilizando este tipo de evento.
Cancelamento	Cancelam uma transação ou um processo.
Final	Termina o processo

Fonte: Adaptado de OMG (2011)

Conector de Sequência de fluxo (*Sequence flow*): Um fluxo de sequência mostra a ordem em que as atividades serão executadas no processo. O conector indica que existe uma dependência e uma prioridade entre as atividades conectadas para seu início e fim. Uma sequência de fluxo conecta atividades, decisões e eventos uns aos outros dentro de uma piscina (*pool*) (RADEMAKERS, 2012). Tipicamente, uma atividade terá uma única entrada e uma única saída (WHITE; MIERS, 2008). A Figura 13 mostra os diferentes tipos de fluxo de sequência.

Figura 13 – Tipos de fluxo de sequência da notação BPMN

Fonte: adaptado de SILVA, 2013

Decisão: É um elemento de modelagem utilizado para controlar como a sequência do fluxo interage dentro de um processo ao convergir ou divergir. Um *gateway* pode ser entendido como uma pergunta que é feita em um ponto no fluxo do processo. A Figura 14 mostra os principais tipos de decisão da notação BPMN e abaixo são descritos seus diferentes tipos de uso.

Figura 14 – Tipos de Gateways da notação BPMN

Fonte: adaptado de SILVA, 3013

Decisão Exclusiva (XOR) – Baseada em Dados: Neste caso, apenas um fluxo será escolhido. O símbolo pode aparecer com ou sem o marcador “x”. Além de realizar separação de fluxos, o elemento de decisão também pode unificar fluxos distintos em uma única sequência de atividades (Figura 15). Neste caso, o gateway exclusivo implica no entendimento que, dos caminhos que convergem a ele, o primeiro que chegar dará continuidade no fluxo do processo (SGANDERLA, 2013).

Figura 15 – Decisão Exclusiva (XOR) – Baseada em Dados

Fonte: adaptado de VALLE, OLIVEIRA (2013)

Decisão Exclusiva (XOR) – Baseada em Eventos: As decisões são baseadas em um evento que ocorre naquele ponto do fluxo. Normalmente o recebimento de uma mensagem, determina qual dos caminhos será tomado ou seguido. O primeiro evento disparado cancela os demais (Figura 16).

Figura 16 – Decisão Exclusiva (XOR) – Baseada em Eventos

Fonte: adaptado de VALLE, OLIVEIRA (2013)

Decisão Inclusiva (OR): Quando o processo em execução atingir este gateway, deverá avaliar a condição relacionada, e um ou mais dos fluxos poderão ser escolhidos para continuar. Ao menos um caminho deve ser verdadeiro. Também pode-se juntar um ou mais caminhos (Figura 17).

Figura 17 – Decisão Inclusiva (OR)

Fonte: adaptado de VALLE, OLIVEIRA (2013)

Execução de Fluxos em paralelo: Este elemento de decisão representa a divisão de um fluxo em dois ou mais que serão executados paralelamente. Todos os caminhos que saem deste gateway são executados. No exemplo Figura 18, as tarefas 2, 3 e 4 serão executadas paralelamente. A tarefa 5 só será iniciada após o término das tarefas 2, 3 e 4.

Figura 18 – Execução de Fluxos em paralelo

Fonte: adaptado de VALLE, OLIVEIRA (2013)

Piscina e Raia – (Pool e Lane – Swimlanes) são compostas por Piscinas (*Pools*) e Raias (*Lanes*). São usadas para representar processos, colaboradores, organizações, papéis, áreas e responsabilidades (Figura 19). A prática comum é desenhar as piscinas e suas raias horizontalmente, mas a notação permite a representação vertical. As Piscinas podem representar:

- (a) Processos: denominada “Piscina de Processo” (*Process Pool*) ou “Piscina Caixa-branca” (*White-box Pool*) (SILVA, 2013). São piscinas que contêm representações de processos.

Nesse caso o título da piscina será o nome do processo, e cada piscina poderá conter apenas um processo, mas poderá conter quantas raias forem necessárias para caracterizar os participantes envolvidos na realização das atividades do processo (SGANDERLA, 2013).

- (b) Colaboradores: Quando a piscina representa colaboradores, ela é denominada “Piscina Caixa-preta” (*Black-box Pool*) (SILVA, 2013). Nesta representação a piscina não contém raias internas. No título terá o nome do colaborador (empresa, usuário, departamento).

Figura 19 – Exemplo de Piscina e Raia

Fonte: adaptado de SGANDERLA (2013)

Fluxo de Mensagem (Message flow): Um fluxo de mensagem é usado para mostrar o fluxo de mensagens entre participantes que estão preparados para enviar e receber mensagens. Na notação BPMN, duas piscinas separadas podem representar dois participantes em um diagrama de processos (Figura 20).

Figura 20 – Conectores para fluxo de mensagem e associação de elementos

Fonte: adaptado de VALLE, OLIVEIRA (2013)

Associação: Uma associação (dirigida ou não dirigida) é usada para associar informações a objetos de fluxo. Texto e objetos gráficos podem ser associados a objetos de fluxo, como eventos, tarefas ou *gateways*.

O exemplo da Figura 21 foi baseado em (SGANDERLA, 2013) e mostra a troca de informações da piscina “Cliente” e da piscina “Serasa” com o piscina “Processo de Concessão de Crédito”. “Cliente” e “Serasa” configuram cada uma “piscina Caixa-preta” (*Black-box Pool*), na qual seu conteúdo é transparente ao processo principal de “Concessão de Crédito”.

O “Processo Crédito” possui dois colaboradores “Gerente de Produto” e o “Analista de Crédito” cada um com tarefas diferentes e reponsabilidades diferentes. Um se comunica diretamente com o Cliente e o outro com o “Serasa”. Essa comunicação de piscina é feita por meio de fluxo de mensagem (Figura 21) e cada mensagem é passada utilizando o processo intermediário “mensagem” (Tabela 1).

Figura 21 – Processo com uso de Piscinas, Raias e Fluxo de Mensagem

Fonte: adaptado de SGANDERLA (2013)

Artefatos: Permitem que sejam mostradas informações além da estrutura básica do diagrama do processo. Há três tipos de artefatos que são padrões na notação BPMN: Objetos de dados, Grupos e Anotações (SILVA, 2013).

- **A. Objeto de dados (Data Object):** - Objetos de dados podem representar um objeto singular ou uma coleção de objetos. Objetos de dados fornecem informações sobre as atividades que serão realizadas. É representado por uma página com a ponta dobrada.
- **B. Grupos (Groups):** Um grupo é representado por um retângulo pontilhado e pode ser usado para documentação e análise de fluxos.
- **C. Anotações (Text annotation):** As anotações são mecanismos para fornecer informações adicionais para o leitor de um diagrama BPMN.

Figura 22 – Uso de Artefatos

Fonte: Autor

4.5. CONSIDERAÇÕES FINAIS DESTE CAPÍTULO

A Modelagem de processos de negócio é uma solução que está sendo muito utilizada para análise de requisitos de sistemas em conjunto com o desenvolvimento de software tradicional (PHALP; SHEPPERD, 2000). O avanço de sistemas como de comércio eletrônico levaram profissionais de Tecnologia da Informação (TI) a buscar alternativas para gerenciar os fluxos de atividades em seus sistemas. Isso em

conjunto com o avanço das ferramentas e linguagens para modelagem de processos possibilitaram que profissionais de TI utilizassem recursos que antes eram empregados somente por profissionais de administração, engenharia de produtos e engenharia de produção para modelagem ou melhoria de processos internos (MILI et al., 2010).

O uso de processos para refatorações de banco de dados foi proposto por Ambler e Sadalage (2006) e discutido em diversos trabalhos apresentados no Capítulo 3. Porém, nenhum desses autores fizeram uma análise do processo de refatoração, levando em conta seu diagrama, as atividades e condições envolvidas. A notação BPMN discutida neste Capítulo, mostrou-se uma linguagem simples para entender e ser usada por diferentes perfis de usuários (administradores, desenvolvedores, analistas ou usuários finais) e o ponto de vista de refatoração de bancos de dados, abrange todas as necessidades de modelagem de uma grande alteração. No próximo Capítulo é apresentado um novo processo para refatoração de banco de dados utilizando a notação BPMN.

5. UM NOVO PROCESSO PARA REFATORAÇÃO DE BANCOS DE DADOS

O trabalho inicial do administrador de banco de dados é coletar os requisitos de alterações dos usuários, identificar os problemas de modelagem ou que podem influenciar o desempenho da consulta. Esse conjunto de alterações no esquema de banco de dados será a entrada do processo de refatoração que levará o esquema do banco de dados do modelo original até o final. A Figura 23 mostra o contexto do processo de refatoração proposto.

Figura 23 – Contexto do processo da refatoração.

Fonte: Autor

5.1. O NOVO PROCESSO

O novo processo, apresentado na Figura 24, foi escrito seguindo as normas da notação BPMN. As tarefas propostas, fluxos e verificações são descritas a seguir.

Figura 24– Um novo processo para refatoração de banco de dados

Fonte: Autor

5.1.1. Backup do Esquema Original

Antes de se iniciar todas as refatorações, é necessário fazer um *backup* do esquema que será alterado. Em qualquer ambiente, essa tarefa é necessária. Mesmo no ambiente de desenvolvimento, é importante que se faça o teste do *backup* para a recuperação do esquema alterado.

5.1.2. Identificar o Conjunto de Refatorações

A entrada do processo é um conjunto de requisitos do usuário e uma lista dos problemas identificados no ambiente de produção. Os problemas e os requisitos devem ser traduzidos para uma lista de refatorações. Esta tarefa deve ser feita pelo administrador do banco de dados que conhece o domínio do negócio e também toda sua estrutura.

O mapeamento de um requisito ou problema para refatoração tem uma grande influência da experiência do administrador do banco de dados, mas o processo possibilita também um administrador que não tem muita experiência fazer uma refatoração e verificar se o resultado foi o esperado.

5.1.3. Selecionar uma Refatoração

O administrador deverá selecionar, do conjunto de refatorações, uma refatoração para começar. O administrador deve verificar somente se a refatoração escolhida tem pré-requisitos. Não é necessário, em um primeiro momento, escolher uma melhor ordem de execução das refatorações, pois uma melhor ordem vai depender muito do ambiente, do banco de dados e do tipo do problema ou requisito que está sendo resolvido.

