FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

JAIR MORATORE JUNIOR

ESTUDO DE MÉTODOS E PROCESSOS DE REFATORAÇÃO DE BANCO DE DADOS

MARÍLIA
2015

JAIR MORATORE JUNIOR

ESTUDO DE MÉTODOS E PROCESSOS DE REFATORAÇÃO DE BANCO DE DADOS

Trabalho de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação da Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador:
Prof. Dr. Elvis Fusco

MARÍLIA
2015

Junior, Moratore, Jair.

Estudo de Métodos e Processos de Refatoração de Banco de Dados /Prof. Dr. Elvis Fusco. Marília, SP: [s.n.], 2015.

121 folhas

Monografia (Bacharelado em Sistemas de Informação): Centro Universitário Eurípides de Marília.

1. Modelagem de Dados Conceitual Lógica e Normalização
2. Refatoração
3. Refatoração de Banco de Dados
4. Categorias de Refatoração em Banco de Dados
5. O processo de Refatoração em Banco de Dados
6. Estudo Comparativo de Técnicas de Refatoração de Banco de Dados
7. Casos de Uso

CDD: 005.2

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA - UNIVEM
MANTIDO PELA FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Jair Moratore Junior

Título:Refatoração de Banco de Dados.

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Sistemas de Informação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Sistemas de Informação.

Nota: 9,5 (nove e meio)

Orientador: Elvis Fusco

1º.Examinador: Emerson Alberto Marconato

2º.Examinador: Leonardo Castro Botega

Marília, 30 de novembro de 2015.

DEDICATÓRIA

Dedico este trabalho primeiramente a Deus pelo dom da vida, a minha Mãe por todo o apoio e suporte necessário para a conclusão desta jornada, ao meu Pai (Em memória), aos meus amigos e familiares pela paciência e compreensão.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus por ter permitido que eu chegasse ao fim de mais esta jornada, por ter me concedido forças e paciência não só ao longo dos quatro anos de curso, mas também durante a vida, principalmente nos momentos mais difíceis.

Agradeço a minha ***mãe Maria Aparecida Ribeiro Moratore em especial*** *por toda ajuda incentivo e paciência nos momentos difíceis deste caminhar. A ela que esteve sempre colaborando para o meu desenvolvimento profissional e pessoal, e por estar sempre comigo em todos os momentos da minha existência te amo, e ao meu* ***Pai em Memória.***

Agradeço todos os meus familiares pela compreensão e colaboração para a conclusão deste curso.

Agradeço a todos os meus amigos pela paciência e amizade durante esses anos.

Agradeço a todos os amigos de Faculdade pela ajuda e compreensão em diversos momentos.

Agradeço a todos os professores do curso pelo aprendizado nas disciplinas e pela evolução profissional.

Agradeço em Especial à professora ***Giulliana Marega Marques*** *e o Professor* ***Ricardo Sabatine*** *pelo apoio nos momentos difíceis, e por toda ajuda necessária para concluir este projeto.*

Agradeço em especial a ***Bruna Sato*** *por tudo, e por ser especial em minha vida e por ser uma princesa, apesar da distância te amo.*

Agradeço a Professora de português, Maria de Lourdes Barros pela ajuda e correção da monografia.

Agradeço ao meu Orientador Professor Dr. ***Elvis Fusco*** *pela paciência, dedicação e orientação durante este projeto.*

RESUMO

O presente trabalho apresenta técnicas de refatoração em banco de dados relacional, divididas em seis categorias que são: estrutural, qualidade de dados, integridade referencial, arquitetural, método e transformações-não-refatoração, que visam corrigir erros e anomalias que surgem com o passar do tempo, devido a imperfeições e falhas nos esquemas conceituais do banco, que a primeiro momento foram passados despercebidos, mas que acarretam duplicações de informações, inconsistências, entre outros problemas, deixando o banco menos confiável e suas informações cada vez mais difíceis de entender. Propõe nessa pesquisa a aplicação das técnicas de refatoração de banco de banco de dados de quatro das seis categorias que serão: estrutural, qualidade de dados, integridade referencial e arquitetural em casos de uso (exemplos), que tende a minimizar erros e facilitar o entendimento das informações contidas no banco, assim facilitar consultas, manter a semântica e melhorar sua estrutura, conservando suas funcionalidades originais sem adicionar novas funcionalidades, melhorando os esquemas para projetos futuros em banco de dados.

Palavras Chaves: Banco de dados, Refatoração, Refatoração de Banco de dados.

ABSTRACT

This paper presents refactoring techniques in relational database, divided into six categories which are: structural, data quality, referential integrity, architectural, method and transformations-non-refactoring, aimed at correcting errors and anomalies that arise over time due to imperfections and flaws in the conceptual schemes of the bank, which in the first instance were passed unnoticed, but which carry information of duplications, inconsistencies, among other problems, leaving the less reliable bank and its increasingly difficult to-understand information. Proposes in this research the application of the technical database refactoring four of the six categories are: structural, data quality, referential integrity and architectural in use cases (examples), which tends to minimize errors and facilitate understanding the information contained in the database, thus facilitating consultations, keeping the semantics and improve its structure, preserving its original features without adding new features, improving schemes for future projects in the database.

Keywords: Database, Refactoring, Database Refactoring.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUP	Agile Unified Process
BD	Banco de Dados
BPMN	Business Process Model and Notation
CRUD	Create, Read, Update, Delete
CRM	Customer Relationship Management
DBA	Database Administrator
DER	Diagrama-Entidade-Relacionamento
DDL	Data Definition Language
DML	Data Manipulation Language
E-R	Entidade Relacionamento
FN	Forma Normal
M-ER	Modelo-Entidade- Relacionamento
RUP	Rational Unified Process
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
TI	Tecnologia da Informação
UML	Unified Modeling Language
XP	Extreme Programming

SUMÁRIO

INTRODUÇÃO

Hoje, cada vez mais se utilizam bancos de dados para armazenamento de informações em processos de gestão e tomada de decisão. Para ser destaque no mercado competitivo, ter muito mais do que apenas informações básicas sobre clientes e fornecedores e também para se tornar um diferencial, mas quanto mais dados ser administrados e armazenados nos bancos, mais cuidados com a integridade e confiabilidade devem ser tomados, não apenas com as informações contidas no banco, mas também na estrutura e no desempenho do banco de dados que contém essas informações.

O banco de dados é um bem vital para qualquer organização, sendo a mesma de pequeno, médio ou principalmente de grande porte, nesse ambiente todos os dados e informações não apenas da organização, mas também de pesquisas de mercado e faturamento estão armazenadas no banco de dados. Todas as organizações de modo geral se preocupam com a segurança de suas respectivas informações, sem levar em conta que uma boa modelagem conceitual no esquema lógico do banco é um modo de segurança preventiva para o futuro banco de dados que estará funcionando dentro da organização.

Um projeto de banco de dados contém mais do que futuras tabelas e relacionamentos entre elas, contém cenário, modelagem lógica, modelagem física, tudo para que o banco de dados a ser instalado não contenha erros como: redundâncias de informações, duplicações, dados isolados, anomalias, dentre outros, imperfeições que de algum modo afetam a confiabilidade, semântica e integridade do banco de dados e de suas informações.

As técnicas de refatoração podem ajudar a minimizar esses erros, que por algum motivo passam despercebidos, pode facilitar nas consultas a dados, e melhorar os esquemas conceituais do banco, deixar o projeto mais confiável mantendo a semântica e integridade não apenas dos dados, mas também do banco de dados em questão.

Em conjunto com as técnicas de refatoração do banco de dados, a engenharia reversa também pode ajudar na minimização de erros nos esquemas conceituais do mesmo, as técnicas de refatoração em banco de dados são divididas em 6 categorias que são: refatoração estrutural, refatoração de qualidade de dados, refatoração de integridade referencial, refatoração arquitetural, refatoração de método, e transformações não refatoração; cada

técnica tem a sua utilidade na refatoração, todas elas descritas por Ambler e Sadalage (2006).

O processo de refatoração em banco de dados difere-se da refatoração de código, além de manter a estrutura interna, deve-se também manter a integridade das informações e as funcionalidades do banco, caso contrário causa um problema ou impedimento na refatoração, o acoplamento, que é a medida de dependência entre dois elementos, quanto mais alto o acoplamento mais difícil se torna a refatoração.

Motivação e Justificativa

Com o grande volume de dados armazenados pelas corporações, ter um banco de dados confiável e sem redundâncias é fundamental. Quanto melhor elaborado o projeto do banco, menos retomada de trabalho será feito, quanto mais minucioso o levantamento de requisitos, menores são as chances de ter custos com manutenção.

A refatoração tende a melhorar os esquemas do banco a partir do modelo lógico, onde os erros são tratados com menor frequência, esta pesquisa toma como premissa, a utilização de banco de dados relacionais.

Objetivo Geral

Esta pesquisa tem como objetivo estudar e aplicar as técnicas de refatoração em banco de dados relacional para o melhor aproveitamento dos dados no banco, corrigindo erros e redundâncias a partir do modelo lógico do banco de dados.

Objetivos Específicos

- Apresentar conceitos e tipos de refatoração em banco de dados;
- Aplicar os conceitos e tipos de refatoração apresentados;
- Realizar testes com as técnicas de refatoração;
- Documentar os testes realizados no presente trabalho;
- Deixar uma revisão da literatura sobre o tema para trabalhos e pesquisas futuras.

Metodologia

- Revisão bibliográfica sobre modelagem de dados, refatoração, refatoração de banco de dados e metodologias ágeis;
- Verificação da aplicação dos conceitos de refatoração e metodologias ágeis no processo de refatoração de banco de dados;
- Utilização de um banco de dados legado para aplicação das técnicas de refatoração de banco de dados selecionadas.

Trabalhos Correlatos

Outros trabalhos refletiram sobre o processo de refatoração em banco de dados como a Tese de doutorado de Domingues (2014) que utilizou a própria pesquisa e programou-a sobre as categorias de refatoração em banco de dados e em suas respectivas técnicas, no qual se utilizam para detalhar e contextualizar a evolução de banco de dados e um novo processo de refatoração dos mesmos, apresentando problemas no processo de refatoração original proposto por Scott W Ambler (2006).

Desta forma, Domingues (2011) abordou o tema de replicação assíncrona em banco de dados evolutivos, e dividiu refatoração em banco de dados em quatro categorias de refatoração, que são: estrutural, qualidade de dados, integridade referencial, e, arquitetural, com suas respectivas refatorações como parte do projeto enfatizado na seção 2.2 do capítulo 2, e também no capítulo 3 com os exemplos e contextualização sobre as refatorações, e, mostrou que em algumas refatorações de banco de dados encontram-se replicação de dados para permanecerem atualizados.

Em seu trabalho Domingues (2014) abordou o tema e apontou um novo processo para refatoração de banco de dados, incluindo problemas no processo de refatoração de banco de dados descritos por Ambler (2006) e também discorreu sobre quatro categorias de refatoração em banco de dados que são: estrutural, qualidade de dados, integridade referencial, e, arquitetural; a autora em questão abordou sobre replicação de dados em algumas técnicas de refatoração para manter os dados atualizados, e também sobre o BPMN (*Business Process Model and Notation*).

Organização do Trabalho

O presente trabalho está fundamentado em uma introdução, sete capítulos e conclusão, nesta introdução são abordados a contextualização e os objetivos do trabalho.

No primeiro capítulo é abordado o assunto sobre Modelagem de Dados Conceitual, Lógica e Normalização, no segundo capítulo está contextualizada a Refatoração, no terceiro capítulo apresenta-se a Refatoração de Banco de Dados, já no quarto capítulo estão contextualizados os exemplos sobre as técnicas de refatoração de banco de dados mais propostos nos trabalhos correlatos, no quinto capítulo é abordado O processo de refatoração de banco de dados, e, no sexto capítulo demostra-se um Estudo comparativo entre as técnicas de refatoração de banco de dados e, no sétimo capítulo é contextualizados e exemplificados casos de uso (exemplos) sobre as técnicas refatoração de banco de dados, e a conclusão.

1 MODELAGEM DE DADOS CONCEITUAL LÓGICA E NORMALIZAÇÃO

Neste capítulo serão apresentados os conceitos sobre modelagem de dados conceitual, lógica e normalização em um projeto de banco de dados. A modelagem de dados é importante e indispensável para que se compreendam os problemas específicos de um cenário, para o desenvolvimento de um banco de dados adequado ao negócio, que o atenda tanto logicamente quanto no domínio real. A modelagem de dados é constituída de etapas que são: modelagem conceitual, modelagem lógica e modelagem física, no modelo de dados para um projeto em banco de dados contêm graus de abstração, entidades, atributos, relacionamentos, diagramas E-R (entidade relacionamento), cardinalidade mínimas e máximas.

A normalização em banco de dados é importante para detecção e correção de dados com redundância e duplicações, a normalização é denominada basicamente de três formas que são: 1° forma normal, 2° forma normal, e 3° forma normal, em algumas literaturas pode-se encontrar a 4° forma normal e também a 5° forma normal, será mostrado o conceito sobre normalização com as três formas básicas, e também exemplos sobre modelagem de dados nas seções seguintes.

Segundo Chen (1990) um banco de dados é uma coleção de registros de tipos diferentes, os registros em um banco de dados são interligados, de forma que itens de dados relevantes em registros diferentes podem ser recuperados sem dificuldade.

Mas, Yong (1990) afirma que um banco de dados é uma base de dados constituída de um conjunto de arquivos relacionados entre si.

> "Chen (1990, p.2) aponta que um banco de dados, é constituído de coleção de registros com diversos tipos, com algum relacionamento entre os mesmos, desta forma o que se tem mais importância é extraído em diferentes registros interligados no banco de dados".

1.1 Modelo de Dados

É uma representação simples, normalmente gráfica, de estrutura de dados reais mais complexas, sendo esta um modelo de uma abstração de um objeto ou evento real de maior complexidade. Sua função é auxiliar na compreensão das complexidades do ambiente real. Pela visão do BD, nada mais é que um modelo que representa estruturas de dados com suas características, relações, restrições, transformações e outros elementos que tenham finalidade

de dar suporte ao problema específico de um domínio (MACEDO, 2011).

Yong (1990) enfatiza a importância do modelo de dados principalmente:

- Na possibilidade de fornecer informações sobre todos os dados envolvidos e de forma global, dando oportunidade ao usuário de conhecer a semântica do banco de dados.
- A possibilidade adicional de conhecer a estrutura das informações das condições de restrição quanto à integridade, de segurança e privacidade do banco de dados.

Yong (1990) conceitua que em um modelo de dados devem-se considerar alguns critérios para sua implantação e construção que são:

- O modelo deve ser simples, isto é, conter o menor número possível de elementos referentes á estrutura de dados.
- O modelo deve ser exibido com boa visualização, por meio de métodos e técnicas descritivas facilitando a compreensão do modelo.
- O modelo deve conter elementos que sejam contrapartidas diretas dos conceitos básicos do mundo real percebido.
- O modelo deve conter elementos tanto de estrutura como de substancia, que sirvam para a construção fácil de esquemas.
- O modelo deve conter o mínimo possível de atributos típicos da fase de implantação, isto é, seja independente de SGBD, no entanto não deve ser muito distante da possível estrutura física, a fim de diminuir a complexidade de transformação do modelo lógico ao físico.
- O modelo deve conter indicações para a formação de partições de dados, com finalidades de controle de acessos ao banco de dados, assim como maior eficiência de pesquisa online e controle de redundâncias.

Yong (1990) afirma que um modelo de dados é constituído de duas fases, que são: Conceituação do modelo, e implantação do modelo de dados, e que nas mesmas estão contidos quatro etapas que são: percepção do mundo real, representação do mundo real, através do modelo conceitual, implantação lógica, e implantação física que serão representadas a seguir na figura 1 que ilustra as respectivas fases e etapas das mesmas descritas por Yong (1990).

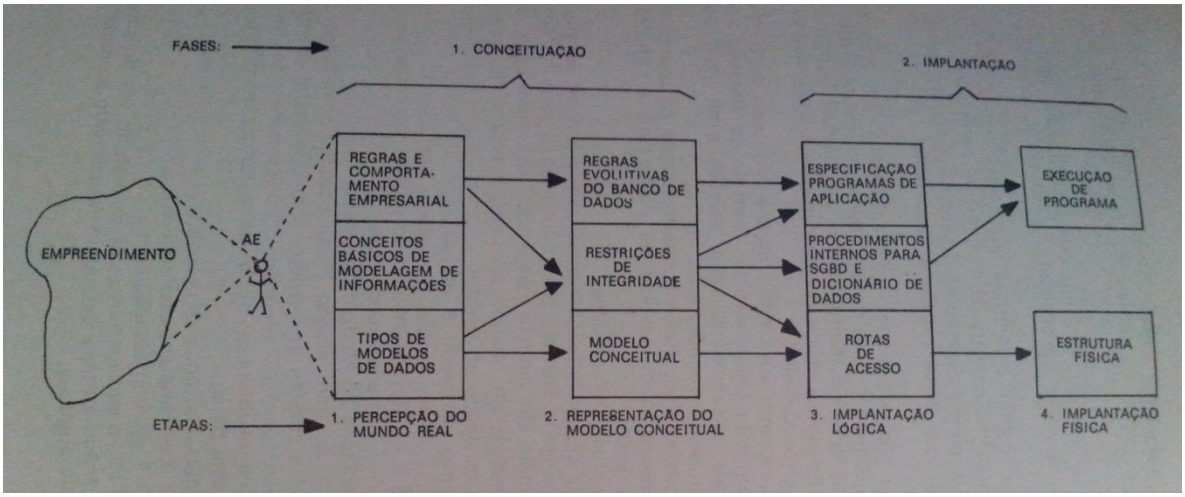

Figura 1. Fases na conceituação e implantação dos modelos de dados (processo dedutivo)

Fonte: YONG, (1990).

1.2 Percepção do mundo real

Segundo Yong (1990) percepção do mundo real é um processo de análise na observação das regras e comportamentos dos elementos básicos envolvidos nos diversos aspectos organizacionais, administrativos e operacionais do empreendimento. A descrição de tais elementos envolve a adoção de determinados conceitos básicos como entidades, associações, tipos, e demais. Isto acontece durante a caracterização particular do mundo real em questão, dentro do processo gradual de modelagem de informações:

1 Percepção do mundo real composto de entidades e suas associações.

2 Indução de propriedades das entidades e associações.

3 Abstração na classificação das entidades e associações por tipo.

Os conceitos básicos mais utilizados para descrever o mundo real de forma completa são: entidade, propriedades e associações entre entidades (YONG, 1990).

- Entidade

Uma entidade é tudo aquilo que um indivíduo ou um grupo de indivíduos (por

exemplo: uma organização, um grupo de cientistas, etc.) vê como um todo. Ela tem existência própria e é caracterizada por um conjunto de propriedades quantitativas e qualitativas e um comportamento permanente (YONG, 1990).