A refatorações serão executadas primeiro no ambiente de desenvolvimento, depois no ambiente de testes e por último no ambiente de produção. Conforme as refatorações sejam executadas, o administrador vai acumular experiência para reordenar as refatorações para gastar menos tempo.

5.1.4. Identificar as Entidades Envolvidas

Muitas refatorações exigem alterações em várias entidades. Nestes casos, o administrador, conhecedor da aplicação, deve listar todas as entidades envolvidas na refatoração e para executar a próxima tarefa, também documentar as dependências entre elas.

5.1.5. Escolher a Melhor Ordem das Entidades

Neste momento é necessário ordenar as entidades que serão alteradas durante a refatoração. A melhor ordem é aquela que permite que todas as entidades sejam alteradas sem a perda de integridade, dados ou referências.

5.1.6. Executar um Conjunto de Consultas Antes da Refatoração

Para cada entidade envolvida na refatoração, é necessário escrever consultas que tragam os dados de suas tabelas e também das tabelas mães e filhas (aquelas que fazem referência e aquelas referenciadas). Uma possibilidade de automatizar essa tarefa é coletar do *log* do gerenciador de banco de dados, por um período representativo, todas as consultas feitas ao banco de dados. Identificar as mais executadas e as mais complexas para que sejam utilizadas nesta tarefa.

5.1.7. Guardar o Tempo de Execução das Consultas Antes da Refatoração

Mesmo que a refatoração não tenha como objetivo melhorar o desempenho do banco de dados, é muito importante coletar e guardar o tempo de execução das consultas antes de fazer qualquer refatoração. O objetivo é medir o impacto no desempenho que uma refatoração pode provocar. Caso o objetivo não seja melhorar o desempenho, também é necessário ter o cuidado de não piorar o desempenho do banco de dados.

5.1.8. Executar uma Refatoração

Esta tarefa irá alterar o esquema do banco original em direção ao modelo final esperado. Cada refatoração irá exigir um conjunto de ações que é muito específico. Assim, o administrador de banco de dados deve ler com atenção as instruções existentes nos catálogos de refatorações, existentes na literatura de banco de

dados, como o desenvolvido por Ambler e Sadalage (2006) e o catálogo reorganizado e em português desenvolvido por Domingues (2011). Em BPMN esta atividade é um subprocesso (Figura 24).

5.1.9. Recuperar o Esquema Original

Esta tarefa somente será executada se houver falha na execução de uma das refatorações. Para garantir a segurança e integridade dos dados, todo o processo será cancelado, não importando quantas refatorações já tenham sido feitas com sucesso.

5.1.10. Salvar o Que Foi Executado No Banco de Dados

Todos os *scripts* executados no banco de dados devem ser salvos para que seja possível executá-los em outros ambientes. Além deste objetivo, é importante que se salve todos os *scripts* para que seja possível realizar uma auditoria, caso ocorra algum problema. Esta tarefa pode variar bastante de complexidade, conforme a necessidade. De um modo geral, o administrador de banco, após executar os *scripts* no banco de dados, deve armazenar esse *script* em um gerenciador de versão de código (exemplo SVN ou Git).

No extremo oposto, pode ser necessário que antes da atividade “Executar uma refatoração”, seja necessário seguir um processo de mudança vigente na empresa, com todos os passos definidos neste processo para permitir a aprovação e o horário correto para executar a alteração. Neste caso, essa atividade significa submeter o *script* de alteração do banco de dados ao processo de mudança, que é será responsável em salvar o que será executado no banco de dados.

Em ambientes de desenvolvimento e teste, a ordem desta atividade deve ser o que foi proposto aqui, mas no ambiente de produção, deverá ser realizada antes da atividade “Executar uma refatoração”.

5.1.11. Executar um Conjunto de Consultas Depois da Refatoração

Com a refatoração executada, repete-se a execução do conjunto de consultas ao banco de dados. A execução com sucesso desta tarefa, indica que a refatoração foi

executada conforme o planejado. Pode ocorrer que alguma consulta específica não seja executada, devido às alterações realizadas, mas é importante obter e salvar os *logs* de erros.

Os erros ocorridos na execução desta consulta são indicativos que a refatoração realmente foi realizada, provocando a interrupção do funcionamento de uma consulta. Um conjunto desses erros, pode ser usado para exemplificar, no final de todo o processo, o que os desenvolvedores não podem mais realizar nas suas aplicações.

5.1.12. Guardar o Tempo de Execução das Consultas Depois da Refatoração

Para aquelas consultas que foram executadas com sucesso, é necessário coletar o tempo de execução ou qualquer outra métrica (por exemplo: plano de execução) para que seja possível verificar o impacto no desempenho do banco de dados.

5.1.13. Gerar o Relatório das Refatorações

Esta tarefa do processo é executada quando se esgotaram todas as refatorações do conjunto definido na tarefa: “Identificar o conjunto de refatorações”. O objetivo é compilar todas as informações existentes no processo para que se tenha um relatório. As principais informações que devem fazer parte deste relatório são:

- Lista das refatorações que foram executadas e a ordem;
- *Scripts* que *fizeram* as alterações no banco de dados;
- *Scripts* que contêm as consultas realizadas antes e depois de cada refatoração;
- Tempos antes e depois da execução destas consultas; e
- Lista das observações, comentários ou alertas dos problemas encontrados durante o processo.

Os objetivos deste relatório são permitir uma análise de impacto deste processo, documentar tudo o que foi feito e divulgar o novo esquema de banco de dados para todos os envolvidos no processo.

A seguir é apresentado o processo proposto por Ambler e as vantagens do processo proposto na Figura 24 em relação ao processo apresentado na Figura 26.

5.2. O PROCESSO PROPOSTO POR AMBLER

O processo proposto por Ambler e Sadalage (2006) não foi escrito em BPMN, como pode ser observado no diagrama original do autor (Figura 25). O processo de Ambler representa apenas um fluxograma das tarefas para o processo de uma refatoração.

É importante ressaltar, que foi feita uma extensa pesquisa bibliográfica em busca de outras definições de processos de refatoração, mas não foi encontrado nenhum trabalho neste sentido,

Olhando o diagrama da Figura 25, já se pode visualizar um círculo infinito entre as tarefas *“Write a Unit Test”* e *“Run The Tests”*. Para evitar o problema com possíveis círculos infinitos que o diagrama original possa induzir o leitor, foi necessário reescrevê-lo em BPMN como pode ser visto na Figura 26.

Pode ocorrer que o autor tenha planejado algo mais complexo que o diagrama não represente muito bem, mas para que seja possível discutir as tarefas envolvidas na refatoração foi capturado o essencial do diagrama atual e analisado com cuidado o Capítulo 3 do livro **“Refactoring Databases: Evolutionary Database Design”** (AMBLER; SADALAGE, 2006).

Figura 25 – Diagrama original para o processo de uma refatoração proposto por Ambler

Fonte: AMBLER, SADALAGE (2006)

Figura 26 – Processo de uma refatoração proposto por Ambler escrito em BPMN

Fonte: adaptado de AMBLER, SADALAGE (2006)

5.2.1. Problemas do processo proposto por Ambler

1. Tarefa: Verificar que uma refatoração é necessária. Esta tarefa não deveria fazer parte do processo, mas sim do contexto do processo, cuja responsabilidade é do administrador de banco de dados ou fazer parte de uma análise dos requisitos dos usuários ou do desempenho do banco de dados.
2. Tarefa: Escolher a refatoração certa. A forma como é definida esta tarefa, já limita todo processo para executar somente uma refatoração. Para um problema do banco de dados ou um requisito de usuário pode não existir uma refatoração certa, mas um conjunto de refatorações que levam o banco de dados a um estado final correto.
3. Tarefa: Tornar o esquema atual obsoleto (opcional). Basicamente essa tarefa tem o objetivo de avisar todos os desenvolvedores o que será feito no banco de dados, para que se preparem para realizar as possíveis alterações nos códigos fontes das aplicações que acessam o banco de dados. Considera-se que essa tarefa é muito específica à empresa que utiliza o banco de dados e ao seu processo de controle de mudança. Pode variar entre uma grande empresa que tem que planejar com muita antecedência qualquer alteração o que o administrador terceirizado irá fazer no banco de dados, até aquela empresa que possui uma equipe interna para a manutenção do banco de dados. Considerando também que Ambler coloca como opcional, deixou-se fora da proposta do novo processo.
4. Tarefas: Escrever teste unitário e Executar os testes. A metodologia e as ferramentas para escrever testes unitários para código fonte estão muito evoluídas, mas para bancos de dados falta um grande trabalho de pesquisa e desenvolvimento. Assim, optou-se por limitar essas tarefas para algo viável atualmente para verificar a situação do banco de dados: executar um conjunto de consultas (*queries*). Se essas consultas forem executadas com sucesso e com um desempenho satisfatório, significa que as tabelas envolvidas nestas consultas passaram por um teste.

5. Tarefa: Modificar o banco de dados. No processo definido por Ambler, no Capítulo 3 do seu livro “**Refactoring Databases: Evolutionary Database Design**”, não se preocupa em guardar o esquema original. Ele define outro processo no capítulo 4, intitulado “Colocando em produção” que basicamente repete o teste, salva o esquema e se der errado restaura tudo como era antes. Como o objetivo é ter uma proposta de processo de refatoração, considera-se imprescindível que se tenha um único processo que inclua todas as tarefas necessárias, até porque em qualquer ambiente (desenvolvimento, teste ou produção), é necessário ter armazenado o modelo antes de se fazer uma alteração, pois todos os ambientes devem ser preservados.
6. Tarefa: Migrar os dados. Esta tarefa é muito específica para as refatorações que precisam de migração de dados. Várias refatorações, tais como: Eliminar Coluna, Adicionar Restrição de Chave estrangeira e Introduzir Índice, não têm nenhuma migração de dados. Mesmo aquelas que têm migração de dados, há opções de abordagem para realizar essa tarefa, como a tese de Domingues (2011) para replicação assíncrona de dados em refatoração. Deste modo, é mais coerente colocar essa atividade como um detalhe da tarefa de executar a refatoração, pois cada uma tem uma exigência sobre os dados envolvidos.
7. Tarefa: Modificar os programas externos. Aqui há uma confusão entre um processo de refatoração de banco de dados com o processo de refatoração de código. O escopo de um processo de refatoração de banco de dados consiste em todas as tarefas envolvidas no banco de dados. Após o banco de dados ser alterado, isto é, após o processo terminar, pode, ser for necessário, começar um processo de refatoração do código das aplicações que acessam esse banco de dados. Não é possível alterar o código da aplicação antes de se terminar com sucesso o processo de refatoração do banco de dados. A confusão aqui é imaginar que o processo de refatoração de bancos de dados só termine quando as aplicações estiverem prontas para acessarem o novo banco. O correto é considerar que um processo de refatoração de banco de dados possa ser aplicado em ambiente de produção depois que as

aplicações já estejam preparadas para a refatoração ou que o processo não irá interferir no funcionamento das aplicações.

8. Tarefa: Controlar a versão do seu trabalho. Como o foco, novamente, é o processo de bancos de dados relacionais existentes no mercado, e estes não têm atualmente, nenhum suporte ao controle de versão, não se trata desta tarefa. Poder-se-ia imaginar que os *scripts* que irão alterar o banco de dados estejam em um controlador de versão (exemplos: SVN e Git (ROONEY; BERLIN, 2006; CHACON, 2009), mas como isso é o básico para qualquer código escrito, manter o controle dos arquivos fontes, não foi incluído na proposta.
9. Tarefa: Anunciar a refatoração. Esta tarefa tem como objetivo informar todos os envolvidos sobre as alterações que foram feitas no banco de dados. Como é uma tarefa que depende muito da organização da empresa, das equipes que estão trabalhando no banco de dados e qual é o ambiente (desenvolvimento, teste ou produção) que a refatoração foi aplicada, considera-se que essa tarefa faz parte do contexto da refatoração, após o banco de dados chegar no estado final e ficar estável. Como bem orientado pelo próprio Ambler, na página 47 do seu livro ***Refactoring Databases: Evolutionary Database Design***, *“Do not Publish Data Models Prematurely”*.

5.3. CONSIDERAÇÕES FINAIS DESTE CAPÍTULO

O novo processo tem as seguintes vantagens em relação ao processo anterior:

- a) Pode trabalhar com uma grande alteração, pois se preocupa na definição de um conjunto de refatorações e a execução de todas para considerar que o processo foi finalizado. O modelo de Ambler não se preocupa com esse aspecto, mesmo que o autor cite que pode haver refatorações que dependam de outras.
- b) A primeira tarefa do novo processo guarda um *backup* do esquema a atual. O novo processo coloca essa tarefa necessária para todos os ambientes.

- c) Trata exclusivamente das tarefas do administrador de banco de dados, não tem tarefas do desenvolvedor de aplicação, tais como “Modificar os programas externos”, como mostrado na Figura 26
- d) Ao contrário de uma tarefa genérica chamada: escrever um teste unitário, considera que é necessário escrever um conjunto de consultas para as entidades envolvidas na refatoração. Esse conjunto de consultas pode ser obtido dos logs do gerenciador de banco de dados.
- e) Coleta os tempos de um conjunto de consultas antes e depois da refatoração. Essa preocupação com o desempenho é muito relevante para todos os bancos de dados.
- f) Unifica as tarefas “Modificar o Banco de Dados” e “Migrar os Dados” em uma única, com o nome de “Executar uma Refatoração”. A documentação da refatoração informará se é necessário se preocupar com migração de dados e as alternativas disponíveis. É importante essa unificação, pois somente 20% das refatorações exigem migração de dados, conforme Domingues (2011).
- g) A tarefa genérica, “Anunciar a Refatoração”, foi substituída por uma tarefa bem definida denominada “Gerar o Relatório das Refatorações”. Esse relatório terá informações importantes sobre o impacto do processo no esquema de banco de dados. Esse relatório será a base para a decisão se o processo foi executado com sucesso e se é viável continuar o processo de refatoração para outro ambiente, como o de produção.
- h) O novo processo foi escrito em BPMN que possibilita validação, avaliação, melhoramentos e automatização. São características que oferecem segurança ao processo, pois é importante garantir que todas as tarefas sejam repetidas em outros ambientes.
- i) Se preocupa em ordenar as entidades envolvidas. Um problema prático de qualquer alteração de banco de dados que envolve mais de uma entidade é decidir qual entidade alterar primeiro. Essa questão relevante deve ser resolvida antes de executar uma refatoração.

- j) Por último, como foi feito o *backup* no início do processo e um conjunto grande de refatorações pode ser executado, se alguma refatoração deste conjunto falhar ou ocorrer qualquer problema que comprometa o processo, tem-se a tarefa de recuperar o esquema anterior e finalizar o processo.

6. ESTUDO DE CASO

Como estudo de caso foi utilizado o banco de dados geográfico de um sistema desenvolvido no âmbito do projeto FINEP-ESALQ do Laboratório de Automação Agrícola da Escola Politécnica da Universidade de São Paulo (<http://www.laa.pcs.usp.br/html/pagina.php?p=projetos>).

Esse sistema tem como objetivos: monitoramento, tomada de decisão, recomendação e intervenções para aplicação de fertilizantes em taxa variável e avaliação de indicadores. Uma série de atividades são conduzidas no projeto para utilização de sensores ativos e de imagens orbitais como indicadores de estresse, especialmente relacionados à adubação nitrogenada, prioritariamente nas culturas da cana-de-açúcar, algodão, trigo e milho. Paralelamente as tarefas do sistema devem avaliar o potencial da utilização de sensores de solo, eletroquímicos e óticos, para a implementação de algoritmos de decisão.

Espera-se poder gerar indicadores para automação do diagnóstico ou a viabilização de sensoriamento em tempo real com vistas à adubação, especialmente de nitrogênio, além da correção do solo. Para atingir esse objetivo, o projeto trabalha com uma série de tipos de sensores e conceitos de sensoriamento aplicados ao manejo de lavouras e integração de informações de diferentes fontes e níveis em sistemas que permitam tomadas de decisões rápidas, seguras e acessíveis ao usuário agricultor.

Um banco de dados que utiliza o SGDB PostgreSQL (SMITH, 2010) foi utilizado com extensão PostGIS (OSGEO, 2014) para fornecer o tipo de dados espacial e funções, a fim de armazenar e recuperar informações sobre a localização e mapeamento. Este banco de dados e suas funções programadas no PostGIS foram usados como parte da solução para o desenvolvimento de processamento de *web-services*, publicação e visualização dinâmica de dados agrícolas através de um servidor de mapas (MURAKAMI et al., 2007).

Uma fazenda é dividida em várias partes chamadas de talhões. Os talhões são criados para alternar as culturas de acordo com a safra, possibilitando um controle mais apurado dos custos de produção e as necessidades de reposição de nutrientes

de cada talhão. Os talhões são representados no banco de dados por suas dimensões geográficas. Com isso é possível pelas ferramentas de georreferenciamento do SGDB mostrar uma imagem do desenho geométrico do talhão. Com o desenho geométrico do talhão pronto, é possível dividir esse espaço em pequenos polígonos que representam o local geográfico de cada amostra de solo coletada. De cada amostra de solo são coletados informações dos nutrientes para análise posterior. Como resultado dessa análise, o sistema fornece ao usuário uma recomendação das taxas de nutrientes que precisam ser repostas ao solo. Uma abstração da descrição desse trabalho é mostrada na Figura 27.

Figura 27– Exemplo de Fazenda e Talhão

Fonte: Autor

O sistema precisa arquivar, recuperar e processar dados para análises futuras sem negligenciar os aspectos do modelo de entidade-relacionamento (CHEN, 1976). O modelo de banco de dados original do sistema é apresentado na Figura 28, onde há muitos relacionamentos que descrevem os estágios para analisar os ciclos de uma cultura (COSTA et al., 2011).

Figura 28– Modelo de dados Original

Fonte: Autor

O modelo de banco de dados foi construído com chaves naturais compostas. Há um forte acoplamento entre o modelo de banco de dados e o domínio do negócio da

agricultura de precisão. Algumas colunas de chave primária, tais como: **farm_id**, **plot_id** e **prod_id** estão presentes em quase todas as relações. Uma breve descrição das tabelas que fazem parte do modelo da Figura 28 é apresentada a seguir:

1. **Plot:** um contorno gráfico representa uma parcela da fazenda que é usada como campo de cultura, também chamado de talhão. Esta entidade deve ter um polígono associado devido ao fato de que esta informação é usada para os métodos de geoprocessamento (por exemplo, a produtividade da máquina de filtragem e métodos de interpolação)
2. **Farm:** representa o conjunto de parcelas utilizadas como campos de culturas agrícolas. A fazenda pode ser representada por uma geometria de múltiplos polígonos.
3. **Productivity:** a produtividade representa a colheita de um campo de cultura. Esta entidade contém os dados que descrevem a colheita tais como data de início e fim da colheita. Os dados reais da produtividade estão representados nas entidades *Productivity_filtered* e *Productivity_raw*.
4. **Productivity_raw:** esta entidade possui os dados obtidos da máquina colheitadeira sem qualquer tipo de limpeza de dados.
5. **Plot_grid:** representação matricial de todos os pontos utilizados para interpolações de amostras química/física do solo ou produtividade.
6. **Productivity_interpolated_poly:** resultado do processamento do *raster (bitmap)* gerado na interpolação para representação vetorial, com o objetivo de acelerar a visualização nos servidores de mapas.
7. **Productivity_filtered:** pontos com informação de produtividade da colheitadeira resultantes do algoritmo de limpeza de dados de produtividade. São removidos pontos inválidos, com resultados nulos ou fora do talhão.

8. **Analysis:** entidade que armazena os tipos de análises efetuadas nos talhões. As análises podem ser de atributos físicos/textura do solo, nutrientes químicos (Nitrogênio, Fósforo e Potássio), calagem ou outras medições.
9. **Analysis_lab:** pontos amostrados no talhão para análise efetuada em laboratório.
10. **Analysis_lab_interpolated:** pontos contendo o resultado da interpolação dos dados oriundos da tabela **Analysis_lab**.
11. **Analysis_lab_poly:** resultado do processamento do *raster (bitmap)* gerado na interpolação para representação vetorial dos pontos amostrados no talhão, oriundos da tabela **Analysis_lab**.
12. **Recommendation:** entidade de registro de criação de recomendação para aplicação de insumos.
13. **Recommendation_poly:** representação em polígonos, representando a grade de pontos para a aplicação da recomendação gerada, utilizando a fórmula gravada na tabela **Recommendation_formula**.
14. **Recommendation_formula:** fórmulas ou equações utilizadas para o cálculo da recomendação para aplicação de insumos e reposição de nutrientes.
15. **Measurements:** unidades de medida utilizadas em agricultura.
16. **User_farm:** armazena o cadastro de usuários do sistema (agricultores, técnicos agrícolas e pesquisadores).