Uma entidade é uma "coisa" que pode ser distintamente identificada. As entidades podem ser classificadas em diferentes tipos, tais como funcionário e acionista. No diagrama E-R, um tipo de entidade é representado por um retângulo (CHEN, 1990).

- Associação

Uma associação é um conjunto de duas ou mais entidades, onde cada entidade desempenha um determinado papel. Uma associação pode possuir diferentes propriedades e sua existência depende da existência das entidades consideradas. Se a associação envolve duas entidades, tem-se uma associação binária; se envolve três entidades, então tem-se uma associação ternária, assim por diante (YONG, 1990).

- Propriedade

Propriedade é uma característica importante de uma entidade ou associação, não tem significado por si mesma; adquire significado quando relacionado a uma entidade ou associação.

Um valor de propriedade que pertence a uma entidade ou associação é uma qualidade assinalada a ela própria. O valor realiza a propriedade. Os três conceitos básicos delineados são tomados como axiomas para a descrição completa do mundo real (YONG, 1990).

- Tipos de Entidades

Um tipo de entidade é uma abstração representada por uma classe ou conjunto de entidades similares que possuem as mesmas propriedades.

- Tipos de Associações

Um tipo de associação é uma abstração representada por uma coleção de associações envolvendo entidades pertencentes aos correspondentes tipos de entidades tendo, se necessários, as mesmas propriedades (YONG, 1990).

- Tipo de propriedade

É um conjunto de valores de propriedade cujo espaço de definição corresponde ao conjunto de entidades ou associação referido por ela. Um determinado valor de propriedade é uma ocorrência do tipo de propriedade (YONG, 1990).

1.3 Tipos de modelos de dados

Yong (1990) descreve quatros modelos de dados que são: Modelo de rede, Modelo hierárquico, Modelo relacional e Modelo entidade, que serão descritos logo a seguir.

- Modelo de rede

Segundo Yong (1990) o modelo de rede utiliza o conceito de associação como unidade básica.

- Modelo hierárquico

Yong (1990) descreve que o modelo hierárquico pode ser considerado como um caso particular do modelo de rede, que utiliza como unidade básica o conceito de "nesting" entre as entidades identificadas no empreendimento.

- Modelo relacional

O modelo relacional emprega a relação matemática como unidade básica, para tanto, usa a teoria dos conjuntos como base formal para a descrição de modelos de dados (YONG, 1990).

- Modelo entidade

O modelo entidade utiliza o conceito de entidade como elemento básico, outros conceitos como propriedades, associação, entre outras. São também utilizados, na fase de percepção do mundo real (YONG, 1990).

A modelagem de dados é composta por três graus de abstração de dados que são: modelo conceitual, modelo lógico, e modelo físico como ilustra a figura 2.

Figura 2. Graus de abstração de dados

Fonte: http://www.diegomacedo.com.br/modelagem-conceitual-logica-e-fisica-de-dados/

1.4 Representação do Modelo Conceitual

A elaboração do modelo conceitual constitui basicamente em obter a representação do mundo real percebido (etapa 1), em termos de modelo de informações. Cada tipo de modelo de informações utiliza elementos básicos diferentes, possuindo forma de definição e manipulação de dados e técnica de diagramação distinta (YONG, 1990).

Segundo Yong (1990) em resumo o modelo conceitual é composto de:

1) Um conjunto de elementos básicos de representação de dados, ou também conhecidos como tipos de estruturas de dados distintos dentro do modelos de dados.

2) Regras de composição de cada elemento em função das outras.

3) Para cada elemento básico é definido um conjunto de atributos e o conjunto de valores de cada atributo.

4) Regras evolutivas.

5) Restrições de integridade.

6) Forma e substância para a construção de esquemas através de linguagens de definição de dados.

Para Macedo (2011) o objetivo aqui é criar um modelo conceitual de forma gráfica, sendo este chamado de Diagrama Entidade e Relacionamento (DER), que identificará todas as entidades e relacionamentos de uma forma global, O modelo conceitual mais utilizado é o de ER, que é ajudado pelo DER, que na prática, constitui o modelo básico do BD. Este é

utilizado para representar graficamente o esquema conceitual.

Segundo Luís (2008) a modelagem conceitual baseia-se no mais alto nível e deve ser usada para envolver o cliente, pois o foco aqui é discutir os aspectos do negócio do cliente e não da tecnologia.

Os exemplos de modelagem de dados vistos pelo modelo conceitual são mais fáceis de compreender, já que não há limitações ou aplicação de tecnologia específica. O diagrama de dados que deve ser construído aqui é o Diagrama de Entidade e Relacionamento, onde deverão ser identificados todas as entidades e os relacionamentos entre elas. Este diagrama é a chave para a compreensão do modelo conceitual de dados (LUIS, 2008).

Atributos

Na concepção de Alexandruk (2011) dados que são associados a cada ocorrência de uma entidade ou de um relacionamento.

Cardinalidades

Alexandruk (2011) número (mínimo, máximo) de ocorrências de entidade associadas a uma ocorrência da entidade em questão através do relacionamento.

- Cardinalidade mínima

É o número mínimo de ocorrências de entidade que são associadas a uma ocorrência da mesma.

Á cardinalidade mínima 1 recebe a denominação de associação obrigatória, já que ela indica que o relacionamento deve obrigatoriamente associar uma ocorrência de entidade a outra. A cardinalidade mínima 0 recebe a associação opcional.

- Cardinalidade máxima

É o número máximo de ocorrências de entidade que são associadas a uma mesma ocorrência da mesma ou de outra entidade através de um relacionamento. Apenas duas cardinalidade máximas são relevantes: á cardinalidade máxima 1 e á cardinalidade máxima **n** (muitos).

1.5 Implantação Lógica

Com referência a modelo de dados, a fase de implantação lógica pode ser considerada como sendo análise de rotas lógicas de acesso dos diversos modelos de dados.

Tais rotas de acesso tem maior facilidade de aplicação nos modelos de rede e hierarquia, dada à facilidade de visualização via diagramas de estruturas de dados (YONG, 1990).

Os outros modelos, mais próximos do nível conceitual, metal (por exemplo, relacional, e entidade relacionamento), não efetuam análise de rotas de acesso, nem pela finalidade de utilização de modelo (o modelo entidade-relacionamento é um modelo típico para modelagem e inadequado a análises de rotas) nem como técnicas de manipulação especiais (o modelo relacional utiliza operadores e álgebra relacional na composição, em tempo de consulta, “rotas de acesso”) (YONG, 1990).

Segundo Chen (1990) o projeto lógico de banco de dados é o processo de planejar a estrutura lógica de dados para o banco de dados, isto envolve uma análise do ambiente de aplicação e dois tipos de estruturas lógicas de dados disponíveis no sistema do banco de dados.

O modelo lógico já leva em conta algumas limitações e executa recursos como adequação de padrão e nomenclatura, define as chaves primárias e estrangeiras, normalização, integridade referencial, entre outras. O modelo lógico deve ser criado de acordo com os exemplos de modelagem de dados criados no modelo conceitual (LUIS, 2008).

A Figura 3 ilustra o modelo lógico de dados, o diagrama E-R (entidade-relacionamento) de uma fábrica de móveis, com suas respectivas entidades, cardinalidades mínimas e máximas de cada relação, e chaves primárias.

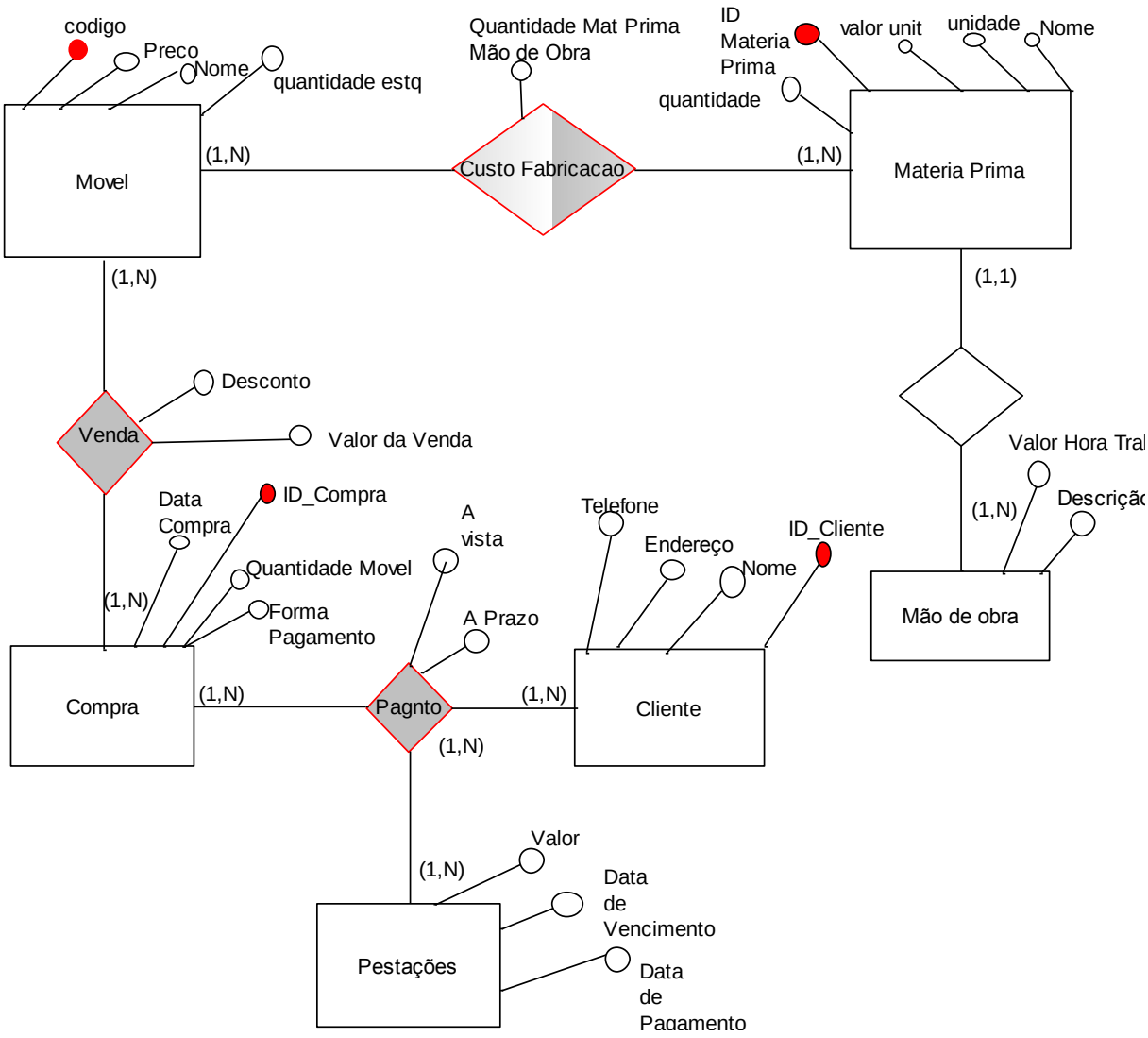

Figura 3. Modelo Lógico de Dados

Fonte: Próprio Autor

1.5.1 Principais Etapas no Projeto Lógico de banco de dados

Segundo Chen (1990) o modelo Entidade- Relacionamento para projeto lógico de banco de dados consiste nas seguintes etapas:

1. Identificar tipos de entidades.
2. Identificar tipos de relacionamentos.
3. Desenhar um diagrama E-R com tipos de entidade e relacionamentos.
4. Identificar tipos de valores e atributos.
5. Traduzir o diagrama E-R em um diagrama de estrutura de dados.
6. Projetar formatos de registros.

1.5.2 Problemas no Projeto Lógico de Banco de dados

Chen (1990) relata alguns problemas no modelo ou projeto lógico de banco de dados.

1) O projetista de banco de dados tem de considerar muitas questões ao mesmo tempo, o que torna a tarefa de projeto de banco de dados muito difícil.

2) O resultado final do projeto lógico de banco de dados é o esquema (ou seja, uma descrição da visão do banco de dados pelo usuário). Uma vez que o esquema do usuário representa a solução do projetista para as questões complicadas.

1.5.3 Modelo Entidade-Relacionamento

Para Chen (1990) a ideia chave do modelo E-R é acrescentar um estágio intermediário ao projeto lógico de banco de dados.

A figura 4 demonstra um exemplo do método E-R, com entidades, e seus respectivos relacionamentos, com cardinalidades mínimas e máximas.

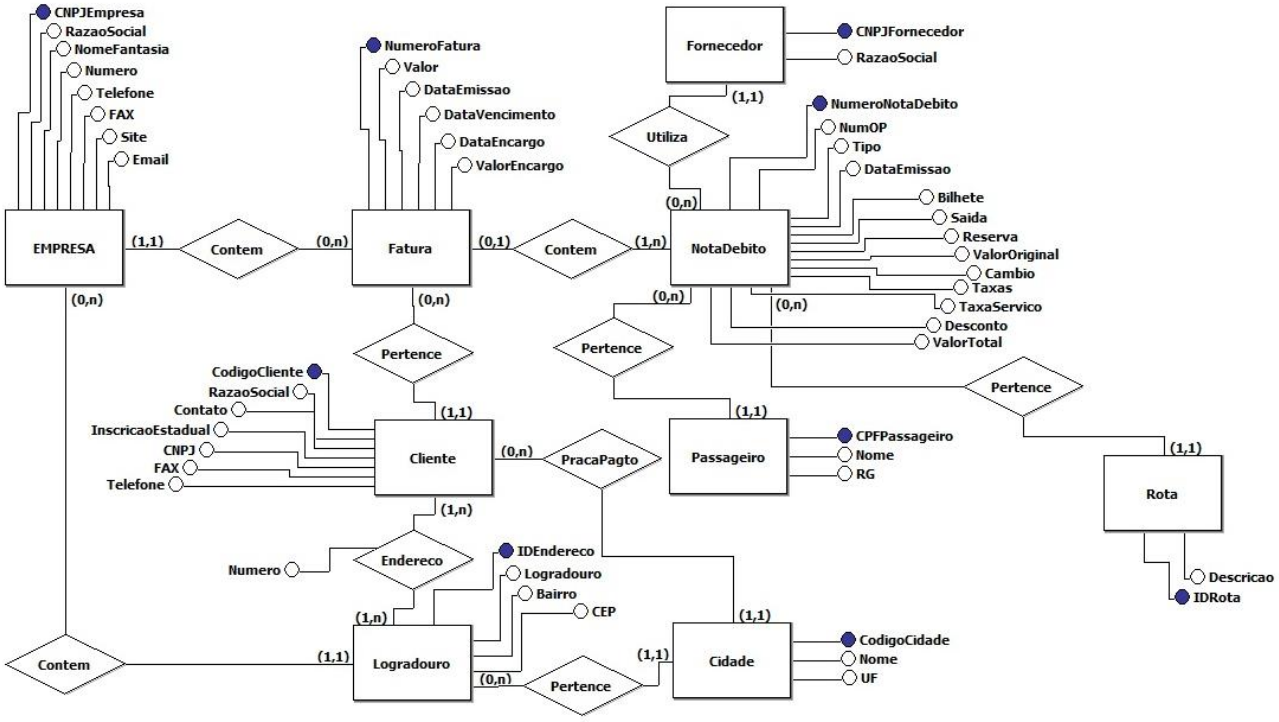

Figura 4. Representação do Modelo E-R Entidade- Relacionamento

Fonte: FUSCO (2014)

1.5.4 Vantagens do Modelo Entidade Relacionamento

Segundo Chen (1990) as vantagens do modelo E-R são:

1. A divisão da funcionalidade e trabalho em duas fases torna o processo do projeto de banco de dados mais simples e mais bem organizados.
2. O esquema da empresa é mais fácil de ser projetado do que o esquema do usuário, uma vez que não precisa ser restrito pelas características do sistema de banco de dados e é independente de considerações sobre armazenamento e eficiência.
3. O esquema da empresa é mais estável do que o esquema do usuário. Se uma pessoa quiser mudar de um sistema de banco de dados para outro, provavelmente terá de alterar o esquema do usuário, mas não o esquema da empresa, uma vez que este é independente dos sistemas de banco de dados usados. O que precisa ser feito é um remapeamento do esquema da empresa para um esquema do usuário compatível com o novo sistema de banco de dados. De forma semelhante, se uma pessoa quiser mudar o esquema do usuário para aperfeiçoar um novo programa de aplicação, não é necessário alterar o esquema da empresa, mas apenas representá-lo através de gráficos para um novo esquema do usuário.
4. O esquema da empresa expresso pelo diagrama de entidade-relacionamento é mais facilmente compreendido por pessoas não ligadas ao processamento eletrônico de dados.

1.6 Representação do Modelo Físico

Segundo Chen (1990) o projeto físico de banco de dados é o processo de selecionar uma estrutura física de dados para uma dada estrutura lógica de dados. A finalidade do projeto físico de banco de dados é selecionar a estrutura física de dados que seja mais adequada para determinado ambiente de aplicação.

A Figura 5 ilustra o modelo físico de dados Cliente, com os respectivos registros, scripts de banco de dados, que contem a criação da tabela, com as respectivas colunas, e o tipos de dados que cada coluna deve-se armazenar, e também a restrição de primary key (chave primária) a coluna Id_Cliente.

```
Create table Cliente(
Id_Cliente int  not null,
Nome varchar(50) not null,
Sobrenome varchar(50) not null,
Idade int not null,
Sexo char(1) not null,
Logradouro varchar(80) not null,
Bairro varchar(50) not null,
Telefone varchar(20) not null,
Celular varchar(20) not null,
constraint PK_Cliente primary key (Id_Cliente));
```

Figura 5. Modelo Físico de Dados

Fonte: Próprio Autor

1.7 Normalização

Normalização de dados é o processo formal, é o passo a passo que examina os atributos de uma entidade, com o objetivo de evitar anomalias observadas na inclusão, exclusão e alteração de registros (LUIS, 2008).

A normalização de dados é um processo em que os atributos de dados dentro de um modelo de dados são organizados para aumentar a coesão dos tipos de entidade. Em outras palavras, o objetivo da normalização de dados é reduzir e até eliminar a redundância de dados (AGILEDATA, 2015).

Podem-se considerar três níveis de normalização que são 1° forma normal, 2° forma normal, e 3° forma normal, em algumas literaturas é citada a 4° forma normal e também a 5° forma normal.

Algumas refatorações de banco de dados possuem o conceito de normalização, como a refatoração renomear coluna, por exemplo, da categoria de refatoração estrutural.

1.7.1 Primeira Forma Normal (1FN)

Segundo Yong (1990) uma relação R está na primeira forma normal (1FN) se e somente todos os domínios considerados contenham somente valores atômicos (isto é, valores que não são relações).

1.7.2 Segunda Forma Normal (2FN)

Segundo Yong (1990) uma relação R está na segunda forma normal (2FN) se ela está na (1FN) e cada atributo dependente é funcionalmente dependente do atributo-chave e sem ser funcionalmente dependente de qualquer subconjunto do atributo-chave (no caso ser composto de vários atributos).

1.7.3 Terceira Forma Normal (3FN)

Segundo Yong (1990) uma relação R está na terceira forma normal (3FN) se está na 2FN e os atributos dependentes não são dependentes transitivos do atributo-chave.

1.8 Considerações Finais Sobre o Capítulo

Neste capítulo foram abordados os conceitos sobre modelagem de dados, abordados suas respectivas etapas, como, a percepção do mundo real onde é feita a análise do cenário em questão, modelagem conceitual que é a modelagem de mais alto nível, sem levar em conta tecnologia, modelagem ou implantação lógica. Foram analisadas as informações demonstradas no modelo conceitual, e explicitado que por conta disso serão criadas as entidades, relacionamentos entre as entidades, atributos, restrições de integridade, e a criação do modelo lógico de dados que pode ser mostrado pelo diagrama E-R (entidade-relacionamento), ou o ME-R (Método ou modelo entidade-relacionamento).

Foi citado rapidamente o modelo ou projeto físico de dados que é uma etapa importante na modelagem de dados, onde é definido literalmente o SGBD do banco de dados que será implantado e utilizado, foi trabalhado o conceito sobre normalização e suas forma básicas de normalização que são: primeira forma normal denominada pela sigla (1FN), segunda forma normal (2FN), e terceira forma normal (3FN), e explicado que em algumas literaturas e fontes da web enfatiza-se a quarta forma normal (4FN), e a quinta forma normal (5FN), além de discorrer sobre algumas técnicas de refatoração de banco de dados que contém o conceito de normalização.

No capítulo seguinte serão abordados e contextualizados os conceitos sobre refatoração, com exemplos sobre o assunto, serão abordados também, estudos de como a

refatoração pode melhorar um projeto de software, o quanto ela ajuda a encontrar falhas, o porquê e quando se deve refatorar.

2 REFATORAÇÃO

Neste capítulo serão contextualizados os conceitos sobre refatoração, que se encontram no livro de Martin Fowler (1999), Aperfeiçoando o projeto de código existente, definição de refatoração pelo autor, o porquê de refatorar, o melhoramento da refatoração em um projeto de software, como fazer a refatoração e ajudar a encontrar falhas, e também os problemas com a mesma.

Martin Fowler (1999) define refatoração em dois termos, em um substantivo e um verbo que é apresentado a seguir.

De acordo com Martin Fowler (1999, p.50 e 51)

> "[...] Refatoração (substantivo) uma alteração feita na estrutura interna do software para torna-lo mais fácil de ser entendido e menos custoso de ser modificado sem alterar seu comportamental observável". [...]
>
> [...] Refatoração (verbo) reestruturar software aplicando uma serie de refatorações sem alterar seu comportamento observável. [...]".

Refatoração fornece uma técnica para limpar código de uma maneira mais eficiente e controlada (FOWLER, 1999).

2.1 Refatoração e a qualidade do projeto de software

Sem refatoração, o projeto do programa acaba por se deteriorar. A medida com que as pessoas alteram o código, alterações para executar objetivos de curto prazo ou então alterações feitas sem uma compreensão total do projeto do código, o código se desestrutura (FOWLER, 1999).

Torna-se mais difícil visualizar o projeto ao ler o código. Refatoração é como arrumar o código. As partes do código que não estejam no lugar apropriado são removidas, a perda de estrutura do código tem um efeito cumulativo, quanto mais difícil é visualizar o projeto a partir do código, mais difícil será preservá-lo e mais rapidamente ele se desestruturará, a refatoração sistemática ajuda o código a conservar a sua forma (FOWLER, 1999).

Um sistema mal projetado normalmente precisa de mais código para fazer as mesmas coisas, muitas vezes porque o mesmo código é replicado em diversos lugares diferentes, assim, um aspecto importante na melhoria do projeto é a eliminação de código duplicado, a

importância disto recai sobre futuras modificações no código, reduzir a quantidade de código não fará o sistema rodar mais rapidamente, porque o efeito no desempenho dos programas é raramente significativo, reduzir a quantidade de código faz, todavia, uma grande diferença sobre a manutenção deste código, quanto mais código houver, mais difícil será modificá-lo corretamente (FOWLER, 1999).

2.2 Refatorar ajuda a encontrar falhas

Refatorar pode ajudar a encontrar falhas no código, que não foram encontradas no início do desenvolvimento, o objetivo aqui é entender o código, o que o código faz, e sendo encontradas falhas, corrigi-las, para um código mais confiável e com menos erros e falhas possível.

Segundo Martin Fowler (1999, p.53)

> "[...] A melhora no entendimento do código também me ajuda a localizar falhas. Admito que não sou muito bom em encontrar falhas, algumas pessoas conseguem ler um bloco de código e achar falhas, eu não, entretanto descubro que, se recapturo código, trabalho pesado na compreensão do que ele faz, e aplico esse novo conhecimento de volta no código[...]".

2.3 Refatorar ajuda a programar mais rapidamente

Segundo Fowler (1999) no final, todos os pontos anteriores levam a este. Refatorar ajuda você a desenvolver código mais rapidamente. As pessoas podem ver facilmente que ela melhora a qualidade. Melhorar o projeto, melhorar a legibilidade, reduzir falhas, tudo isto melhora a qualidade.

Segundo Martin Fowler (1999, p. 54)

> "[...] Um bom projeto é essencial na manutenção da velocidade de desenvolvimento de software. Refatorar ajuda você a desenvolver software mais rapidamente, por que evita que o projeto do sistema se deteriore. A refatoração pode até mesmo melhorar um projeto [...]".

2.4 Quando refatorar

Refatorar exige tempo, mas, com o desenvolvimento em andamento separar tempo para refatorar pode deixar o desenvolvimento mais demorado, refatoração não deve ser feita de uma vez ou em certo período do desenvolvimento mas, a todo tempo como afirma Fowler.

De acordo com Martin Fowler (1999, p. 54)

> "[...] Quando falo em refatoração, muitas vezes me perguntaram sobre como ela deve ser programada". Devemos alocar duas semanas a cada dois meses para refatorar.
> Na maioria dos casos, sou contra separar tempo para refatorar. Na minha visão, a refatoração não é uma atividade para a qual você separar tempo. Refatorar é algo que você faz todo o tempo em pequenas rajadas. Você não decide refatorar, você refatora porque quer fazer alguma outra coisa qualquer e refatorar ajuda a fazê-lo [...]".

2.5 Refatore quando acrescentar funções

Segundo Martin Fowler (1999, p. 54 e 55)

> "[...] A hora mais comum para refatorar é quando quero adicionar uma nova característica a algum software". Muitas vezes a primeira razão para refatorar aqui é me ajudar a compreender algum código que preciso modificar. Este código por outra pessoa ou por mim. Toda vez que tenho que pensar para entender o que o código faz, me pergunto se posso refatorar o código para tornar esse entendimento mais rápido. Então eu o refatoro [...]
>
> [...] Outro motivo que conduz á refatoração aqui é um projeto que não ajuda a acrescentar uma nova funcionalidade facilmente. Olho o projeto e digo para mim mesmo "se eu tivesse projetado o código desta forma, acrescentar esta característica seria fácil [...]".

Refatorar é um processo rápido tranquilo, uma vez que você tenha refatorado, adicionar uma nova característica pode ser muito mais rápido e tranquilo (FOWLER, 1999).

2.6 Refatore quando precisar consertar uma falha

No conserto de falhas, muito da utilidade da refatoração vem do fato de tornar o código mais compreensível (FOWLER, 1999).

Segundo Martin Fowler (1999, p. 55)

> "[...] Quando olho o código tentando compreendê-lo, refatoro para ajudar a melhorar minha compreensão. Muitas vezes descubro que este processo ativo de trabalho com o código ajuda a encontrar falha. Um modo de encarar isso é que, se você recebe uma notificação de falha, é um sinal de que precisa refatorar, porque o código não estava claro o suficiente para que percebesse que havia uma falha [...]".

2.7 Refatore enquanto revisa o código

Segundo Fowler (1999) algumas organizações fazem revisões de código regulares. Aquelas que não o fazem seriam melhores se fizessem. Revisões de código ajudam a difundir conhecimento através de uma equipe de desenvolvimento. As revisões facilitam a transmissão do conhecimento dos desenvolvedores mais experientes para os menos experientes. Elas ajudam as pessoas a entender melhor os aspectos de um sistema grande de software. Elas

também são muito importantes na escrita de código.

Martin Fowler afirma (1999, p. 55)

> "[...] Meu código pode parecer claro pra mim, mas não para minha equipe". Isto é inevitável é muito difícil para as pessoas se colocarem no lugar de alguém que não esteja familiarizado com as coisas nas quais estão trabalhando. Revisões também dão a oportunidade de mais pessoas sugerirem ideias uteis. Eu precisaria de uma semana para pensar tantas boas ideias. Fazer outras pessoas contribuírem torna minha vida mais fácil, então busco sempre por novas revisões [...]
>
> [...] Descobri que refatorar me ajuda a revisar o código de outras pessoas. Antes de começar a refatorar, podia ler o código, obter algum grau de compreensão do mesmo e fazer algumas sugestões. Agora tenho uma ideia, considero se ela pode ser fácil e imediatamente implementada com refatoração [...]".

2.8 Problemas com refatoração

Segundo Fowler (1999) uma área problemática para refatoração são os bancos de dados. A maioria das aplicações comerciais são altamente acopladas ao esquema de banco de dados que as suportam. Esse é o motivo pelo qual os bancos de dados são difíceis de modificar. Outro motivo é a migração de dados.

Mesmo se você tiver colocado seu sistema cuidadosamente em camadas para minimizar as dependências entro esquema do banco de dados lhe força a migrar os dados, o que pode ser uma tarefa longa e pesada.

2.9 Considerações Finais Sobre o Capítulo

Neste capítulo foram abordados os conceitos sobre refatoração descritos por Martin Fowler (1999) em seu livro "Aperfeiçoando o projeto de código existente", com a participação de outros autores, foi definido o significado de refatoração na percepção de Fowler, como refatoração pode ajudar a encontrar falhas no código do software, quando e por que refatorar o código. Também esteve presente neste capítulo a preocupação de esclarecer como a refatoração é importante no desenvolvimento de software, para detecção de falhas e correção das mesmas, mostrar que refatoração além de corrigir falhas no código também ajuda a programar mais rapidamente e ajuda entender melhor o propósito do código em questão, Fowler não prega que a refatoração resolve todos os problemas, assim como a refatoração de banco de dados também não resolve todos os problemas, mas, é uma técnica que ajuda o programador de software a programar de forma mais clara e coerente, deixando o sistema mais correto possível e que outras pessoas possam entender o que o sistema faz e suas

funcionalidades.

No capítulo seguinte será discorrido sobre os conceitos de refatoração de banco de dados, será mostrado como a refatoração de banco de dados tem o propósito de detectar e corrigir erros nos esquemas de dados, como a refatoração de código e a refatoração de banco de dados não resolve todos os problemas, mas, ajuda a corrigir o que não foi percebido no começo do desenvolvimento. Mostrará como a refatoração de banco de dados é uma técnica que melhora a estrutura do banco deixando-o mais confiável e mantém a semântica das informações contidas no mesmo, além de informar que a refatoração de banco de dados possui seis categorias descritas por Scott W. Ambler e Sadalage (2006) que são: estrutural, qualidade de dados, integridade referencial, arquitetural, método e, transformações não refatoração, todas estas categorias de refatoração possui ao todo 70 técnicas de refatoração, nos trabalhos correlatos são referenciados quatro categorias com suas respectivas ações e refatorações.

O capítulo seguinte também mostrará as vantagens e desvantagens que a refatoração de banco de dados traz, diferenças entre refatoração de código e refatoração de banco de dados, além de contextualizar que algumas refatorações contêm um período de transição e um código de apoio por que algumas refatorações ou técnicas de refatoração que possuem replicação de dados para manter os mesmos atualizados.

3 REFATORAÇÃO DE BANCO DE DADOS

Segundo Ambler e Sadalage Apud Domingues (2006 e 2014) uma refatoração em banco de dados pode ser definida como uma alteração simples no 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 (DOMIGUES, 2014).

A refatoração de um banco de dados consiste em uma melhoria junto a sua estrutura, onde se aprimora o projeto sem a adição de novas funcionalidades e sem alteração das funcionalidades já existentes ou da semântica informacional (DALPRA e ARAÚJO, 2012).

Refatorar um banco de dados introduz uma fase intermediária necessária entre o ajuste e as correções, onde se propõe a realização dos processos de maneira diferente com ganho significativo em eficiência, em comparação com a mesma aplicação no passado (DALPRA e ARAÚJO, 2012).

Há uma diferença crucial entre refatoração e adição de nova funcionalidade. Na refatoração busca-se uma melhora da estrutura existente, já ao inserir uma nova funcionalidade acrescenta-se uma nova estrutura. Em alguns casos pode ser necessário refatorar o banco de dados para posteriormente inserir uma nova funcionalidade, ou então se pode identificar à necessidade de refatorar a nova funcionalidade inserida junto ao banco. Sendo assim, conclui-se que refatoração e adição de novas funcionalidades são coisas distintas, porém complementares (DALPRA e ARAÚJO, 2012).

Em outras palavras, após a execução de uma refatoração ou um conjunto de refatorações no esquema de dados, o banco de dados deve responder as mesmas consultas que eram realizadas antes deste processo de refatoração (DOMINGUES, 2014).

A estratégia da refatoração definida por Ambler é aplicada em ambientes complexos, nos quais existem várias aplicações sendo acessadas simultaneamente no mesmo banco de dados. Quando esse banco precisa ser alterado, praticamente todas as aplicações precisam sofrer algum tipo de adaptação, no entanto, 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 da refatoração, no qual o esquema antigo e o esquema novo coexistam (DOMINGUES, 2014).

Conforme ilustra a figura 6, em que um ciclo de vida de uma refatoração de banco de dados, tenha-se um período de transição entre o início e o fim de uma refatoração, em que os dois esquemas, o esquema antigo e o esquema novo existam nesse contexto.

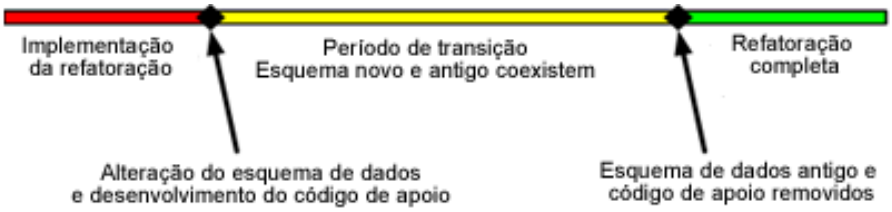

Figura 6. Ciclo de vida de uma refatoração

Fonte: adaptado de AMBLER, SADALAGE, Apud DOMINGUES (2006 e 2014)

Ambler Apud Domingues (2006 e 2014) destacam 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 (DOMINGUES, 2014).

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 Saladage (2006) escreveu um catálogo para as refatorações que foi amplamente discutido por Domingues (DOMINGUES, 2014).

As refatorações são divididas em seis categorias: estrutural, qualidade de dados, integridade referencial, arquitetural, método, e transformações, no projeto apresentado por Márcia, à autora apenas aborda quatro categorias que estão descritas no parágrafo abaixo, e o autor desse presente trabalho acrescenta as duas últimas categorias de refatoração em banco de dados.

3.1 Período de Transição e Código de Apoio

O período de transição em que os esquemas novo e antigo do banco de dados coexistem, exige um código de apoio para manter os dois esquemas sincronizados. No caso mais simples quando as alterações dos dados ocorrem somente no esquema novo e o antigo é utilizado somente para consultas temos um código que deve replicar todas as alterações do modelo novo para o antigo. Desse modo, o código de apoio permite que as aplicações não sejam afetadas pela existência de dois esquemas elas sempre acessam os dados atualizados (DOMINGUES, 2011).

3.2 Categorias de Refatoração de Banco de Dados

As refatorações de banco de dados são divididas em seis categorias: estrutural, qualidade de dados, integridade referencial, arquitetural, método e, transformações-não-refatoração, nas seções seguintes serão apresentadas o conceito de cada categoria de refatoração, e, as ações que cada categoria inclui em um esquema de banco de dados.

3.2.1 Estrutural

As refatorações estruturais têm como objetivo melhorar o esquema do banco de dados. Mudança 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: (DOMINGUES, 2014).

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

3.2.2 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: (DOMINGUES, 2014).

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

3.2.3 Integridade Referencial

As refatorações de integridade referencial têm como objetivo manter consistentes as

referências entre tabelas, incluindo ou removendo regras, armazenando históricos ou executando lógicas de aplicação no banco de dados. As refatorações de integridade referencial podem incluir ações como: (DOMINGUES, 2014).

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

3.2.4 Arquitetural

O objetivo principal das refatorações arquiteturais é mover uma parte do código dos aplicativos para o banco de dados (lógica do banco de dados). Os objetivos secundários são organizar, padronizar, e melhorar o desempenho do banco de dados (DOMNIGUES, 2014).

As refatorações arquiteturais podem incluir ações como:

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

3.2.5 Método

As Refatorações de Método consistem em buscar por uma melhor qualidade em Stored Procedure, Function ou trigger (DALPRA e ARAÚJO, 2012).

As Refatorações de Método estão divididas em refatorações que mudam a interface de um banco de dados por meio de programas externos e refatorações que fazem mudanças no interior de um método (DALPRA e ARAÚJO, 2012).

A refatoração de banco de dados Método, é uma mudança que melhora a qualidade de um procedimento armazenado, função armazenada ou gatilho de modo a melhorar o seu projeto de banco de dados, sem alterar sua semântica (AGILEDATA, 2015).

3.2.6 Transformação Não Refatoração

As refatorações de transformações são mudanças que alteram a semântica do esquema do banco de dados, adicionando novos recursos ao mesmo. As Refatorações de Transformação são as seguintes: (DALPRA e ARAÚJO, 2012).

- Insert Data;

- Introduce New Column;
- Introduce New Table;
- Introduce View;
- Update Data.

A transformação não refatoração é uma mudança que afeta a semântica de seu esquema de banco de dados, adicionando novos elementos ao mesmo ou modificando elementos existentes (AGILEDATA, 2015).

Nem todas as alterações no esquema de 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, Apud DOMINGUES 2006 e 2014). As transformações estruturais podem incluir, no esquema do banco de dados, tabelas, colunas, visões (DOMINGUES, 2014).