A Figura 29 mostra as tabelas *farm*, *plot*, *productivity* e *productivity_raw* que foram selecionadas para a continuidade do estudo de caso, pois armazenam os dados mais significativos do sistema e que posteriormente serão usados para gerar gráficos e relatórios para a análise dos usuários.

Figura 29– Tabelas do modelo original, antes das refatorações.

Fonte: Autor

6.1. O PROCESSO DE REFATORAÇÃO

Antes de começar executar o processo de refatoração é importante descrever o principal requisito do usuário ou problema encontrado no sistema. No estudo de caso, devido ao grande volume de dados, à utilização de georefenciamento, à interface *Web* para os usuários finais e à integração SOAP com outros sistemas, pode-se destacar dois itens que motivaram o processo de refatoração:

- lentidão nas consultas à base dados; e
- complexidade para o desenvolvedor realizar as operações básicas (inserção, consulta, atualização e exclusão) nas principais tabelas.

As atividades do novo processo de refatoração desenvolvido para esta Tese (Figura 24) são descritas a seguir, com base na execução do processo para este estudo de caso, sendo que o novo modelo para as tabelas está representado pelo item 6.1.12.

O ambiente utilizado para realizar a refatoração no modelo da Figura 28, é descrito abaixo:

- **Máquina:** Notebook Dell Inspiro 15R SE 4670.
- **Hardware:** contém 8GB de memória, 500GB de disco rígido e processador Intel(R) Core(TM) i7-3632QM CPU @ 2.20GHz 2.20 GHz
- **Sistema operacional:** Windows 8 de 64 bits.
- **Banco de dados:** PostgreSQL 9.1
- **Extensão do Banco de dados:** PostGIS.

Durante toda a refatoração e também na execução das consultas ao banco de dados, os recursos do sistema foram monitorados (Memória, espaço em disco e uso do processador). Nenhum desses recursos sofreu falhas que comprometessem a execução do processo de refatoração. A seguir são descritas todas as etapas do processo de refatoração para o modelo de dados da Figura 24.

6.1.1. Atividade 1 - Backup do esquema original

Foi feito o backup da base de dados do estudo de caso e esse backup foi armazenado em local seguro para uma possível recuperação se ocorresse algum problema.

6.1.2. Atividade 2 - Identificar o conjunto de refatorações

Considerando as motivações apresentadas pelo usuário do sistema, as refatorações úteis são aquelas que se preocupam em melhorar o modelo de dados e o desempenho. Com essa orientação, tem-se o seguinte conjunto de refatorações para executar o processo:

Tabela 2 – Conjunto de refatorações utilizadas no estudo de caso

Refatoração		Descrição	Tipo	Impacto
1	Introduzir chave de identificação	Troca múltiplos atributos por uma chave primária	Estrutural	Melhora o modelo e o desempenho
2	Adicionar restrição de chave estrangeira	Adicionar uma restrição para garantir a integridade referencial	Integridade referencial	Melhora o modelo
3	Eliminar coluna	Excluir do modelo uma coluna	Estrutural	Melhora o modelo
4	Unir colunas	Transforma várias colunas em uma	Estrutural	Melhora o modelo e o desempenho
5	Introduzir índice	Adicionar um índice que melhora o desempenho	Arquitetural	Melhora o desempenho

Fonte: Autor

6.1.3. Atividade 3 - Selecionar uma refatoração

Para ficar mais clara a exposição das refatorações realizadas, descrevem-se, a seguir, todas as refatorações deste conjunto, considerando o estudo de caso e a ordem em que foram executadas no banco de dados:

a) Introduzir chave de identificação

Alguns arcabouços para desenvolvimento de *software*, Ruby on Rails (MICHAEL HARTL, 2012) ou Yii (MAKAROV, 2013), trazem grandes facilidades para realizar as operações básicas de inserção, exclusão, consulta e atualização. Por outro lado, exigem que a tabela tenha uma chave de identificação numérica auto incremental

com o nome de *id*. Esta refatoração introduz uma chave de identificação com essa característica e retira a anterior, que é a chave natural.

Nenhuma tabela do estudo de caso utiliza uma coluna *id* como chave de identificação. Deste modo, devido aos problemas relatados pelos usuários do sistema, foi decidido que todas as tabelas deveriam receber essa refatoração.

b) Adicionar restrição de chave estrangeira

A principal motivação é garantir a integridade referencial, utilizando regras no banco de dados, pois uma coluna que é chave estrangeira só poderá ter um valor que exista previamente.

No estudo de caso, após a introdução de chaves de identificação nas tabelas envolvidas, foi necessário colocar as restrições de chave estrangeira em cada tabela que participou desta refatoração.

c) Eliminar coluna

Quando se faz uma manutenção na tabela, é necessário verificar se alguma coluna não mais será utilizada. Caso exista, é muito importante eliminar essa coluna para limpar o modelo de dados e evitar o seu uso inadequado. Como resultado da refatoração anterior, há várias colunas cuja eliminação é possível, por exemplo as colunas **farm_id**, **plot_id** e **prod_id** que serão substituídas por colunas com o nome **id**.

d) Unir colunas

Durante a evolução do banco de dados, são criadas colunas que contêm valores complementares que nunca são utilizados isoladamente. Assim, a existência de duas colunas para armazenar os dados é desnecessária, motivando a realização da refatoração unir colunas.

A tabela *productivity_raw*, do estudo de caso, tem duas colunas que sempre são utilizadas em conjunto e não faz sentido ter duas colunas para elas: *longitude* e *latitude*. Existe um tipo especial no gerenciador de banco de dados para armazenar

essas informações: *geometry*. Esse tipo especial permitiu a realização da refatoração de unir colunas.

e) Introduzir índice

O principal motivo de se introduzir um índice em uma tabela é aumentar o desempenho de consultas em seu banco de dados. Também pode ser necessária a introdução de um índice para criar uma chave primária em uma tabela, ou para apoiar uma chave estrangeira em outra tabela. Esta refatoração foi muito útil no estudo de caso, depois da refatoração “Unir Colunas”. A nova coluna, *prod_geom*, é muito utilizada para fazer buscas no sistema.

6.1.4. Atividade 4 - Identificar as entidades envolvidas

Para cada refatoração executada, descrevem-se abaixo os detalhes da execução desta atividade:

a) Introduzir chave de identificação

As entidades envolvidas nesta refatoração são as seguintes: *farm*, *plot*, *productivity* e *productivity_raw*. Todas essas tabelas não têm uma coluna **id** auto incrementada para ser utilizada como chave de identificação.

b) Adicionar restrição de chave estrangeira

Após a refatoração anterior, foi necessário adicionar restrições de chave estrangeira nas seguintes tabelas:

- *plot*, coluna *farm_id* é chave estrangeira para a tabela *farm*, coluna *id*
- *productivity*, coluna *plot_id* é chave estrangeira para a tabela *plot*, coluna *id*
- *productivity_raw*, coluna *prod_id* é chave estrangeira para a tabela *productivity*, coluna *id*

c) Eliminar coluna

Após a refatoração anterior, foi necessário eliminar as seguintes colunas nas tabelas a seguir:

- *farm*, eliminar a coluna *farm_id*
- *plot*, eliminar as colunas *plot_id*
- *productivity*, eliminar as colunas *prod_id* e *farm_id*
- *productivity_raw*, eliminar as colunas *plot_id*, *farm_id* e *prod_point_id*

d) Unir colunas

A refatoração “Unir Colunas” alterou a tabela *productivity_raw* para realizar a seguinte tarefa:

- As colunas *longitude* e *latitude* que são do tipo *DOUBLE* foram unidas na coluna *prod_geom*, criada com o tipo *geometry*

e) Introduzir índice

A refatoração introduzir índice alterou a tabela *productivity_raw* para criar um índice na coluna *prod_geom*.

6.1.5. Atividade 5 - Escolher a melhor ordem das entidades

São apresentados a seguir os detalhes da execução desta atividade e a ordem para refatoração nas tabelas.

a) Introduzir chave de identificação

1. *farm*
2. *plot*
3. *productivity*
4. *productivity_raw*.

Neste caso a ordem não irá afetar a refatoração. Os passos a seguir seguem as definições de chave-estrangeiras do modelo da Figura 28. As tabelas filhas devem ser alteradas antes das tabelas pai.

b) Adicionar restrição de chave estrangeira

1. *productivity_raw*
2. *productivity*
3. *plot*

c) Eliminar coluna

1. *productivity_raw*
2. *productivity*
3. *plot*
4. *farm*

d) Unir colunas

A refatoração unir coluna irá alterar somente uma tabela: *productivity_raw*. Deste modo, não há preocupação com a ordem das entidades.

e) Introduzir índice

A refatoração introduzir índice irá alterar somente uma tabela *productivity_raw*. Como no item anterior, não há preocupação com a ordem das entidades.

6.1.6. Atividade 6 - Executar um conjunto de consultas antes da refatoração

Os seguintes *scripts* foram executados antes das refatorações:

- INSERT INTO PRODUCTIVITY_RAW
- SELECT ALL ROWS FROM PRODUCTIVITY_RAW

Estes *scripts* estão resumidos para não ter que descrever todas as colunas envolvidas nas tabelas, mas os *scripts* completos acrescentam somente as colunas omitidas nestes *scripts* e estão apresentados no Anexo A. Os *scripts* para seleção

de dados consideraram a recuperação de dados específicos de produtividade, ou seja, de uma colheita específica (tabela *productivity*).

A tabela escolhida para se medir os efeitos das refatorações, foi a *productivity_raw*. Esta tabela é a que contém a maior quantidade de linhas e afeta diretamente o desempenho de todo o sistema.

6.1.7. Atividade 7 - Guardar o tempo de execução das consultas antes da refatoração

Para cada *script* mostrado no item anterior, se coletou uma série de dados para avaliar o impacto das refatorações. A Tabela 3 apresenta uma amostra de 10 resultados da simulação, todos com 189.730 linhas. No caso do *script* de inclusão tem-se os seguintes tempos:

Tabela 3 – Tempos gastos para inclusão de dados antes da refatoração.

INCLUSÃO DE DADOS NA TABELA PRODUCTIVITY_RAW (ANTES DA REFATORAÇÃO)		
	LINHAS	TEMPO (MS)
1	189730	10090,705
2	189730	9114,362
3	189730	9330,685
4	189730	9002,696
5	189730	7878,450
6	189730	9268,178
7	189730	8980,788
8	189730	12202,855
9	189730	9365,448
10	189730	9111,095
	Média	9434,526
	Desvio Padrão	1113,467

Fonte: Autor

Para os *scripts* de seleção, tem-se os seguintes tempos:

Tabela 4 – Tempos gastos para seleção de dados antes da refatoração.

SELEÇÃO DE TODAS AS LINHAS DA TABELA PRODUCTIVITY_RAW (ANTES DA REFATORAÇÃO)		
	LINHAS	TEMPO (MS)
1	189730	53045
2	189730	57617
3	189730	51968
4	189730	46008
5	189730	56387
6	189730	45350
7	189730	46857
8	189730	51257
9	189730	49046
10	189730	47701
	Média	50524
	Desvio Padrão	4274

Fonte: Autor

6.1.8. Atividade 8 - Executar uma refatoração

Para cada refatoração executada, descrevem-se os detalhes da execução desta atividade. O resultado pode ser visto na Figura 30.

a) Introduzir chave de identificação

Os seguintes *scripts* foram executados para criar as novas colunas:

1. *ALTER TABLE farm ADD COLUMN id serial;*
2. *ALTER TABLE plot ADD COLUMN id serial;*
3. *ALTER TABLE productivity ADD COLUMN id serial;*

4. *ALTER TABLE productivity_raw ADD COLUMN id serial;*

Os seguintes *scripts* foram executados para acertar os valores das novas colunas:

1. *UPDATE farm SET id = farm_id;*
2. *UPDATE plot SET id = plot_id;*
3. *UPDATE productivity SET id = prod_id;*
4. *UPDATE productivity_raw SET id = prod_point_id;*

Os seguintes *scripts* foram executados para acertar os próximos valores das novas colunas:

1. *SELECT setval('public.farm_id_seq', (select max(farm_id) from farm), true);*
2. *SELECT setval('public.plot_id_seq', (select max(plot_id) from plot), true);*
3. *SELECT setval('public.productivity_id_seq', (select max(prod_id) from farm), true);*
4. *SELECT setval('public.f productivity_raw_seq', (select max(prod_point_id) from farm), true);*

b) Adicionar restrição de chave estrangeira

Os seguintes *scripts* foram executados para adicionar restrições de chave estrangeira:

1. Tabela: productivity_raw

```
ALTER TABLE productivity_raw
ADD CONSTRAINT prod_id_fk FOREIGN KEY (prod_id) REFERENCES
productivity (id)
ON UPDATE NO ACTION ON DELETE NO ACTION;
CREATE INDEX fki_prod_id_fk1
ON productivity_raw(prod_id);
```

2. Tabela: productivity

```
ALTER TABLE productivity
  ADD CONSTRAINT plot_id_fk FOREIGN KEY (plot_id) REFERENCES plot
(id) ON UPDATE NO ACTION ON DELETE NO ACTION;
CREATE INDEX fki_plot_id_fk2 ON productivity(plot_id);
```

3. Tabela: plot

```
ALTER TABLE plot
  ADD CONSTRAINT farm_id_fk FOREIGN KEY (farm_id) REFERENCES
farm (id)
  ON UPDATE NO ACTION ON DELETE NO ACTION;
CREATE INDEX fki_farm_id_fk1
  ON plot(farm_id);
```

c) **Eliminar coluna**

Os seguintes *scripts* foram executados para a refatoração “Eliminar colunas”:

1. Tabela: productivity_raw

```
ALTER TABLE productivity_raw DROP COLUMN plot_id;
ALTER TABLE productivity_raw DROP COLUMN farm_id;
ALTER TABLE productivity_raw DROP COLUMN prod_point_id;
```

2. Tabela: productivity

```
ALTER TABLE productivity DROP COLUMN prod_id;
ALTER TABLE productivity DROP COLUMN farm_id;
```

3. Tabela: plot

```
ALTER TABLE plot DROP COLUMN plot_id;
```

4. Tabela: farm

ALTER TABLE farm DROP COLUMN farm_id;

Figura 30– Tabelas após as refatorações da Atividade 8.

Fonte: Autor

d) Unir colunas

Os seguintes *scripts* foram executados para a refatoração “Unir Colunas”:

1. Tabela: productivity_raw: criar a nova coluna

```
SELECT AddGeometryColumn('productivity_raw','prod_geom',
4326,'POINT',2)
```

2. Tabela: productivity_raw: atualizar o valor da nova coluna

```
UPDATE productivity_raw
SET prod_geom=ST_SetSRID(ST_Point(longitude, latitude), 4326)
```

3. Tabela: productivity_raw: eliminar a coluna antiga

```
ALTER TABLE productivity_raw DROP COLUMN longitude;
ALTER TABLE productivity_raw DROP COLUMN latitude;
```

A Figura 31 mostra o resultado dos *scripts* 1,2 e 3 para a refatoração “Unir Colunas”.

Figura 31– Tabela productivity_raw após a refatoração “Unir Colunas”.

productivity_raw	
id: INTEGER	[PK]
prod_id: INTEGER	[FK]
prod_point_prod_drymat:	REAL
prod_point_moisture:	REAL
prod_point_plat_width:	REAL
prod_point_distance:	REAL
prod_point_geom:	OTHER
prod_point_mac_speed:	REAL
prod_point_date:	DATE
prod_point_direction:	REAL
prod_point_timelength:	REAL
prod_point_shift_x:	REAL
prod_point_shift_y:	REAL
prod_point_culture_flow:	REAL
prod_point_grain_temp:	REAL
prod_point_prod_moist:	REAL
prod_point_prodvola_moist:	REAL
prod_point_prodvola_drymat:	REAL
prod_point_productivity:	REAL
prod_point_round:	INTEGER
prod_geom:	geometry

Fonte: Autor

e) **Introduzir índice**

O seguinte *script* do PostGis foi executado para introduzir um índice:

```
CREATE INDEX prod_raw_geom_gist ON
    productivity_raw USING GIST (prod_geom)
```

6.1.9. Atividade 9 - Salvar o que foi executado no banco de dados

Todos os *scripts* mostrados na atividade anterior foram salvos em arquivos de *scripts* para execução posterior em vários ambientes do sistema, tal como o de produção.

6.1.10. **Atividade 10 - Executar um conjunto de consultas depois da refatoração**

Os seguintes *scripts* foram executados depois das refatorações:

- INSERT INTO PRODUCTIVITY_RAW
- SELECT ALL ROWS FROM PRODUCTIVITY_RAW

Estes *scripts* estão resumidos para não ter que descrever todas as colunas envolvidas nas tabelas, mas os *script* completos acrescentam somente as colunas omitidas nestes *scripts* e estão apresentados no Anexo B. Os *scripts* para seleção de dados consideraram a recuperação de dados específicos de produtividade, ou seja, de uma colheita específica (tabela *productivity*).

6.1.11. **Atividade 11 - Guardar o tempo de execução das consultas depois da refatoração**

Para cada *script* mostrado no item anterior, foi coletada uma série de dados para avaliar o impacto das refatorações. As tabelas a seguir apresentam uma amostra de 10 resultados para cada simulação. As Tabela 5 e 6 mostram os resultados para selecionar e inserir dados na tabela *productivity_raw*, após as refatorações:

- Introduzir chave de identificação
- Adicionar restrição de chave estrangeira
- Eliminar coluna

No caso do *script* de inclusão tem-se os seguintes tempos após estas refatorações:

Tabela 5 – Tempos gastos para inclusão de dados depois da refatoração.

INCLUSÃO DE DADOS NA TABELA PRODUCTIVITY_RAW DEPOIS DA REFATORAÇÃO		
	LINHAS	TEMPO (MS)
1	189730	6683,000
2	189730	8519,799
3	189730	7790,775
4	189730	7284,497
5	189730	6865,953
6	189730	8091,868
7	189730	7736,015
8	189730	7007,305
9	189730	6989,023
10	189730	7254,792
	Média	7422,303
	Desvio Padrão	591,617

Fonte: Autor

Tabela 6 – Tempos gastos para seleção de dados depois da refatoração.

SELEÇÃO DE TODAS AS LINHAS DA TABELA PRODUCTIVITY_RAW DEPOIS DA REFATORAÇÃO		
	LINHAS	TEMPO (MS)
1	189730	41808
2	189730	42258
3	189730	38195
4	189730	38915
5	189730	39137
6	189730	37505
7	189730	39618
8	189730	38867
9	189730	37580
10	189730	36756
	Média	39064
	Desvio Padrão	1789

Fonte: Autor

Os tempos para seleção de dados, após a refatoração “Unir Colunas”, estão apresentados na Tabela 7.

Tabela 7 – Tempos gastos para seleção de dados após a refatoração “Unir colunas”.

SELEÇÃO DE TODAS AS LINHAS DA TABELA PRODUCTIVITY_RAW APÓS MERGE DAS COLUNAS LATITUDE E LONGITUDE		
	LINHAS	TEMPO (MS)
1	187688	1101,856
2	187688	1002,680
3	187688	998,763
4	187688	1006,576
5	187688	1018,245
6	187688	1012,359
7	187688	1002,760
8	187688	999,438
9	187688	1000,135
10	187688	997,396
	Média	1014,021
	Desvio Padrão	31,554

Fonte: Autor

Os tempos para seleção de dados, após a refatoração “Introduzir Índice”, estão apresentados na Tabela 8.

Tabela 8 – Tempos gastos para seleção de dados após a refatoração introduzir o índice.