Algumas refatorações propostas por Ambler e Sadalage Apud Domingues (2006 e 2014) necessitam de replicação 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 duas estratégias que necessitam de sincronização dos dados: o uso de triggers síncronos e processo em lote (batch) (DOMINGUES, 2014).

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 código de apoio que percorre toda a tabela origem para descobrir o que foi alterado e acrescentado, comparando a tabela destino para manter a sincronização dos dados (DOMINGUES, 2014).

Ambler e Sadalage Apud Domingues (2006 e 2014) enfatiza que o uso de trigger síncronos é a melhor solução para a 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 (DOMINGUES, 2014).

Para superar essas dificuldades, uma abordagem alternativa consiste 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 (DOMINGUES, 2014).

3.3 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 foram a operação (inserção, exclusão ou atualização), quais os valores anteriores existentes na tabela e quais os novos valores usados para a atualização (DOMINGUES, 2014).

Conforme é ilustrado na figura 7 o trigger coletor.

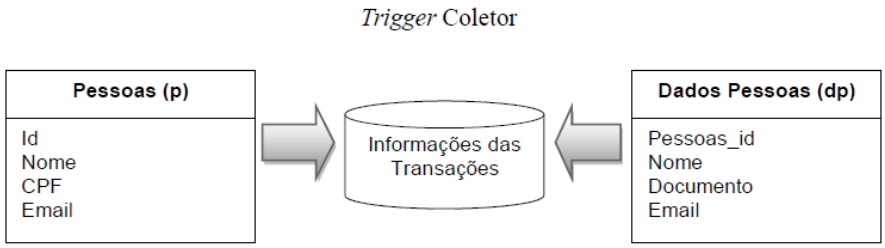

Figura 7. Replicação de dados Assíncrona: etapa de coleta de dados
Fonte: Adaptado de DOMINGUES Apud DOMNINGUES (2011 e 2014)

3.4 Mapeamento

Nessa etapa são definidos os mapeamentos de origem e destino das transações (DOMINGUES, 2014). Conforme é ilustrado na figura 8 o mapeamento.

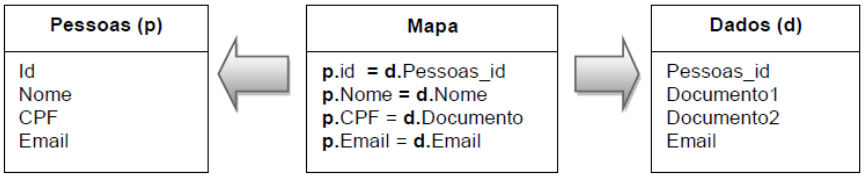

Figura 8. Replicação de dados Assíncrona: Etapa de mapeamento
Fonte: Adaptado de DOMINGUES Apud DOMINGUES (2011 e 2014)

3.5 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, 2014). Conforme é ilustrada na figura 9 a execução.

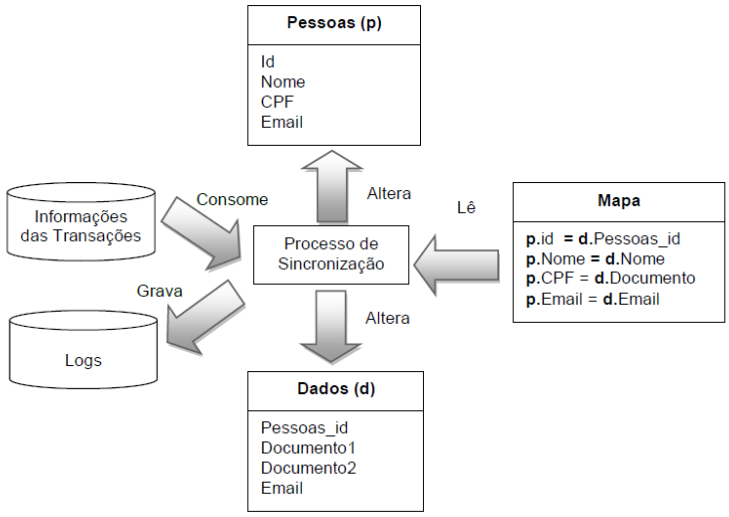

Figura 9. Replicação de dados Assíncrona: Etapa de execução
Fonte: Adaptado de DOMINGUES Apud DOMINGUES (2011 e 2014)

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 *log* 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 (DOMINGUES, 2014).

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 (DOMINGUES 2014).

Na organização de uma sequência de refatorações, deve-se pensar nas possibilidades

de realizar uniões (*merges*), alinhamentos e intercalações de refatorações.

Uma organização bem feita permitirá minimizar a duração de uma grande alteração de banco de dados (DOMINGUES, 2014).

3.6 Vantagens e Desvantagens da Refatoração

Em projetos concluídos, a refatoração pode ser vantajosa por melhorar de forma significativa um projeto mal elaborado. Ou seja, ao refatorar o banco de dados continuamente, é possível corrigir problemas graves, caso existam, que podem se tornar evidentes à medida que se aplicam as refatorações. Além desse benefício, a refatoração pode impedir que o projeto se deteriore com as possíveis alterações que tendem a ocorrer ao longo do tempo (MIGUEL e ARAÚJO, 2011).

Outra vantagem que se torna evidente ao se usar as técnicas de refatoração é o aprendizado. Refatorar ajuda a tornar mais fácil a tarefa bastante comum de analisar e entender os bancos de dados. Nesse processo, é preciso investir intensamente no aprendizado do que a refatoração realmente faz e aplicar este aprendizado no banco de dados. Num nível mais alto, a refatoração permite aprender também sobre a estrutura do projeto, já que à medida que partes do banco de dados se tornam melhor entendidos, entender o projeto como um todo se torna uma tarefa muito mais fácil (MIGUEL e ARAÚJO, 2011).

A velocidade é outra vantagem da refatoração, por que torna mais rápido desenvolver software por todos os motivos explicados anteriormente. Quando se tem um banco de dados e projeto mais claros e fáceis de entender, com maior nível de conhecimento pelos desenvolvedores e com menos falhas, consequentemente pode-se ter maior velocidade de desenvolvimento e manutenção a menores custos (MIGUEL e ARAÚJO, 2011).

3.7 Diferenças entre a refatoração de código fonte e a refatoração de banco de dados

A refatoração consiste em melhorar a estrutura ou legibilidade de um software ou de um sistema de software sem que seu comportamento seja alterado. O processo de refatoração foi inicialmente utilizado por desenvolvedores que se utilizavam da linguagem de programação orientada a objetos Smalltalk. Posteriormente tornou-se essencial no desenvolvimento de frameworks, já que estes, em geral, não ficam totalmente prontos em uma primeira tentativa, tendo evolução no decorrer do tempo. Como o código será lido e

modificado com certa frequência, a refatoração torna-se imprescindível. Usando refatoração, um projeto ruim ou caótico pode ser transformado em algo bem projetado. A junção de pequenas alterações melhora radicalmente o projeto (DALPRA e ARAÚJO, 2012).

Já a refatoração de banco de dados propõe-se a realizar os processos de alteração do banco de maneira diferente, com ganho significativo em eficiência, em comparação à mesma aplicação no passado. É similar à refatoração de código fonte, porém aplica-se a bancos de dados. Essa técnica consiste em um método disciplinado de reestruturação de tabelas em um esquema de banco de dados, a qual ocorre em alterações gradativas. Por meio desta, busca-se o aprimoramento da estrutura de um banco relacional, otimizando consultas, e identificando pontos de melhoria junto às tabelas (DALPRA e ARAÚJO, 2012).

3.8 Considerações Finais Sobre o Capítulo

Neste capítulo foi apresentado o conceito de refatoração de banco de dados definido por Ambler, foi referenciada a tese de doutorado apresentada por Márcia Beatriz Pereira Domingues na Escola Politécnica da Universidade de São Paulo como um guia no presente projeto, além de apresentar as categorias de refatoração e suas respectivas técnicas, foram abordadas e contextualizadas nas técnicas de refatoração as ações que cada categoria possui em suas técnicas, no texto foram abordados temas como replicação de dados, algumas refatorações possuem replicação de dados, e foi visto que não é qualquer mudança que representa refatoração se tratando de banco de dados.

No capítulo seguinte serão apresentados exemplos das quatro categorias de refatoração de banco de dados, que são: estrutural, qualidade de dados, integridade referencial, e arquitetural, todos os exemplos serão retirados do catálogo de refatoração de Ambler que se encontra disponível no endereço eletrônico logo abaixo: http://www.agiledata.org/essays/databaseRefactoringCatalog.html, todos os exemplos sobre as quatro categorias e suas respectivas técnicas serão analisadas e explicadas pelo autor deste presente projeto, além destes conceitos e exemplos sobre refatoração de banco de dados, o site web http://www.agiledata.org também oferece outras leituras complementares sobre vários temas como normalização em banco de dados e testes de regressão de banco de dados.

4 CATEGORIAS DE REFATORAÇÃO EM BANCO DE DADOS

Neste capítulo serão apresentados exemplos sobre as categorias de refatoração de banco de dados, e exemplos sobre as técnicas de refatoração de cada categoria, que como já dito as categorias são: estrutural, qualidade de dados, integridade referencial, arquitetura, método, e transformações não refatoração, serão contextualizados apenas quatro categorias de refatoração nesse capítulo que serão: estrutural, qualidade de dados, integridade referencial, e arquitetural.

4.1 Refatoração Estrutural Exemplos

Como contextualizado na seção 3.2.1 do capítulo 3, a refatoração estrutural tem como objetivo melhorar o esquema do banco de dados, sem alterar a sua semântica, será apresentado a seguir exemplos de refatoração estrutural (DOMINGUES, 2014).

A refatoração de banco de dados estrutural muda a estrutura de uma tabela, coluna ou vista de modo a melhorar o seu projeto de banco de dados sem alterar sua semântica (AGILEDATA, 2015).

- Eliminar Tabela

No exemplo da figura 10 eliminar tabela, tem-se a tabela Tax Jurisdictions, que significa jurisdições fiscais, durante o período de transição (refatoração) a tabela jurisdições fiscais foi removida do banco de dados conforme é ilustrado na mesma.

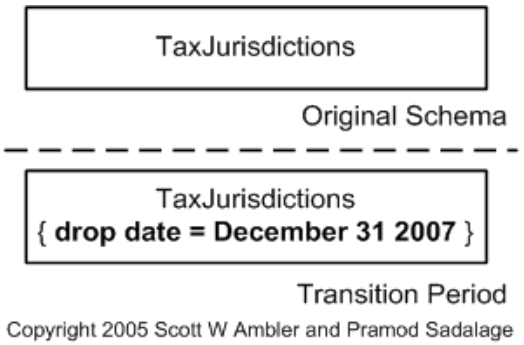

Figura 10. Eliminar Tabela

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Eliminar Coluna

No exemplo da figura 11 eliminar coluna é demonstrado a tabela Customer que traduzindo para o português significa Cliente, com três colunas Customer ID que está denominada como chave primária, FirstName e Favorite Color, que significa (id cliente) primeiro nome e cor favorita como demonstra a figura 11 no esquema original, logo após o período de transição (refatoração) a coluna Favorite Color é removida resultando no esquema da tabela Customer (cliente) apenas duas colunas Customer ID (id cliente) e FirstName (primeiro nome).

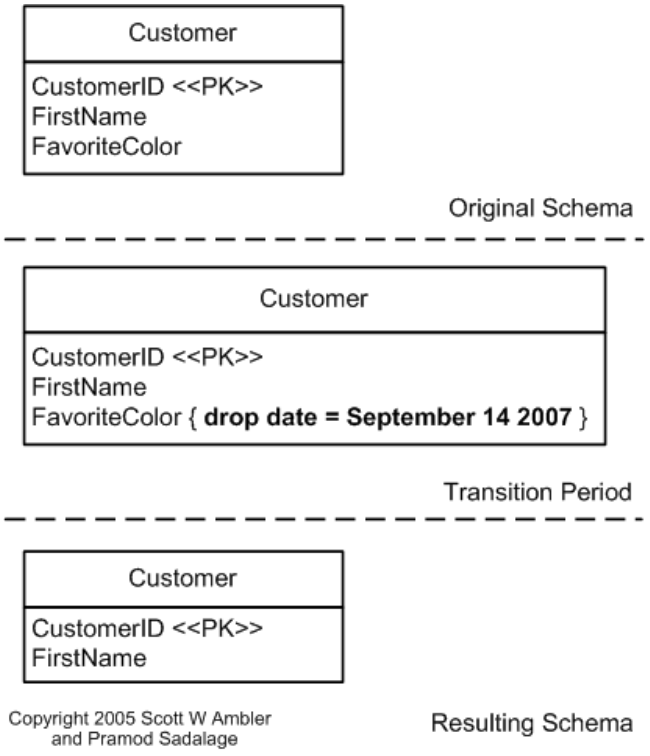

Figura 11. Eliminar Coluna

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Eliminar Visão

No exemplo da figura 12 eliminar visão, no esquema existente contém uma view (visão) Cust Ords, que seria uma abreviação de Customer Orders, ou seja, traduzindo pedido de clientes, logo após o período de transição (refatoração) o esquema resultante é a view (visão) Customer Orders, Pedidos de Clientes.

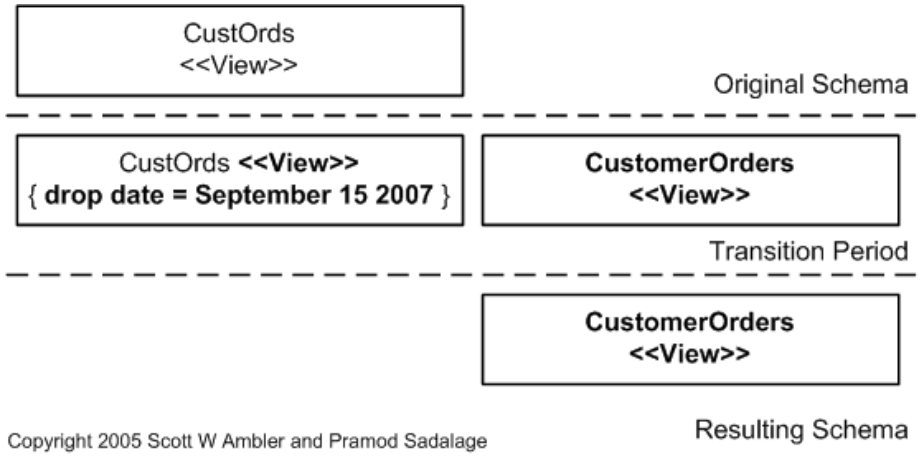

Figura 12. Eliminar Visão

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Introduzir coluna calculada

No exemplo da figura 13 introduzir coluna calculada é demonstrado através de duas tabelas, a tabela Customer que em português significa cliente, e a tabela Account que em português significa (conta), ambas as tabelas possuem três colunas, sendo que a tabela Account (conta) se relaciona com a tabela Customer (Cliente), e a tabela Customer (Cliente) se relaciona com a tabela Account (conta), nota-se que na tabela Account (conta) contêm uma chave primária, a mesma também possui uma chave estrangeira, que denomina- se o relacionamento entre as duas tabelas, além disso, as duas tabelas estão em um relacionamento um para muitos.

No esquema original entre as duas tabelas como demonstrado na figura 12, é cada tabela com três colunas, após o processo de transição (refatoração), o esquema resultante é que na tabela Customer (cliente) é introduzido uma nova coluna, chamada de total Account Balance que significa em português saldo total da conta.

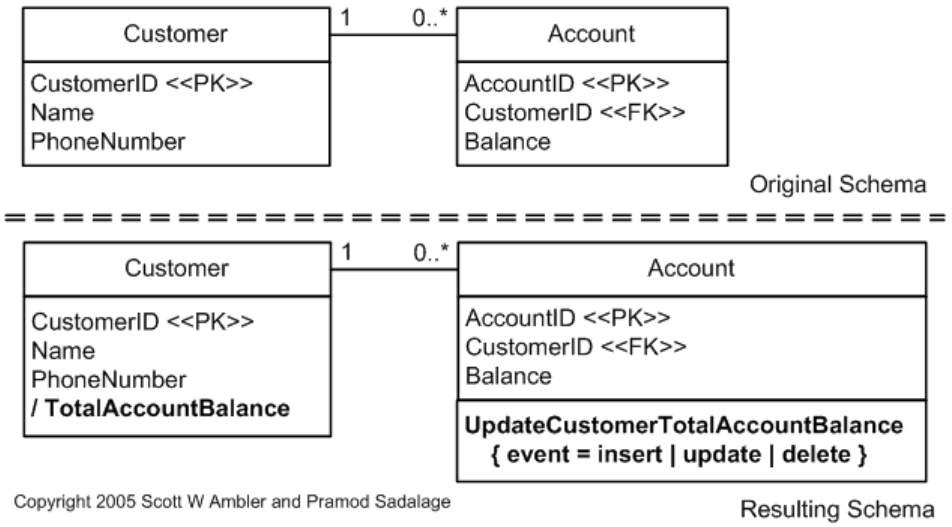

Figura 13. Introduzir coluna calculada

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Introduzir chave de identificação

No exemplo da figura 14 introduzir chave de identificação são demonstradas duas tabelas: a Order que em português significa (ordem), e a Order Item que em português significa item de ordem, a tabela Order (ordem) possuí seis colunas enquanto a tabela Order Item (item de ordem) contém cinco colunas, o esquema original é apresentado desta forma na figura 14, logo após o processo ou período de transição (refatoração) a chave primária natural de ambas as tabelas é substituída.