SELEÇÃO DE TODAS AS LINHAS DA TABELA PRODUCTIVITY_RAW COM ÍNDICE NA COLUNA PROD_GEOM		
	LINHAS	TEMPO (MS)
1	187688	1082,238
2	187688	1063,252
3	187688	1059,208
4	187688	1055,774
5	187688	1046,793
6	187688	1054,664
7	187688	1051,260
8	187688	1059,832
9	187688	1060,566
10	187688	1062,486
	Média	1059,607
	Desvio Padrão	9,474

Fonte: Autor

6.1.12. Atividade 12 - Mostrar o relatório das refatorações

Após todas as refatorações tem-se o modelo final do sistema apresentado na Figura 32.

Figura 32 - Modelo final após todas as refatorações

Fonte: Autor

Como citado na seção anterior, foram coletados os tempos de execução para inserir e selecionar dados antes e depois de cada refatoração. Abaixo são apresentados os gráficos comparativos para cada uma das refatorações.

- Introduzir chave de identificação;
- Adicionar restrição de chave estrangeira; e
- Eliminar coluna

As Figuras 33 e 34, apresentam graficamente, os tempos coletados para selecionar dados na tabela *productivity_raw* no modelo original (Figura 28) e no modelo após a

refatoração (Figura 30). Pode-se observar que, o tempo consumido foi de 21% menor para inserir e 23% mais curto para selecionar a maior tabela do banco de dados. Conforme as Figuras 33 e 34 o desempenho do banco de dados reformulado foi melhor do que o original, considerando-se os dados coletados.

Figura 33– Tempos coletados para selecionar dados antes e depois das refatorações

Fonte: Autor

Figura 34– Tempos coletados para inserir dados antes e depois das refatorações

Fonte: Autor

Após a refatoração de unir colunas a operação de seleção foi 20% melhor em relação ao tempo de consumo para selecionar os dados do que no modelo original conforme a Figura 35.

Figura 35– Tempos coletados para selecionar dados antes e depois da refatoração de “Unir colunas”

Fonte: Autor

A nova coluna, resultado da união, era uma candidata para ser indexada, mas quando a refatoração introduzir índice foi aplicada, o tempo gasto para selecionar os dados aumentou em 4,5% (Figura 36). Com esse desempenho, a refatoração adicionar índice foi considerada desnecessária

Figura 36– Tempos coletados para selecionar dados antes e depois de adicionar o índice

Fonte: Autor

6.2. CONSIDERAÇÕES FINAIS DESTE CAPÍTULO

O sistema desenvolvido para a área de agricultura que utiliza georreferenciamento, com grande volume de dados foi uma boa escolha para o estudo de caso. Permitiu coletar tempos importantes para avaliar o desempenho. Possibilitou analisar uma parte do sistema, um conjunto de quatro tabelas, mas bastante representativo para o comportamento do sistema. Por último, não havia nenhum trabalho anterior de revisão do modelo de dados desse sistema, que possibilitou executar um número considerável de refatorações.

O processo de refatoração de banco de dados foi executado passo a passo para que fosse verificada a validade, necessidade e a importância de todas as atividades. A verificação foi feita com sucesso e o processo não precisou de nenhum ajuste durante a sua execução.

O estudo de caso reforçou a importância da coleta de tempos antes e depois das refatorações. Uma das refatorações, “Introduzir Índice”, normalmente considerada a solução básica para melhorar o desempenho, mostrou-se prejudicial no caso de campos de tipos especiais: geométricos, como o campo **prod_geom**, que pode ser visto na Figura 31. A última atividade do processo de refatoração, quando se faz o relatório das refatorações e analisa os dados coletados, mostrou a desvantagem de criar um índice como o que foi proposto inicialmente. Com essa conclusão do relatório, uma recomendação importante é que, os administradores de banco de dados, devem rever o conjunto de refatorações antes de executar o processo nos outros ambientes (teste e produção).

7. CONCLUSÕES

O trabalho inicial do administrador de banco de dados é coletar os requisitos de alterações dos usuários, identificar os problemas de modelagem ou que podem influenciar o desempenho de consultas. Esse conjunto de alterações no esquema de banco de dados é a entrada do processo de refatoração que leva o esquema do banco de dados do modelo original até o final.

As tarefas do novo processo de refatoração foram comparadas com as tarefas do processo desenvolvido por Ambler, que é amplamente utilizado na literatura de refatoração de banco e dados. O novo processo resolveu deficiências do processo de Ambler e incluiu novas funcionalidades que contribuem para a integridade e consistência do banco de dados após uma refatoração.

Após o estudo de caso foi evidenciada a viabilidade da aplicação do novo processo de refatoração e confirmou-se a necessidade e a importância de todas as atividades.

7.1. EVOLUÇÃO DO TRABALHO

O desenvolvimento desta Tese teve como base o método apresentado na Figura 2, e abaixo são descritos alguns pontos relevantes para a evolução do trabalho:

- a) o objetivo foi estudar o problema de refatoração de banco de dados;
- b) a evolução de bancos de dados utilizando refatorações é algo recente e ainda há pouco material disponível. No início da pesquisa só havia como referências os trabalhos de Ambler e Sadalage (2006) e Domingues (2011). No decorrer da pesquisa foram encontrados outros autores que pesquisavam refatoração de banco de dados, muitos deles se basearam no trabalho de Ambler (Capítulo 3);
- c) a pesquisa bibliográfica revelou que os autores pesquisados tratavam somente uma refatoração. Em ambientes complexos, os bancos de dados sofrem constantes alterações, o que exige que as refatorações sejam feitas em conjunto, ou em um único processo;

- d) o processo base que a bibliografia dispunha era de Ambler. Esse processo foi cuidadosamente estudado, para apontar problemas críticos como o fato de executar uma refatoração por vez, e não ter nenhuma etapa de documentação e recuperação do modelo antigo, caso ocorram problemas;
- e) várias opções de notação e linguagens para representação de processos foram estudadas. A notação BPMN foi escolhida para a representação do processo desenvolvido, como mostrado no Capítulo 3; e
- f) os estudos iniciais foram feitos com exemplos de banco de dados desenvolvidos para os testes ou exemplos encontrados nas referências de banco de dados. Após essa validação inicial, foram feitos testes com um banco de dados real, como mostrado no Capítulo 6.

7.2. CONTRIBUIÇÕES DO TRABALHO

O novo processo de refatoração tem as seguintes contrições:

- a) apresenta um processo definido em BPMN para fazer refatoração em banco de dados;
- b) o processo tem atividades importantes que não eram colocadas no processo de refatoração amplamente utilizado na literatura de banco de dados, e que foi apresentado na figura 26, tais como: backup do esquema original, coleta de tempo de consultas antes e depois da refatoração, ordenação das entidades envolvidas na refatoração;
- c) considera as refatorações como um conjunto de tarefas e não algo isolado. esse conjunto é denominado no trabalho como uma grande alteração de banco de dados;
- d) validação do processo em um banco de dados com grande volume de dados e que possuía requisitos para realizar uma grande alteração; e
- e) as refatorações realizadas no estudo de caso apresentaram ganhos na ordem de 20% no desempenho das consultas do sistema.

7.3. SUGESTÕES PARA TRABALHOS FUTUROS

Durante o desenvolvimento desta tese, foram verificadas algumas possibilidades relevantes de estudos futuros decorrentes deste trabalho:

- a) automatizar o processo para facilitar a execução por usuários não especialistas em bancos de dados;
- b) desenvolver uma interface para o usuário executar o processo que possibilite, conforme a situação do banco de dados, sugerir refatorações;
- c) pesquisar novas alterações de bancos de dados que não estão catalogadas como refatoração na literatura de bancos de dados e colocá-las disponíveis para os usuários;
- d) desenvolver um algoritmo que, após escolhida uma refatoração, pesquise no esquema alvo todas as entidades candidatas para receber a refatoração; e
- e) pesquisar um algoritmo que forneça uma ordenação das entidades envolvidas na refatoração. no estudo de caso, a refatoração de adicionar restrição de chave estrangeira exigiu a ordenação das entidades para que seja possível definir as restrições.

7.4. CONSIDERAÇÕES FINAIS

O novo processo de refatoração acarreta vários benefícios em relação ao processo da Figura 2 e apresentou um tratamento para grandes alterações de banco de dados que não havia antes. O objetivo deste trabalho sempre foi expor de maneira prática e transparente a importância da refatoração de banco de dados. Isso é mais relevante quando se trata de banco de dados relacionais, tão necessários ainda, mas que com o tempo poderão ser suprimidos por outras tecnologias. Tratar de refatoração de banco de dados relacionais é muito mais que alterar tabelas, ou mudar relacionamentos, é garantir que os dados sobrevivam íntegros e confiáveis, mesmo que o sistema passe por muitas alterações.

REFERÊNCIAS BIBLIOGRÁFICAS

AALST, W. M. P. VAN DER. Business process management demystified: A tutorial on models, systems and standards for workflow management. **Lectures on concurrency and Petri nets**, p. 1–65, 2004.

ABNT. **NBR ISO 9000**. Rio de Janeiro: ABNT, 2005.

ABPMP. **CBOK - Business Process Management Common Body of Knowledge (BPM CBOK)**. Chigago: ABPMP, 2009.

AMBLER, S. **Agile modeling: effective practices for eXtreme programming and the unified process**. New York, NY: John Wiley & Sons, 2002.

AMBLER, S. W.; SADALAGE, P. J. **Refactoring databases: evolutionary database design**. New York: Addison-Wesley Professional, 2006.

BECK, K.; BEEDLE, M.; BENNEKUM, A. VAN; et al. Manifesto for Agile Software Development. Disponível em: <<http://www.agilemanifesto.org/>>. Acesso em: 7/1/2014.

BOEHM, A.; SEIPEL, D.; SICKMANN, A.; WETZKA, M. Squash: A Tool for Analyzing, Tuning and Refactoring Relational Database Applications. In: D. Seipel; M. Hanus; A. Wolf (Eds.); **Applications of Declarative Programming and Knowledge Management**, Lecture Notes in Computer Science.. v. 5437, p.82–98, 2009. Springer Berlin Heidelberg.

CAMPOS, A. L. N. **Modelagem de processos com BPMN**. Rio de Janeiro: Brasport, 2013.

CASANOVA, M. A.; TUCHERMAN, L.; FURTADO, A. L.; BRAGA, A. P. Optimization of relational schemas containing inclusion dependencies. In: Proceedings of the Fifteenth International Conference on Very Large Data Bases. **Anais...** p.317–325, 1989. Morgan Kaufmann.