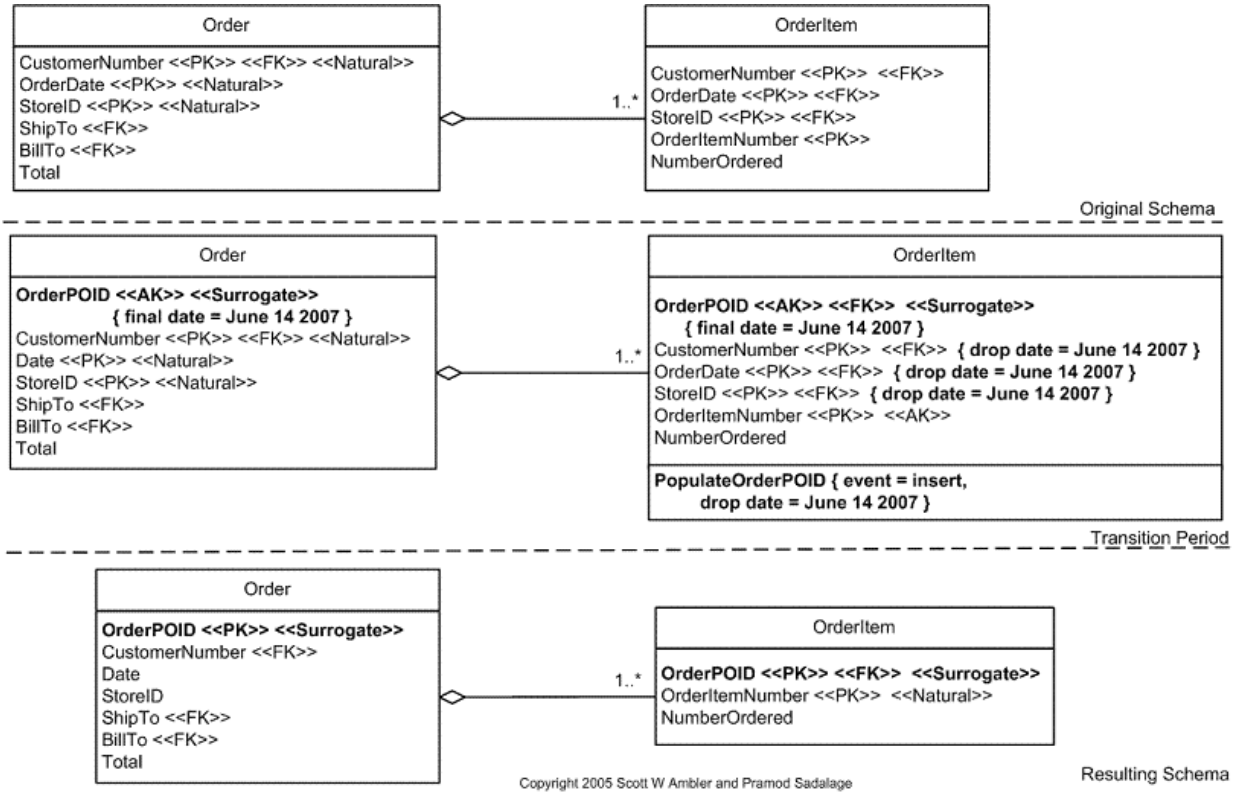

Figura 14. Introduzir Chave de Identificação

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Unir Colunas

No exemplo da figura 15 unir colunas, mostra-se a tabela Customer, já especificado acima, significa em português Cliente, possuí três colunas Phone Country Code, Phone Area Code, Phone Local, que traduzindo para o português significa: telefone código do país, código de área do telefone, e telefone local, no período de transição (refatoração) são excluídas as duas colunas Phone Area Code e Phone Local, e introduzida a uma nova coluna chamada Phone Number que no português significa número de telefone, o esquema resultante é a tabela Cliente com duas colunas Phone Country Code e Phone Number.

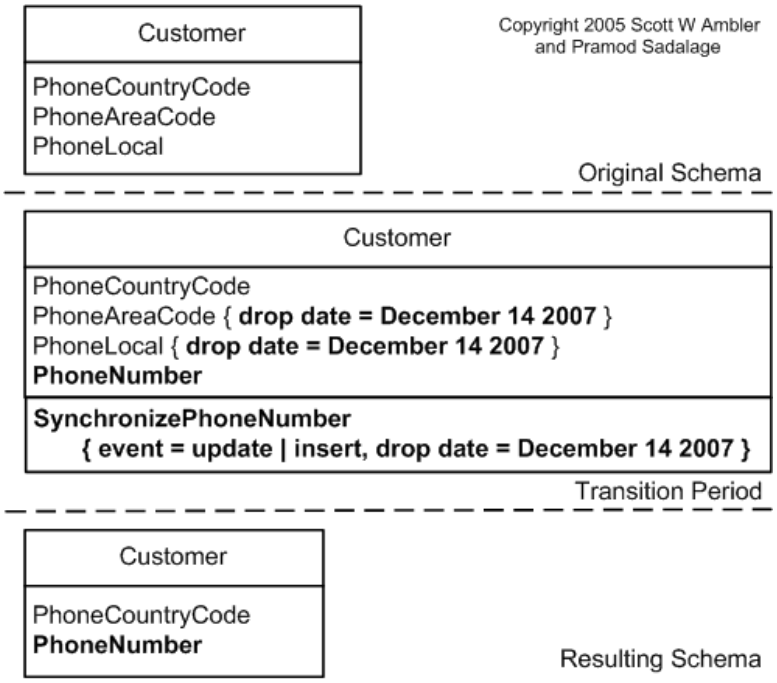

Figura 15. Unir Colunas

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Unir Tabelas

No exemplo da figura 16 unir tabelas, no esquema original do banco de dados como demonstra a figura abaixo, existe duas tabelas Employee que em português, significa empregado, e a tabela Employee Identification que significa identificação do empregado, na tabela Employee (empregado) contém três colunas, sendo a coluna Employee Number, número do empregado como chave primária, enquanto na coluna Employee Identication (identificação do empregado) contém quatro colunas, sendo a coluna Employee Number (número do empregado) como chave primária, as duas tabelas contém um relacionamento um para um, logo após o período de transição (refatoração) as duas tabelas são unidas e se tornam uma só, com as colunas em comum inseridas nesta nova tabela.

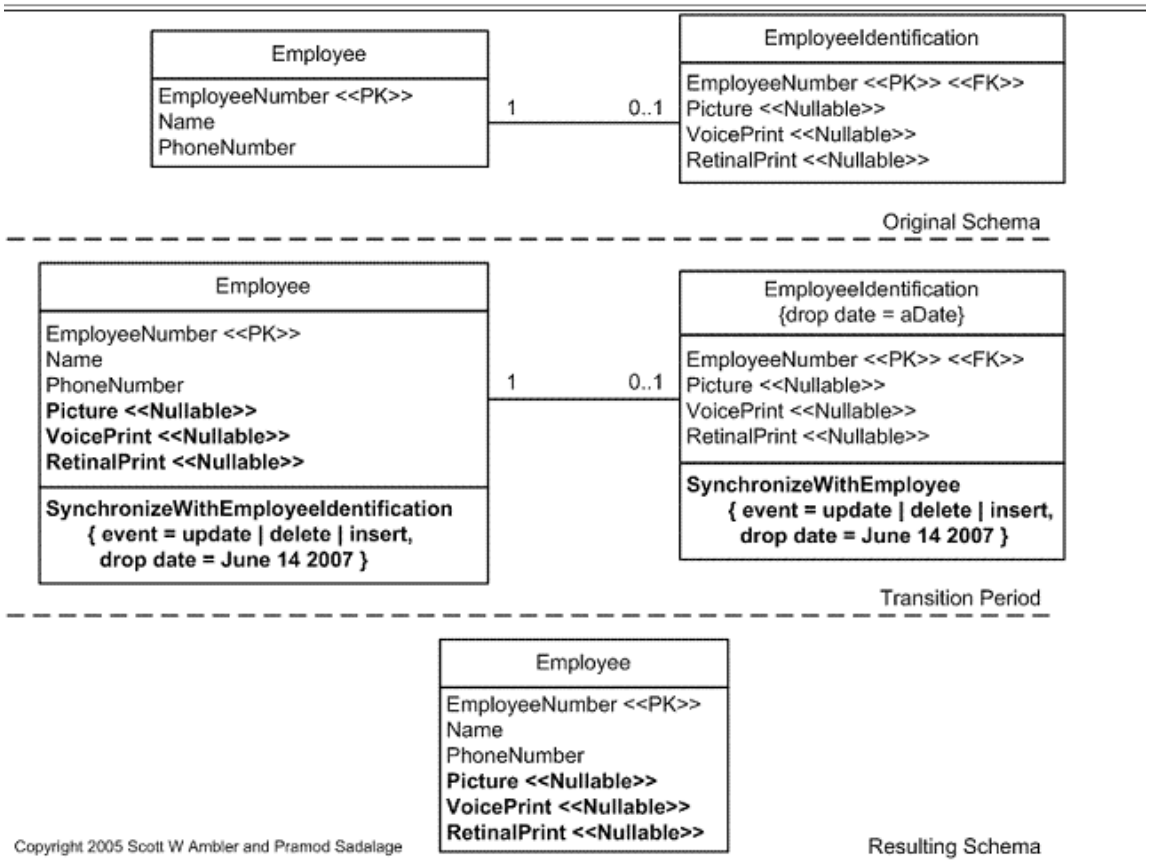

Figura 16. Unir Tabelas

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Mover Coluna

No exemplo da figura 17 mover coluna, a mesma mostra no esquema original duas tabelas, a tabela Customer (cliente) e a tabela Account (conta), a duas tabelas estão relacionadas em um relacionamento um para muitos, a tabela Customer (cliente) contém três colunas sendo a coluna CustomerID como chave primária, enquanto a tabela Account (conta) contém apenas duas colunas, a coluna AccountID, é chave primária enquanto a chave primária da coluna Customer (cliente) também é a chave estrangeira da coluna Account (conta), logo após o período de transição (refatoração) a coluna Balance (balanço) é deletada da coluna Customer (cliente) e migrada a coluna Account (conta).

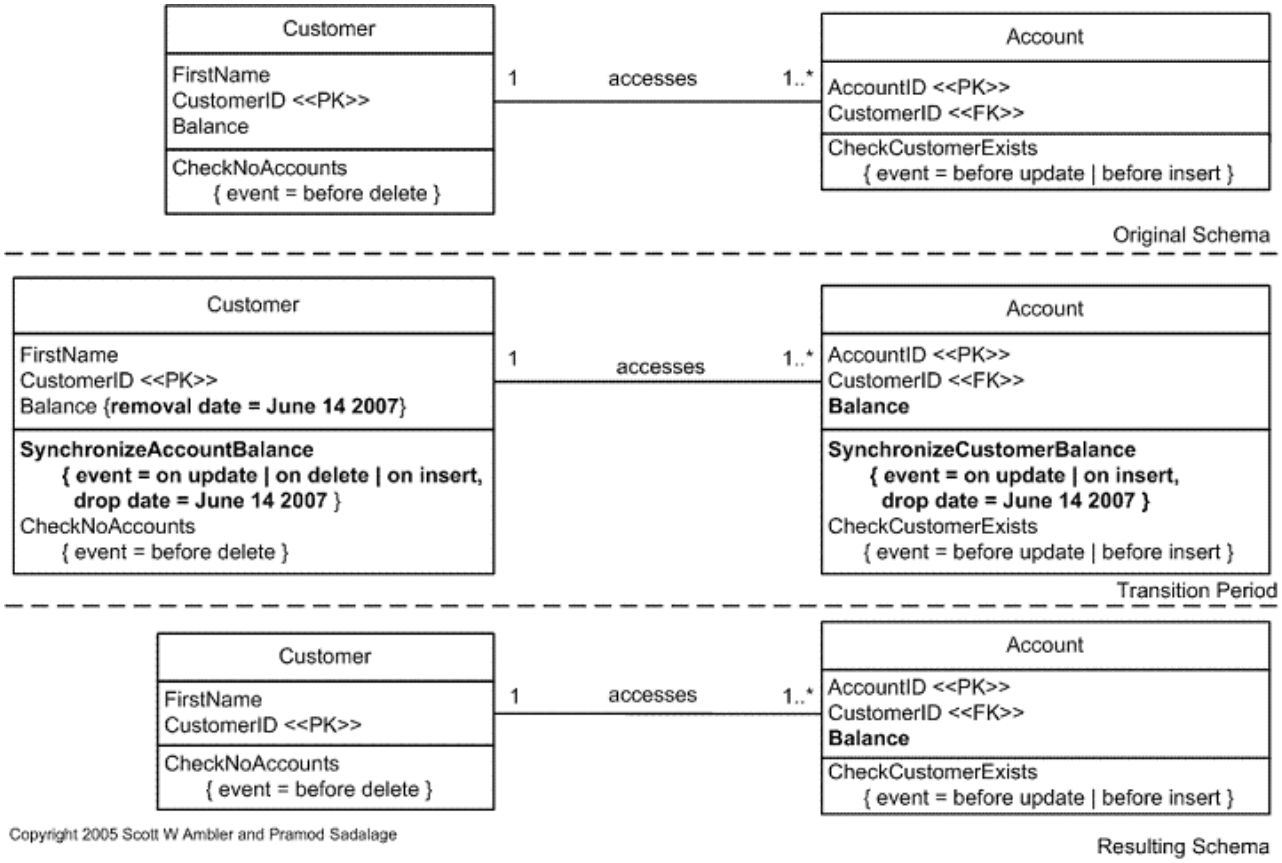

Figura 17. Mover Coluna

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Renomear Coluna

No exemplo da figura 18 renomear coluna, no esquema original tem-se a tabela Customer (cliente), que contém duas colunas, sendo que a coluna Customer ID (id cliente) está denominada como chave primária, logo após o período de transição (refatoração) a coluna Fname abreviação de (primeiro nome) é renomeada, e o esquema resultante da tabela Customer (cliente) é a coluna Customer ID (id cliente) continuando como a chave primária da tabela e a coluna First Name (primeiro nome).

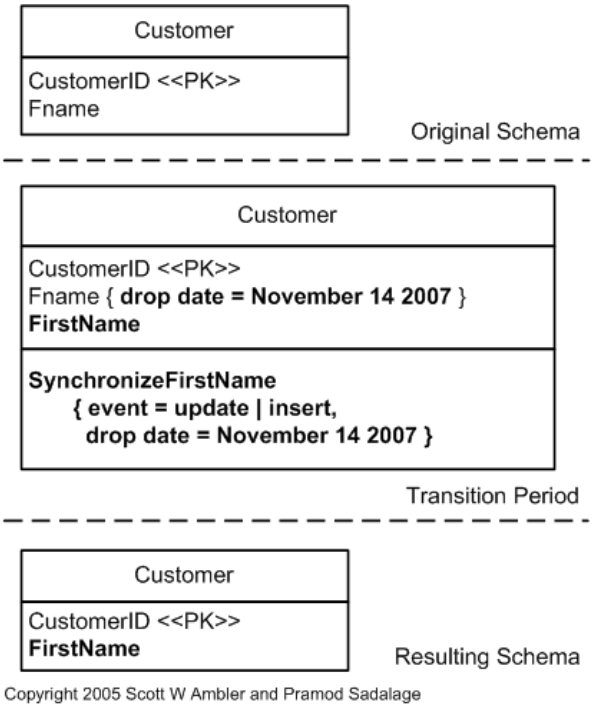

Figura 18. Renomear Coluna

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Renomear Tabela

No exemplo da figura 19 renomear tabela, no esquema original nota-se a Cust_TB_Prod como está ilustrado, logo após o período de transição (refatoração) foi renomeado o nome da tabela, que era apenas uma abreviação, passou para o nome mais concreto, para melhor entender o objetivo da tabela em questão.

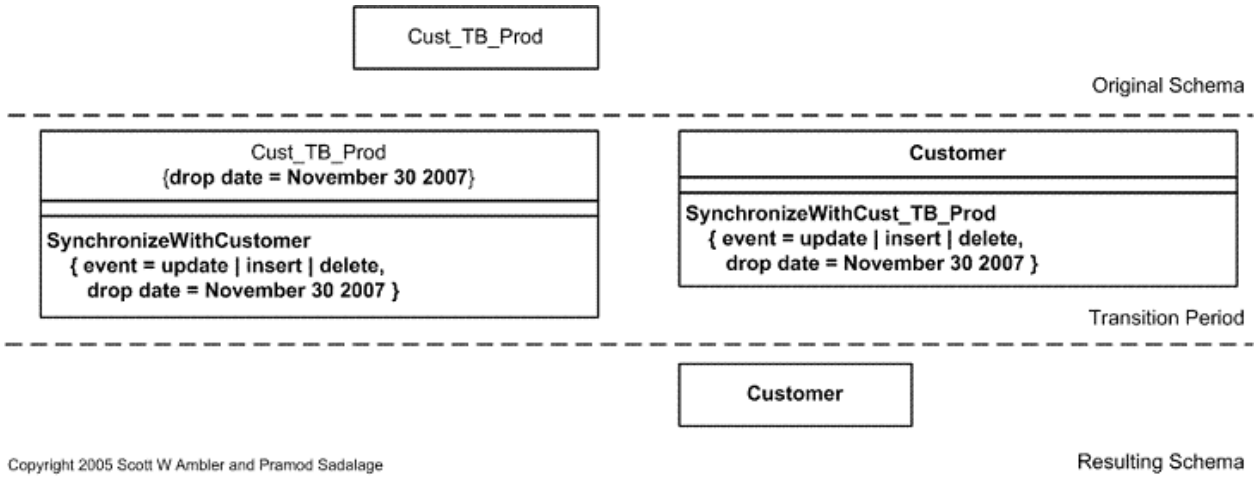

Figura 19. Renomear Tabela

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Renomear Visão

No exemplo da figura 20 renomear visão, no esquema original tem-se uma view (visão) denominada como CustOrds, logo após o período de transição (refatoração) a view (visão) CustOrds é renomeada para Customer Orders que significa em português pedidos de clientes, que explica melhor o propósito da visão.

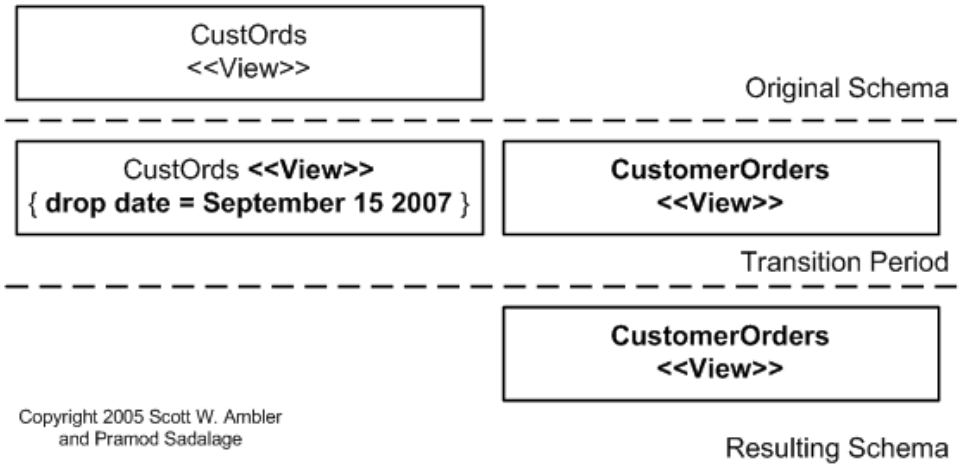

Figura 20. Renomear Visão

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Trocar Coluna

No exemplo da figura 21 trocar coluna, tem-se a tabela Customer (cliente) com quatro colunas, sendo que uma dessas a coluna Customer POID está denominada como chave primária, logo após o período de transição (refatoração) é substituída uma coluna não chave por uma coluna que pode ser uma chave candidata, contextualizada como Customer ID (id cliente) do tipo char com no máximo 12 caracteres.