CASANOVA, M. A.; TUCHERMAN, L.; LAENDER, A. H. F. Algorithms for designing and maintaining optimized relational representations of entity-relationship schemas. In: H. Kangassalo (Ed.); In: Proceedings of the 9th International Conference on Entity-Relationship Approach (ER'90), 8-10 October, 1990, Lausanne, Switzerland. **Anais...** p.361–374, 1990. ER Institute.

CASANOVA, M. A.; TUCHERMAN, L.; LAENDER, A. H. F. On the design and maintenance of optimized relational representations of entity-relationship schemas. **Data Knowl. Eng.**, v. 11, n. 1, p. 1–20, 1993.

CHACON, S. **Pro Git**. New York: Apress, 2009.

CHANG, S.-K.; DEUFEMIA, V.; POLESE, G.; VACCA, M. A logic framework to support database refactoring. In: R. Wagner; N. Revell; G. Pernul (Eds.); **Database and Expert Systems Applications**, Lecture Notes in Computer Science.. v. 4653, p.509–518, 2007. Springer Berlin Heidelberg.

CHEN, P. P. The entity-relationship model - toward a unified view of data. **ACM Transactions on Database Systems**, v. 1, n. 1, p. 9–36, 1976.

CLEVE, A.; BROGNEAUX, A.-F.; HAINAUT, J.-L. A conceptual approach to database applications evolution. In: Proceedings of the 29th international conference on Conceptual modeling. **Anais...**, ER'10.. p.132–145, 2010. Berlin, Heidelberg: Springer-Verlag.

COMYN-WATTIAU, I.; AKOKA, J.; LAMMARI, N. A framework for database evolution management. In: Workshop on Unanticipated Software Evolution. **Anais...** p.105–113, 2003.

COSTA, W. F.; SANTANA, F. S.; SARAIVA, A. M.; MOLIN, J. P. Banco de dados geográficos para a construção de serviços web para agricultura de precisão. In: VIII Congresso Brasileiro de Agroinformática. **Anais...**, 2011. Bento Gonçalves, Rio Grande do Sul, Brasil.

CURINO, C.; MOON, H. J.; ZANIOLO, C. Managing the History of Metadata in Support for DB Archiving and Schema Evolution. Fifth International Workshop on Evolution and Change in Data Management. **Anais...** p.78–88, 2008. Springer-Verlag.

CURINO, C.; MOON, H. J.; ZANIOLO, C. Automating Database Schema Evolution in Information System Upgrades. In: Proceedings of the 2Nd International Workshop on Hot Topics in Software Upgrades. **Anais...**, HotSWUp '09.. p.5:1–5:5, 2009. New York: ACM.

D'SOUSA, A.; BHATIA, S. Refactoring of a database. **International Journal of Computer Science and Information Security**, v. 6, n. 2, p. 307–315, 2009.

DAVENPORT, T. H. **Process Innovation: reengineering work through information technology**. Boston, MA: Harvard Business School Press, 1993.

DBDEPLOY. Database change management tool. Disponível em: <<http://dbdeploy.com/>>. Acesso em: 10/1/2014.

DECKER, M.; CHE, H.; OBERWEIS, A.; STÜRZEL, P.; VOGEL, M. Modeling Mobile Workflows with BPMN. In: 2010 Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR). **Anais...** p.272–279, 2010. IEEE.

DITTRICH, J.-P.; SALLES, M. A. V. iDM: A unified and versatile data model for personal dataspace management. VLDB. **Anais...** p.367–378, 2006.

DOMINGUES, H. H. **Replicação Assíncrona de banco de dados evolutivos**, 2011. Instituto de Matemática e Estatística, Universidade de São Paulo.

DOMINGUES, H. H.; KON, F.; FERREIRA, J. E. Replicação assíncrona em modelagem evolutiva de banco de dados. In: SBBD. **Anais...** p.121–135, 2009.

DOMINGUES, H. H.; KON, F.; FERREIRA, J. E. Asynchronous replication for evolutionary database development: a design for the experimental assessment of a novel approach. In: Proceedings of the 2011th Confederated international conference on On the move to meaningful internet systems - Volume Part II. **Anais...** , OTM'11.. p.818–825, 2011. Berlin, Heidelberg: Springer-Verlag. Disponível em: <<http://dl.acm.org/citation.cfm?id=2075764.2075804>>.

DOMINGUES, M. B. P.; ALMEIDA JÚNIOR, J. R. DE; COSTA, W. F.; SARAIVA, A. M. Database refactoring to increase/retrieve information from precision agriculture information system. In: EFITA-WCCA-CIGR Conference Sustainable Agriculture through ICT Innovation. **Anais...** , 2013a. Torino, Italy.

DOMINGUES, M. B. P.; ALMEIDA JÚNIOR, J. R. DE; COSTA, W. F.; SARAIVA, A. M. Refactoring database to improve queries performance. In: 21th Italian Symposium on Advanced Database Systems (SEBD 2013). **Anais...** , 2013b. Roccella Jonica, Italy.

DOMINGUES, M. B. P.; ALMEIDA JÚNIOR, J. R. DE; SILVA, J. R. Evolution of Databases Using Petri Nets. In: CBA - Congresso Brasileiro de Automática. **Anais...** , 2012. Campina Grande - PB.

DOMÍNGUEZ, E.; LLORET, J.; ZAPATA, M. An architecture for managing database evolution. In: A. Olivé; M. Yoshikawa; E. Yu (Eds.); **Advanced Conceptual Modeling Techniques**, Lecture Notes in Computer Science.. v. 2784, p.63–74, 2003. Springer Berlin / Heidelberg.

DONG, X.; HALEVY, A. Y. A platform for personal information management and integration. In: CIDR. **Anais...** p.119–130, 2005.

DONG, X. L.; HALEVY, A.; YU, C. Data integration with uncertainty. **The VLDB Journal**, v. 18, n. 2, p. 469–500, 2009. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

DUMAS, M.; AALST, W. M. VAN DER; HOFSTEDE, A. H. TER. Process-aware information systems: bridging people and software through process technology. ,2005. John Wiley & Sons, Inc.

FOWLER, M.; BECK, K.; BRANT, J.; OPDYKE, W.; ROBERTS, D. **Refactoring: Improving the Design of Existing Code**. New York: Addison-Wesley Professional, 1999.

GHALLAB, M.; NAU, D.; TRAVERSO, P. **Automated Planning: theory & Practice (The Morgan Kaufmann Series in Artificial Intelligence)**. San Francisco: Morgan Kaufmann, 2004.

GIAGLIS, G. M. A Taxonomy of Business Process Modeling and Information Systems Modeling Techniques. **International Journal of Flexible Manufacturing Systems**, v. 13, p. 209–228, 2001. Disponível em: <<http://dx.doi.org/10.1023/A:1011139719773>>

GIL, A. C. **Métodos e técnicas de pesquisa social**. 6th ed. São Paulo: Atlas, 2008.

GUEDES, G. T. A. **UML 2 - Uma Abordagem Prática**. São Paulo, SP, Brasil: Novatec Editora, 2009.

HEUSER, C. A.; PERES, E. M.; RICHTER, G. Towards a complete conceptual model: Petri nets and entity-relationship diagrams. **Information Systems**, v. 18, n. 5, p. 275–298, 1993. Oxford, UK, UK: Elsevier Science Ltd.

HOWE, B.; MAIER, D.; RAYNER, N.; RUCKER, J. Quarrying dataspace: Schemaless profiling of unfamiliar information sources. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering Workshop. **Anais...** ,

ICDEW '08.. p.270–277, 2008. Washington, DC, USA: IEEE Computer Society.
Disponível em: <<http://dx.doi.org/10.1109/ICDEW.2008.4498331>>

LAENDER, A. H. F.; CASANOVA, M. A.; CARVALHO, A. P. DE; RIDOLFI, L. F. G. G. M. An analysis of SQL integrity constraints from an entity-relationship model perspective. **Information Systems**, v. 19, n. 4, p. 331–358, 1994. Oxford, UK, UK: Elsevier Science Ltd.

LARMAN, C.; BASILI, V. R. Iterative and incremental developments. a brief history. **Computer**, v. 36, n. 6, p. 47–56, 2003.

LESER, U.; NAUMANN, F. (Almost) hands-off information integration for the life sciences. In: CIDR. **Anais...** p.131–143, 2005. Disponível em: <<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.69.962>>. Acesso em: 10/1/2014.

LIQUIBASE. Database change management. Disponível em: <<http://www.liquibase.org>>. Acesso em: 10/1/2014.

LIU, C.-T.; CHRYSANTHIS, P.; CHANG, S.-K. Database schema evolution through the specification and maintenance of changes on entities and relationships. In: P. Loucopoulos (Ed.); **Entity-Relationship Approach — ER '94 Business Modelling and Re-Engineering**. p.132–151, 1994. Springer Berlin Heidelberg.

LIU, J.; DONG, X.; HALEVY, A. Y. Answering Structured Queries on Unstructured Data. In: WebDB. **Anais...** , 2006.

MAKAROV, A. **Yii Application Development Cookbook**. 2ª ed. United Kingdom: Packt Publishing, 2013.

MCCARTHY, J. Programs with Common Sense. Semantic Information Processing. **Anais...** p.403–418, 1968. MIT Press.

MICHAEL HARTL. **Ruby on Rails Tutorial: Learn Web Development with Rails**. 2ª ed. New York: Addison-Wesley Professional, 2012.

MILI, H.; TREMBLAY, G.; JAOUDE, G. B.; et al. Business process modeling languages. **ACM Computing Surveys**, v. 43, n. 1, p. 1–56, 2010. ACM. Disponível em: <<http://dl.acm.org/citation.cfm?id=1824795.1824799>>. Acesso em: 7/2/2014.

MURAKAMI, E.; SARAIVA, A. M.; RIBEIRO JUNIOR, L. C. M.; et al. An infrastructure for the development of distributed service-oriented information systems for precision

agriculture. **Computers and Electronics in Agriculture**, v. 58, n. 1, p. 37–48, 2007. Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V.