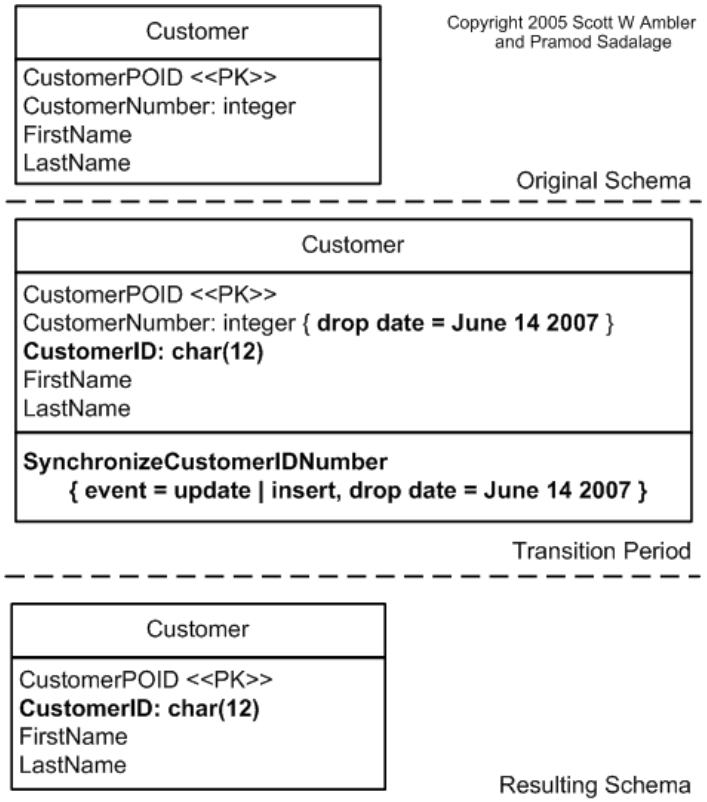

Figura 21. Trocar Coluna

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Trocar coluna complexa por tabela

No exemplo da figura 22 trocar coluna complexa por tabela, no esquema original tem-se a tabela denominada Customer (cliente), que possui quatro colunas, sendo que a coluna Customer Poid está denominada como chave primária, logo após o período de transição (refatoração), é adicionada no esquema uma nova tabela, denominada de Customer Adress (endereço do cliente), com a chave primária da tabela Customer (cliente), como chave estrangeira e mais quatro colunas, com o relacionamento de um para um, para a mesma.

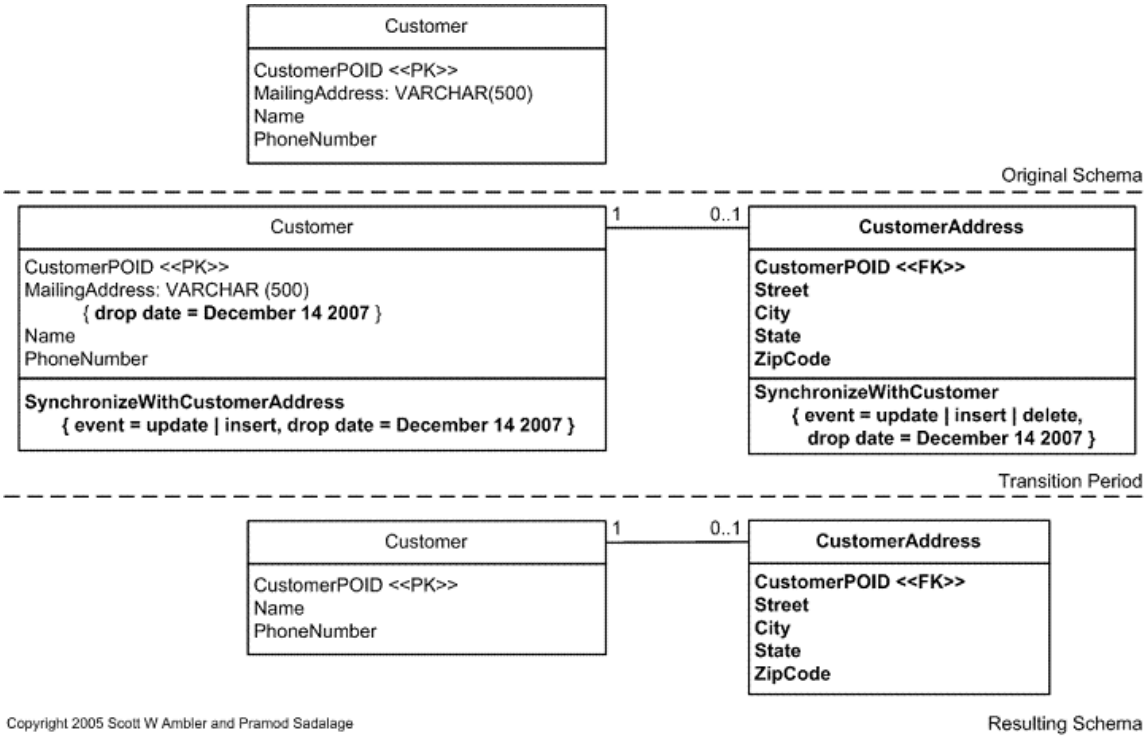

Figura 22. Trocar coluna complexa por tabela

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Trocar um para muitos por tabela associativa

No exemplo da figura 23 trocar um para muitos por tabela associativa, no esquema original tem-se duas tabelas Customer (cliente) e a tabela Policy (política) como é ilustrado na figura abaixo, as duas tabelas contém uma associação entre elas de um para muitos, a tabela Customer (cliente) contém duas colunas uma delas a coluna Customer POID é denominada chave primária, enquanto a tabela Policy (política) contém três colunas, e uma delas PolicyID está denominada como chave primária, e a chave primária da tabela Customer, é a chave estrangeira da tabela Policy, logo após o período de transição (refatoração) é criada uma nova tabela associativa entre as duas tabelas.

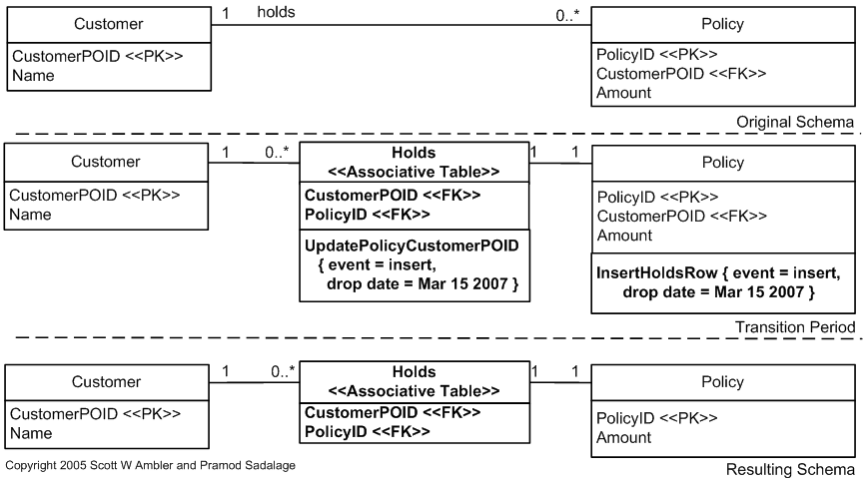

Figura 23. Trocar um para muitos por tabela associativa

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Trocar chave de identificação por chave natural

No exemplo da figura 24 trocar chave de identificação por chave natural, no esquema original têm-se duas tabelas Address (endereço) e State (estado), a tabela Address (endereço) contém cinco colunas, duas dessas cinco colunas são chaves estrangeiras que são: a coluna Country Code e a coluna State Poid, enquanto a segunda tabela do esquema, a tabela State (estado) contém três colunas, uma delas a coluna State POID é a chave primária, logo após o período de transição (refatoração) a chave substituta é substituída com uma chave natural existente.

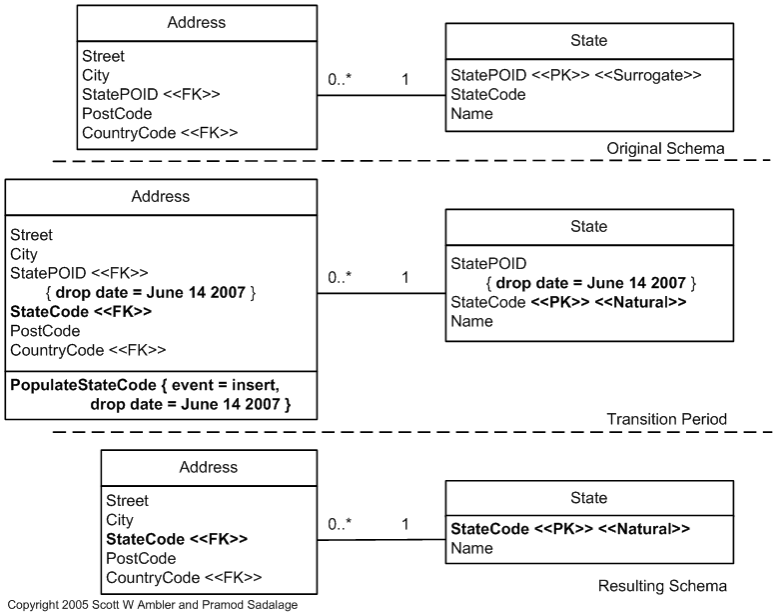

Figura 24. Trocar chave de identificação por chave Natural

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Dividir Coluna

No exemplo da figura 25 dividir coluna, no esquema original tem-se a tabela chamada Customer (cliente) com três colunas, sendo que uma delas a coluna Customer ID (id cliente) está denominada como chave primária, logo após o período de transição (refatoração) a coluna Name (nome) foi dividida em mais colunas.

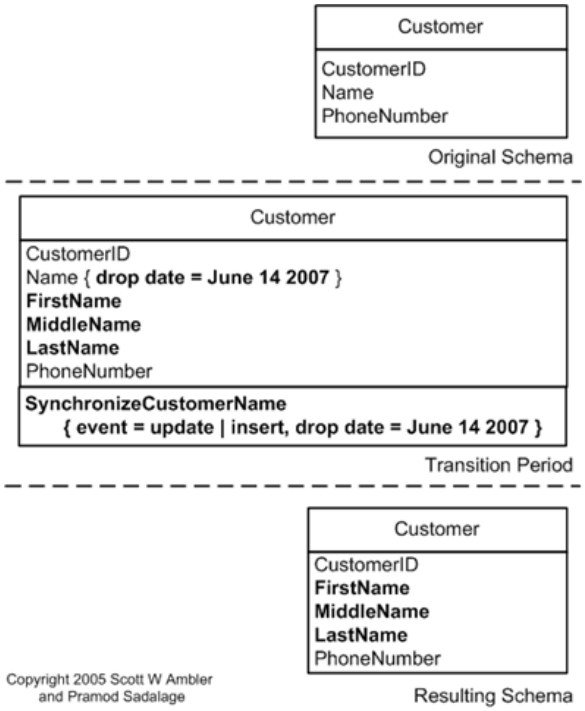

Figura 25. Dividir Coluna

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

- Dividir Tabela

No exemplo da figura 26 dividir tabela, no esquema original tem-se a tabela chamada Addreess (endereço) com cinco colunas, e uma delas, a coluna Address ID (id endereço) é a chave primária, logo após o período de transição (refatoração), a tabela Address é verticalmente dividida em duas, Address (endereço) e State (estado), com o relacionamento de um para um e a chave primária da tabela State (estado) é a chave estrangeira da tabela Address (endereço), que agora contém quatro colunas e não cinco colunas como no esquema original, enquanto a nova tabela State (estado) contém duas colunas sendo uma dessas a coluna State Code (código do estado) é denominada como chave primária.

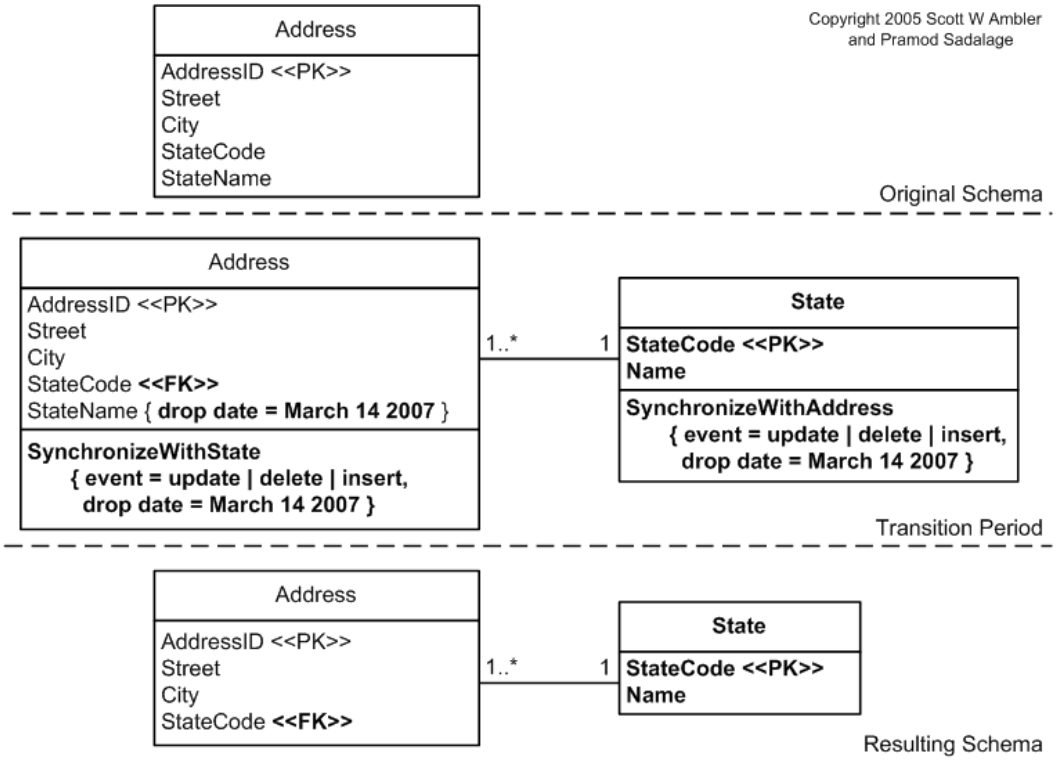

Figura 26. Dividir Tabela

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogStructural.html

4.2 Refatoração de Qualidade dos Dados Exemplos

Como explicado na seção 3.2.2 do capítulo 3, as refatorações de qualidade de dados têm como objetivo aumentar as validações e diminuir as inconsistências do banco (DOMINGUES, 2014).

A refatoração de banco de dados, qualidade de dados é uma mudança que melhora e / ou garante a consistência e uso dos valores armazenados no banco de dados, de modo a melhorar o seu projeto de banco de dados sem alterar sua semântica (AGILEDATA, 2015).

Serão apresentados a seguir exemplos sobre a refatoração de qualidade de dados.

- Adicionar tabela descritiva

No exemplo da figura 27 adicionar tabela descritiva, no esquema original tem-se a tabela chamada Address (endereço), o esquema resultante é introdução de uma nova tabela descritiva chamada State (estado).

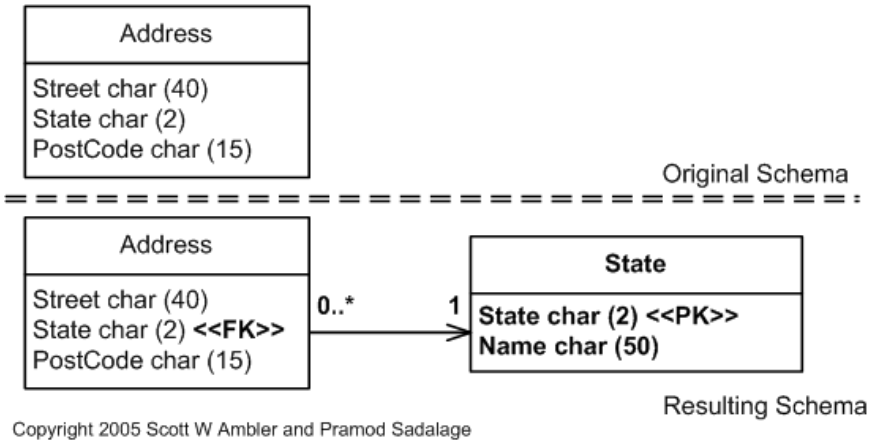

Figura 27. Adicionar tabela descritiva

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogDataQuality.html

- Aplicar código padrão

No exemplo da figura 28 aplicação de código padrão, aplicar um conjunto padrão de valores de código a uma única coluna, para garantir que a mesma está em conformidade com os valores das colunas semelhantes armazenados em outro lugar no banco de dados (AGILEDATA, 2015).

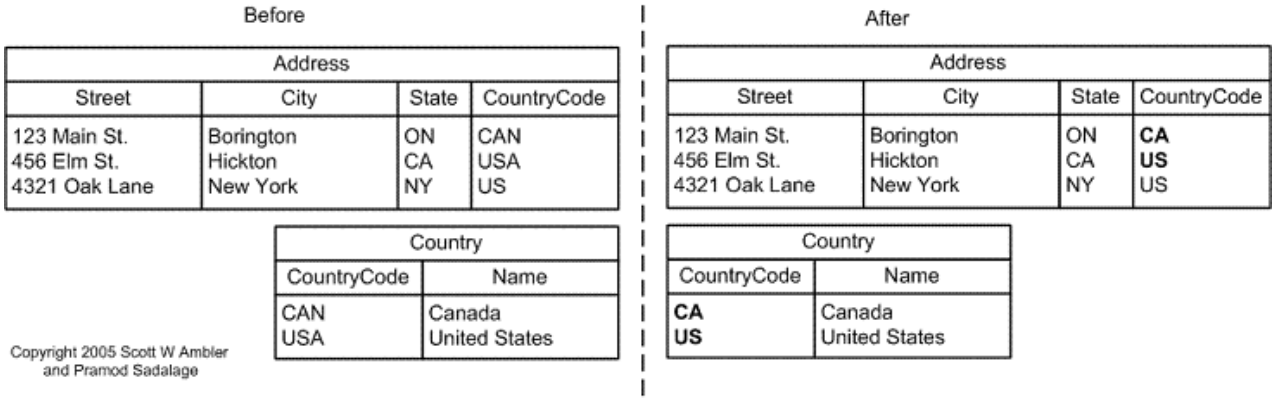

Figura 28. Aplicar código padrão

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogDataQuality.html

- Aplicar tipo padrão

No exemplo da figura 29 aplicar tipo padrão, no esquema original têm-se três tabelas como demonstra a figura, a tabela Customer (cliente), a tabela Branch (filial), e a tabela Employee (empregado), logo após o período de transição (refatoração), são asseguradas que as colunas semelhantes em outras tabelas do banco de dados recebam um tipo de dados único.