OERTLY, F.; SCHILLER, G. Evolutionary Database Design. In: Proceedings of the Fifth International Conference on Data Engineering, February 6-10, 1989, Los Angeles, California, USA. **Anais...** p.618–624, 1989. IEEE Computer Society.

OLIVEIRA, S. L. DE. **Tratado de metodologia científica: projetos de pesquisa, TGI, TCC, monografias, dissertações e teses**. São Paulo: Pioneira Thomson Learning, 2012.

OMG. Business Process Model and Notation (BPMN) Version 2.0. OMG Document Number: formal/2011-01-03. Disponível em: <<http://www.omg.org/spec/BPMN/2.0/PDF>>. Acesso em: 18/3/2014.

OSGEO. PostGIS. Disponível em: <<http://postgis.net/stuff/postgis-2.2.0dev-br.pdf>>. Acesso em: 3/3/2013.

PAVANINI JÚNIOR, O.; SCUCUGLIA, R. **Mapeamento e Gestão por Processos - BPM (Business Process Management). Gestão orientada à entrega por meio de objetos**. São Paulo: M. Books do Brasil, 2011.

PETERS, R. J.; ÖZSU, M. T. An axiomatic model of dynamic schema evolution in objectbase systems. **ACM Transactions on Database Systems**, v. 22, n. 1, p. 75–114, 1997. New York, NY, USA: ACM.

PHALP, K.; SHEPPERD, M. Quantitative analysis of static models of processes. **Journal of Systems and Software**, v. 52, n. 2-3, p. 105–112, 2000. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0164121299001363>>. Acesso em: 2/3/2014.

PRESSMAN, R. S. **Software engineering: a practitioner's approach**. 7th ed. New York: McGraw-Hill, 2010.

RADEMAKERS, T. **Activiti in Action: Executable business processes in BPMN 2.0**. Shelter Island, NY: Manning Publications, 2012.

REITER, R. On Formalizing Database Updates: Preliminary Report. In: EDBT. **Anais...** p.10–20, 1992.

REITER, R. On specifying database updates. **The Journal of Logic Programming**, v. 25, n. 1, p. 53–91, 1995.

RODDICK, J. F. A survey of schema versioning issues for database systems. **Information and Software Technology**, v. 37, n. 7, p. 383–393, 1995. Disponível em: <<http://www.sciencedirect.com/science/article/pii/095058499591494K>>

ROONEY, G.; BERLIN, D. **Practical Subversion**. Apress, 2006.

SARMA, A. DAS; DONG, X.; HALEVY, A. Bootstrapping pay-as-you-go data integration systems. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. **Anais...**, SIGMOD '08.. p.861–874, 2008. New York, NY, USA: ACM. Disponível em: <<http://doi.acm.org/10.1145/1376616.1376702>>

SGANDERLA, K. Um guia para iniciar estudos em BPMN. Disponível em: <<http://blog.iprocess.com.br/2012/11/um-guia-para-iniciar-estudos-em-bpmn-i-atividades-e-sequencia/>>. Acesso em: 24/2/2013.

SHARP, A.; MCDERMOTT, P. **Workflow Modeling: Tools for Process Improvement and Application Development**. 1st ed. Norwood, MA, USA: Artech House, Inc., 2001.

SILVA, A. C. L. BPMN 2.0 - Notação e Modelo de Processo de Negócio (Pôster). Disponível em: <<http://www.itposter.net/itPosters/bpmn/bpmn.htm>>. Acesso em: 18/3/2014.

SILVA, A. S. DA; LAENDER, A. H. F.; CASANOVA, M. A. An Approach to Maintaining Optimized Relational Representations of Entity-Relationship Schemas. In: Proceedings of the 15th International Conference on Conceptual Modeling. **Anais...**, ER '96.. p.292–308, 1996. London, UK, UK: Springer-Verlag.

SILVA, J. R. Applying Petri nets to requirements validation. IFAC Symposium on Information Control Problems in Manufacturing S. **Anais...** v. 1, p.508–517, 2004. Salvador. Disponível em: <http://www.abcm.org.br/symposiumSeries/SSM_Vol1/Section_IV_Discrete_Event_Dynamic_Systems/SSM_IV_01.pdf>.

SMITH, G. **PostgreSQL 9.0 High Performance**. United Kingdom: Packt Publishing, 2010.

VALLE, R.; OLIVEIRA, S. B. **Análise e Modelagem de Processos de Negócio: Foco da Notação BPMN (Business Process Modeling Notation)**. 1. ed. ed. São Paulo: ATLAS, 2013.

VAQUERO, T. S.; ROMERO, V.; TONIDANDEL, F.; SILVA, J. R. itSIMPLE 2.0: An Integrated Tool for Designing Planning Domains. In: ICAPS. **Anais...** p.336–343, 2007.

WFMC. **Workflow Management Coalition Terminology & Glossary (Document Number WFMC-TC-1011. Document Status - Issue 3.0)**. United Kingdom: Workflow Management Coalition Specification, 1999.

WHITE, S. A.; MIERS, D. **BPMN Modeling and Reference Guide**. Lighthouse Point, FL USA: Future Strategies Inc, 2008.

YIN, R. K. **Estudo de Caso - Planejamento e Métodos**. 4th ed. Porto Alegre, 2010.

APÊNDICE

APÊNDICE A – SCRIPTS PARA SELECIONAR E INSERIR DADOS (ATIVIDADE 6 - SEÇÃO 6.1.6)

- a) Escolha do esquema do banco de dados para ser usado:

```
SET SCHEMA 'estudo';
```

- b) *Script* para selecionar todas as linhas da tabela productivity_raw para dados de produtividade com id=1:

```
SELECT prod_point_id, prod_id, p.plot_id, pw.farm_id,
       prod_point_prod_drymat,
       prod_point_moisture, prod_point_plat_width, prod_point_distance,
       prod_point_geom, prod_point_mac_speed, prod_point_date,
       prod_point_direction, prod_point_timelength, prod_point_shift_x,
       prod_point_shift_y, prod_point_culture_flow, prod_point_grain_temp,
       prod_point_prod_moist, prod_point_prodvolt_moist,
       prod_point_prodvolt_drymat, prod_point_productivity,
       prod_point_round, longitude, latitude
FROM   estudo.productivity_raw
WHERE  prod_id = 1
```

- c) Exemplo de *script* para inserir uma linha na tabela productivity_raw

```
INSERT INTO productivity_raw (prod_point_id, prod_id, plot_id, farm_id,
prod_point_prod_drymat, prod_point_moisture, prod_point_plat_width,
prod_point_distance, prod_point_geom, prod_point_mac_speed,
prod_point_date, prod_point_direction, prod_point_timelength,
prod_point_shift_x, prod_point_shift_y, prod_point_culture_flow,
prod_point_grain_temp, prod_point_prod_moist, prod_point_prodvolt_moist,
prod_point_prodvolt_drymat, prod_point_productivity, prod_point_round,
longitude, latitude) VALUES (50245, 11, 1, 1, 1.52400005, 15.5, 10, NULL,
'0101000020E610000058B98233993349C0AD2DE175A2BD38C0', NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
2.91100001, NULL, -50.4031128299999978, -24.7407602000000004);
```


APÊNDICE B – SCRIPTS PARA SELECIONAR E INSERIR DADOS (ATIVIDADE 10 - SEÇÃO 6.1.10)

- d) Escolha do esquema do banco de dados para ser usado:

```
SET SCHEMA 'estudo';
```

- e) *Script* para selecionar todas as linhas da tabela productivity_raw antes da refatoração “Merge Columns” para dados de produtividade com id=1:

```
SELECT id, prod_id, prod_point_prod_drymat,  
       prod_point_moisture, prod_point_plat_width, prod_point_distance,  
       prod_point_geom, prod_point_mac_speed, prod_point_date,  
       prod_point_direction, prod_point_timelength, prod_point_shift_x,  
       prod_point_shift_y, prod_point_culture_flow, prod_point_grain_temp,  
       prod_point_prod_moist, prod_point_prodvол_moist,  
       prod_point_prodvол_drymat, prod_point_productivity,  
       prod_point_round, longitude, latitude  
FROM productivity_raw  
WHERE id = 1
```

- f) Exemplo de *script* para inserir uma linha na tabela productivity_raw antes da refatoração “Merge Columns”:

```
INSERT INTO productivity_raw (prod_id, prod_point_prod_drymat,
prod_point_moisture, prod_point_plat_width, prod_point_distance,
prod_point_geom, prod_point_mac_speed, prod_point_date,
prod_point_direction, prod_point_timelength, prod_point_shift_x,
prod_point_shift_y, prod_point_culture_flow, prod_point_grain_temp,
prod_point_prod_moist, prod_point_prodvola_moist,
prod_point_prodvola_drymat, prod_point_productivity, prod_point_round,
longitude, latitude) VALUES (50245, 11, 1, 1, 1.52400005, 15.5, 10, NULL,
'0101000020E610000058B98233993349C0AD2DE175A2BD38C0', NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
2.91100001, NULL, -50.4031128299999978, -24.7407602000000004);
```

- g) *Script* para adicionar coluna geométrica na tabela productivity_raw

```
SELECT AddGeometryColumn(estudo,'productivity_raw',
'prod_geom',4326,'POINT',2)
```

```
UPDATE e.productivity_raw SET
prod_geom=ST_SetSRID(ST_Point(longitude, latitude), 4326)
```

- h) *Script* para selecionar todas as linhas da tabela productivity_raw depois da refatoração “Merge Columns” para dados de produtividade com id=1:

```
SELECT id, prod_id, prod_point_prod_drymat,
       prod_point_moisture, prod_point_plat_width, prod_point_distance,
       prod_point_geom, prod_point_mac_speed, prod_point_date,
       prod_point_direction, prod_point_timelength, prod_point_shift_x,
       prod_point_shift_y, prod_point_culture_flow, prod_point_grain_temp,
       prod_point_prod_moist,
       prod_point_prodvolum_moist, prod_point_prodvolum_drymat,
       prod_point_productivity, prod_point_round, prod_geom

FROM productivity_raw
WHERE productivity_raw.prod_id=14 and ST_Contains(prod_geom,
          ST_SetSRID(ST_Point(longitude, latitude), 4326) )
```

- i) *Script* para criar o índice na tabela productivity_raw

```
CREATE index prod_refact_plot_geom_gist on estudo.productivity_raw using
GIST (prod_geom)
```