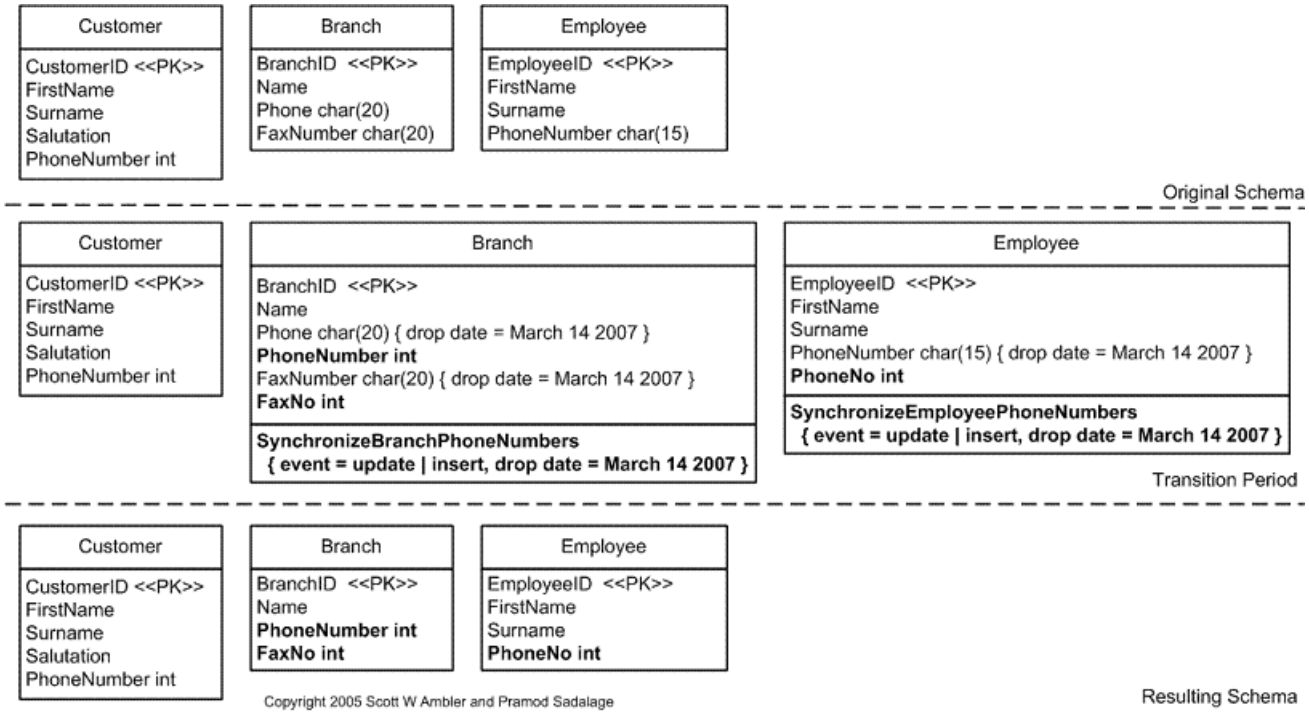

Figura 29. Aplicar tipo padrão

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogDataQuality.html

- Consolidar estratégias de chaves

No exemplo da figura 30 consolidar estratégias de chaves, no esquema original têm-se três tabelas que são: Policy (política), Policy Notes (notas de política), e a tabela Policy Due Diligence (política de diligência devida), logo após o período de depreciação (refatoração), a coluna Policy Number da tabela Policy Notes (notas políticas) é renomeada ou excluída e adicionada uma coluna Policy OID e denominada como chave primária da tabela no esquema resultante.

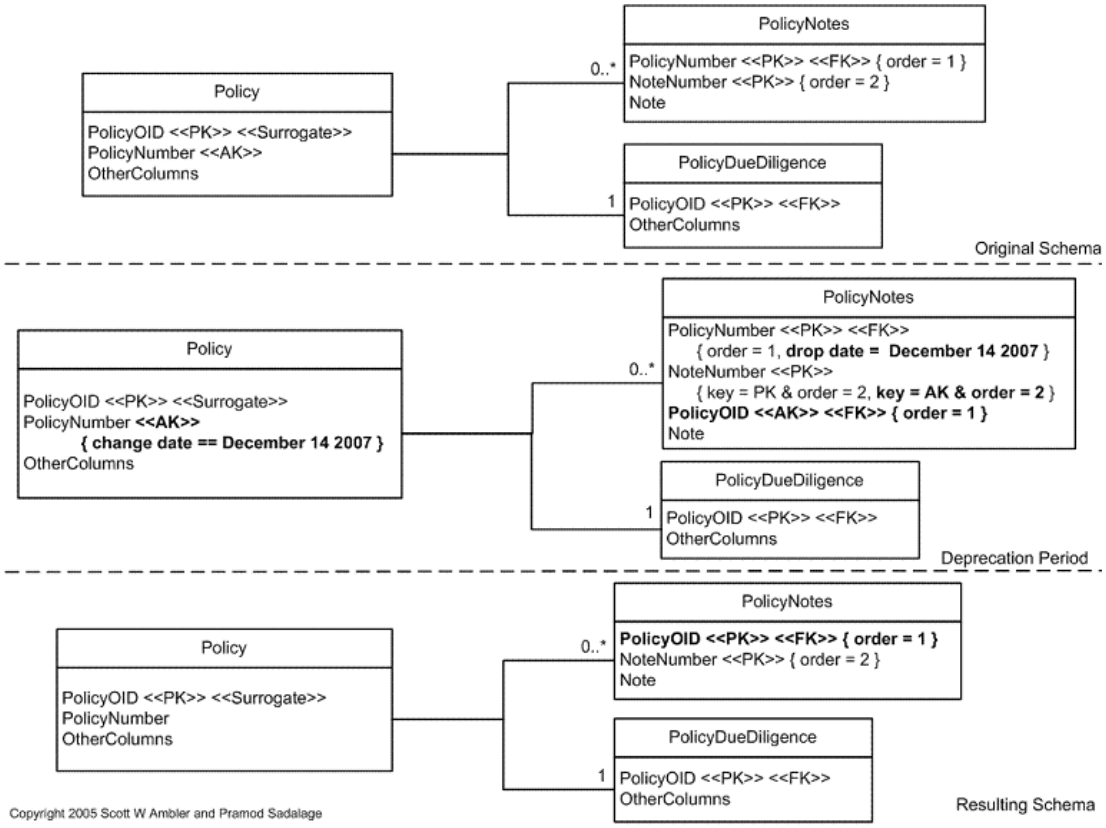

Figura 30. Consolidar Estratégias de Chaves

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogDataQuality.html

- Remover restrição de coluna

No exemplo da figura 31 remover restrição de coluna, no esquema original tem-se a tabela Customer (cliente) com quatro colunas, sendo que a coluna Customer ID (id cliente) é denominada como chave primária, o esquema resultante da tabela Customer (cliente) é a remoção da restrição da coluna Credit Limit (limite de crédito).

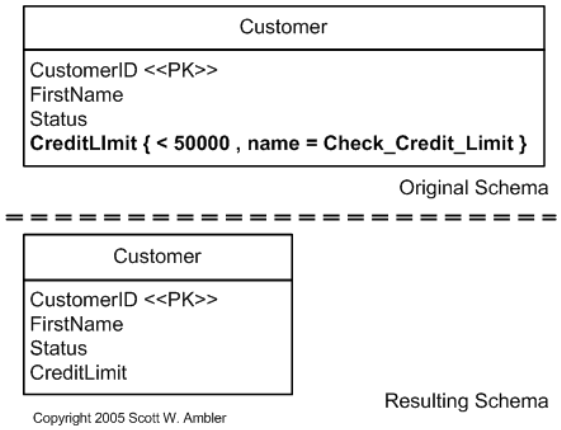

Figura 31. Remover restrição de coluna

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogDataQuality.html

- Remover valor padrão

No exemplo da figura 32 remover valor padrão, no esquema original tem-se a tabela Customer (cliente) com três colunas, sendo que a coluna Customer ID (id cliente) é denominada como chave primária, logo após o período de transição (refatoração) a coluna Status (condição) é removido, o valor padrão que é fornecido pelo banco de dados.

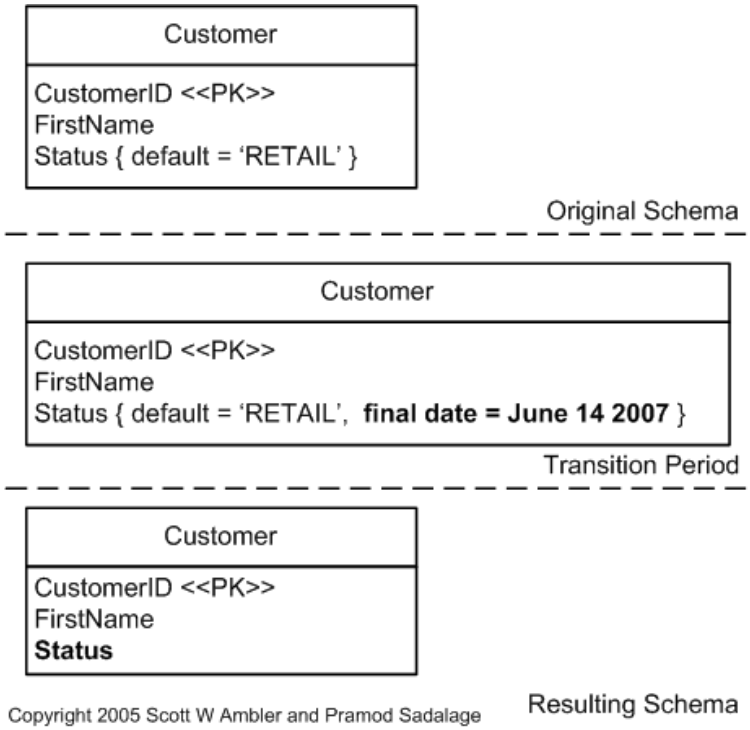

Figura 32. Remover valor padrão

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogDataQuality.html

- Remover restrição de não nulo

No exemplo da figura 33 remover restrição de não nulo, no esquema original tem-se a tabela City Lookup com três colunas, logo após o período de transição (refatoração) a coluna State Code (código do estado) que é anulável foi alterada para aceitar valores nulos.

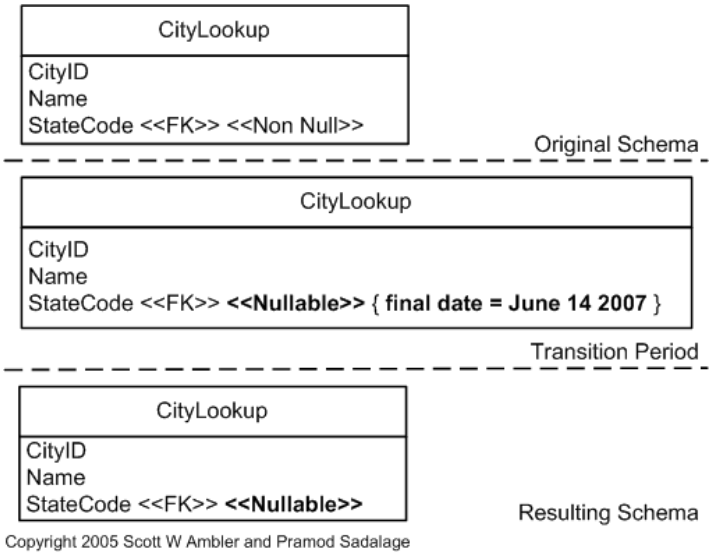

Figura 33. Remover restrição de não nulo

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogDataQuality.html

- Introduzir restrição de coluna

No exemplo da figura 34 introduzir restrição de coluna, no esquema original tem-se a tabela Customer (cliente) com quatro colunas, sendo que a coluna Customer ID (id cliente) é denominada como chave primária, logo após o período de transição (refatoração) é introduzida uma restrição na coluna Credit Limit (limite de crédito).

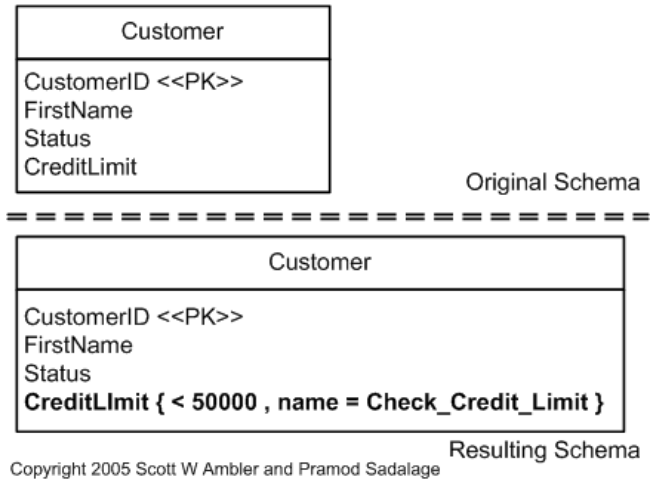

Figura 34. Introduzir restrição de coluna

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogDataQuality.html

- Introduzir formato comum

No exemplo da figura 35 introduzir formato comum, no esquema original tem-se a coluna Phone Number (número de telefone), nesta coluna estão contidos alguns registros,

logo após o período de transição (refatoração) é padronizado o formato para os valores da coluna na tabela existente no banco de dados.

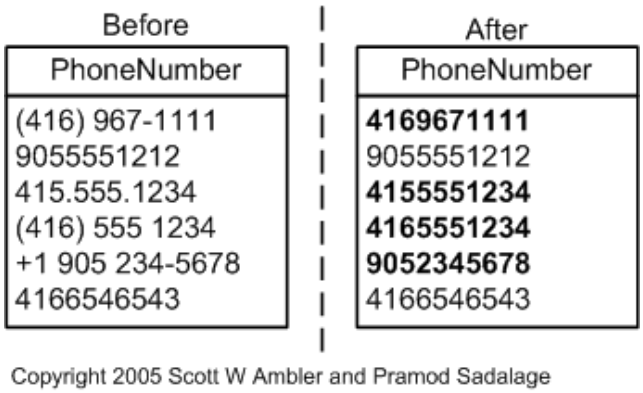

Figura 35. Introduzir formato comum

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogDataQuality.html

- Introduzir valor padrão

No exemplo da figura 36 introduzir valor padrão, no esquema original tem-se a tabela Customer (cliente) com três colunas, sendo que a coluna Customer ID (id cliente) é denominada como chave primária, deixando o banco fornecer um valor padrão para a coluna.

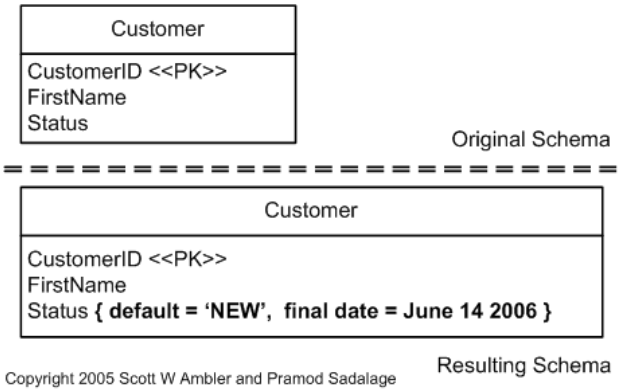

Figura 36. Introduzir Valor padrão

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogDataQuality.html

- Alterar coluna para não nula

No exemplo da figura 37 alterar coluna para não nula, no esquema original tem-se a tabela Customer (cliente) com três colunas, sendo que a coluna Customer ID (id cliente) é denominada como chave primária, o esquema resultante é a alteração da coluna First Name (primeiro nome) para não receber valores nulos.

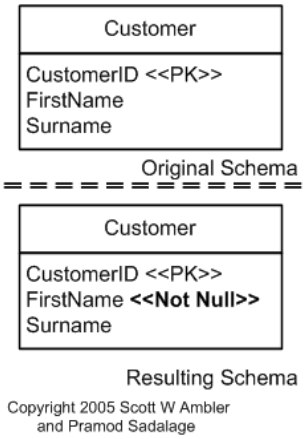

Figura 37. Alterar Coluna para não nula

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogDataQuality.html

- Mover dados

No exemplo da figura 38 mover dados, no esquema original tem-se duas tabelas, a tabela Customer (cliente), e a tabela Customer Status History (histórico de condições do cliente), ambas as tabelas possuem três colunas, foi movido os registros da coluna Customer Status (condição do cliente) da tabela Customer (cliente) para a coluna Customer Status (condição do cliente) da tabela Customer Status History (histórico de condições do cliente).

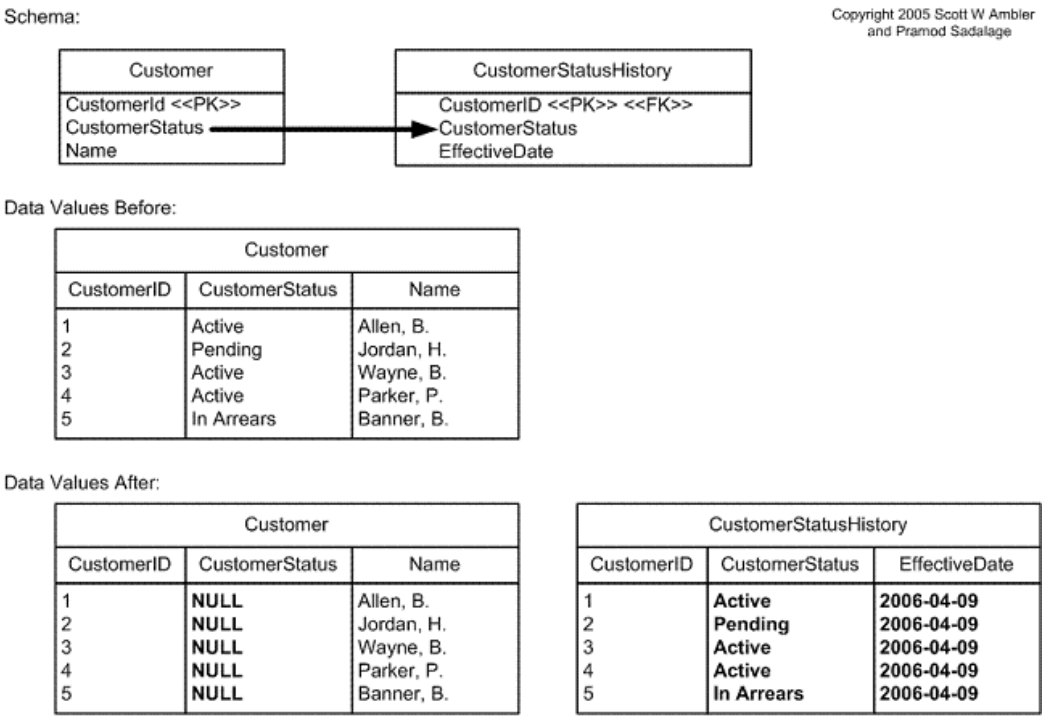

Figura 38. Mover dados

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogDataQuality.html

- Substituir tipo de código por propriedades sinalizadoras

No exemplo da figura 39 substituir tipo de código por propriedades sinalizadoras, no esquema original tem-se a tabela Account (conta) com três colunas, sendo que uma dessas a coluna Account ID (id conta) é denominada como chave primária, logo após o período de transição (refatoração) é substituída a coluna de código Account Type (tipo de conta) por colunas de bandeiras booleanas.

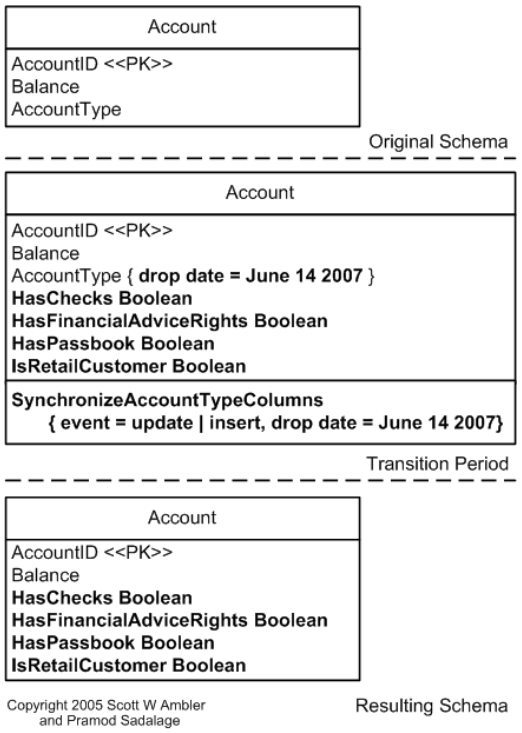

Figura 39. Substituir tipo de código por propriedades sinalizadoras

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogDataQuality.html

4.3 Refatoração de Integridade Referencial Exemplos

Como contextualizado na seção 3.2.3 do capítulo 3, as refatorações de integridade referencial têm como objetivo manter consistentes as referências entre tabelas, incluindo ou removendo regras, armazenando históricos ou implementando lógicas de aplicação no banco de dados (DOMINGUES, 2014).

A refatoração de banco de dados, integridade referencial é uma alteração que garante que uma linha referenciada existe dentro de outra tabela e /ou que garante que uma linha que não é mais necessário, é removida de forma adequada de modo a melhorar o seu projeto de banco de dados sem alterar sua semântica (AGILEDATA, 2015).

Serão apresentados exemplos sobre as refatorações de integridade referencial.

- Adicionar restrição de chave estrangeira

No exemplo da figura 40 adicionar restrição de chave estrangeira, no esquema original têm-se duas tabelas Account (conta) e Account Status (situação da conta), a tabela Account (conta) possui três colunas, sendo que a coluna Account ID (id conta) é denominada como chave primária, enquanto a tabela Account Status (situação da conta) possui duas colunas, sendo que a coluna Status Code (código de condição) é denominada como chave primária, as duas tabelas estão em um relacionamento de zero, no esquema resultante é adicionado na coluna Status Code da tabela Account (conta) uma condição de chave estrangeira.

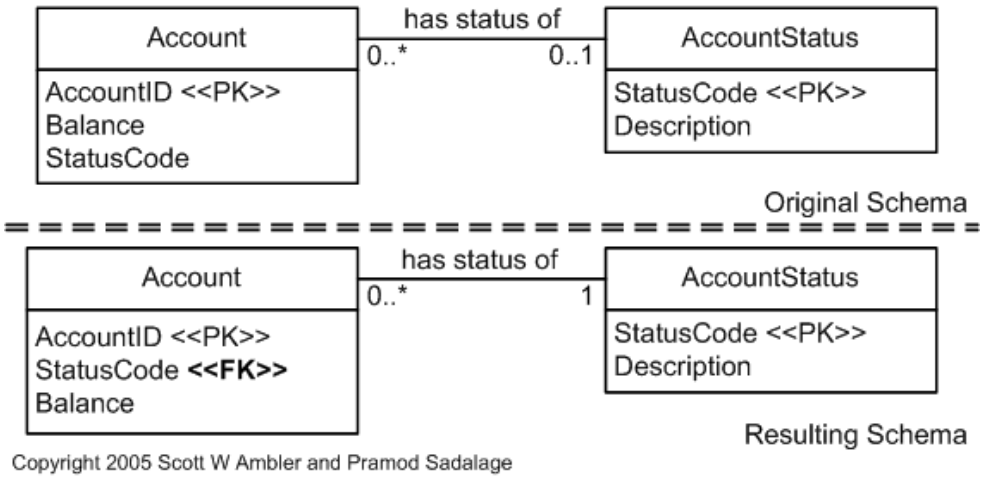

Figura 40. Adicionar restrição de chave estrangeira

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogReferentialIntegrity.html

- Adicionar trigger para coluna calculada

No exemplo da figura 41 adicionar trigger para coluna calculada, no esquema original têm-se as tabelas Customer (cliente), Account (conta) com o relacionamento de um para muitos e uma associação com mais duas tabelas, logo após a refatoração é introduzido um novo gatilho para a tabela Customer assim atualizando o valor da tabela.

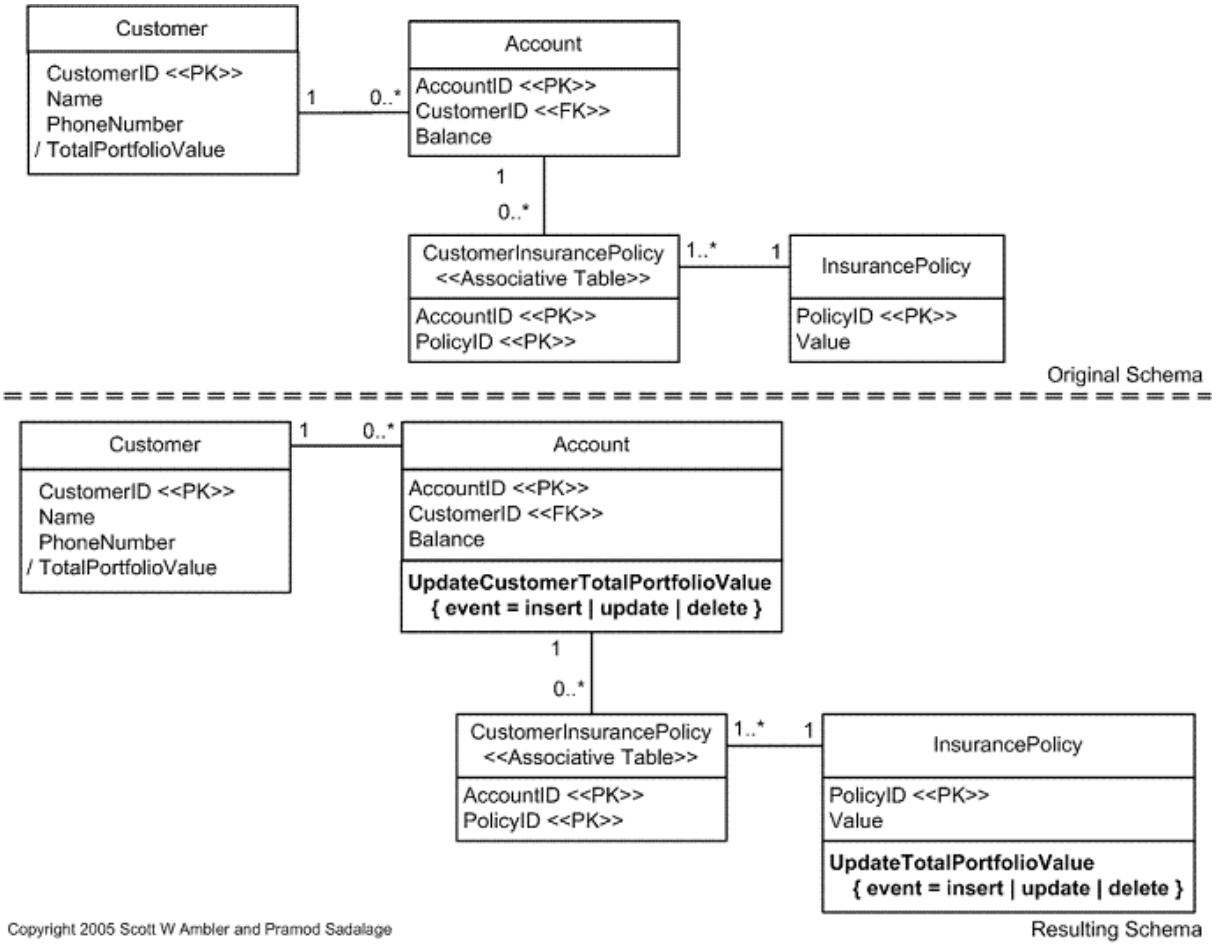

Figura 41. Adicionar trigger para coluna calculada

Fonte:
http://www.agiledata.org/essays/databaseRefactoringCatalogReferentialIntegrity.html

- Remover restrição de chave estrangeira

No exemplo da figura 42 remover restrição de chave estrangeira, no esquema original têm-se duas tabelas Account (conta), e Account Status (condição da conta) ambas as tabelas estão em um relacionamento muitos para um, a tabela Account (conta) possui três colunas, sendo que a coluna Account ID (id conta) é denominada como chave primária, e a coluna Status Code (código de estatus) como chave estrangeira, a coluna Account Status (condição da conta) possui duas colunas, a coluna Status Code (código de status) é denominada como chave primária, no esquema resultante na tabela Account (conta) é removida a restrição de chave estrangeira da coluna Status Code (código de status).

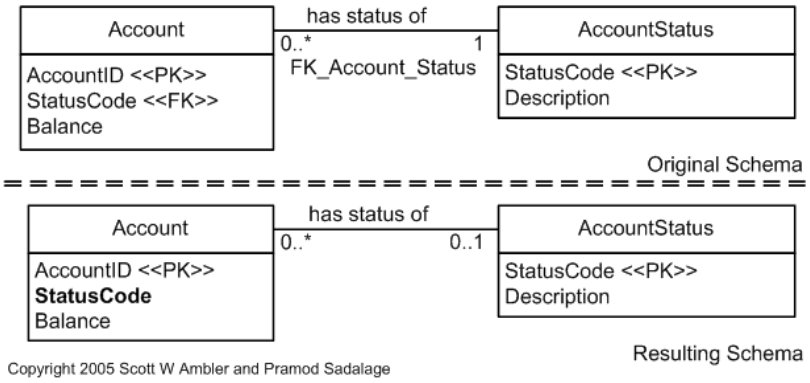

Figura 42. Remover restrição da chave estrangeira

Fonte:
http://www.agiledata.org/essays/databaseRefactoringCatalogReferentialIntegrity.html

- Introduzir exclusão em cascata

No exemplo da figura 43 introduzir exclusão em cascata, no esquema original têm-se três tabelas, a tabela Policy (política), Policy Notes (notas políticas), e Claim (reivindicação) ambas estão em uma associação de um para muitos, no esquema resultante certifica-se se os registros são excluídos no banco.

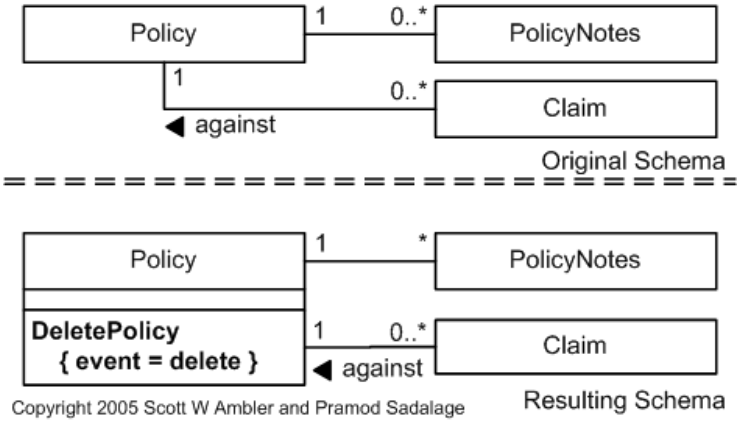

Figura 43. Introduzir exclusão em cascata

Fonte:
http://www.agiledata.org/essays/databaseRefactoringCatalogReferentialIntegrity.html

- Introduzir exclusão física

No exemplo da figura 44 introduzir exclusão física, no esquema original tem-se a tabela Customer (cliente) que possui quatro colunas, sendo que a coluna Customer ID (id cliente) está denominada como chave primária da tabela em questão, logo após o período de transição (refatoração) uma coluna existente foi excluída e o esquema resultante é a tabela Customer (cliente) com três colunas.

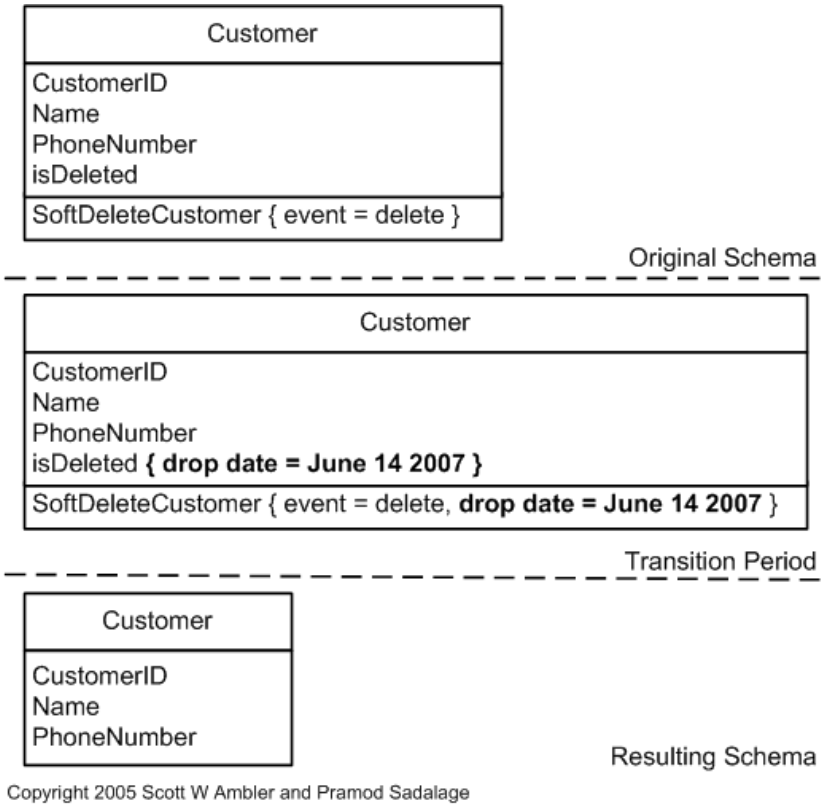

Figura 44. Introduzir exclusão física

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogReferentialIntegrity.html

- Introduzir exclusão lógica

No exemplo da figura 45 introduzir exclusão lógica, no esquema original tem-se a tabela Customer (cliente) que possui três colunas, sendo que a coluna Customer ID (id cliente) está denominada como chave primária, logo após o período de transição (refatoração) foi introduzida um sinalizador para indicar que a linha foi realmente excluída.

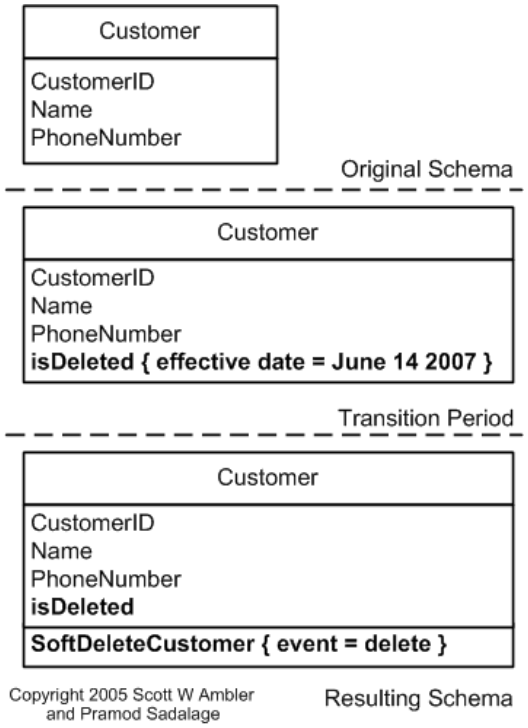

Figura 45. Introduzir exclusão lógica

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogReferentialIntegrity.html

- Introduzir trigger para histórico

No exemplo da figura 46 introduzir trigger para histórico, no esquema original têm-se duas tabelas, a tabela Customer (cliente), e a tabela Customer History (histórico do cliente) ambas as tabelas estão em um relacionamento de um para muitos, o esquema resultante do banco é introduzido um trigger histórico na tabela para a alteração da mesma.

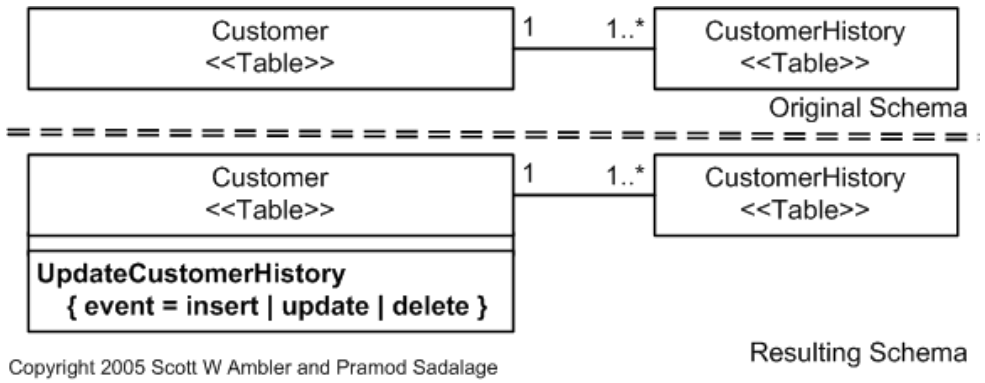

Figura 46. Introduzir trigger para histórico

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogReferentialIntegrity.html

4.4 Refatoração Arquitetural Exemplos

Como explicado na seção 3.2.4 do capítulo 3, o objetivo principal das refatorações arquiteturais é mover uma parte do código dos aplicativos para o banco de dados (lógica do banco de dados). Os objetivos secundários são organizar, padronizar, e melhorar o desempenho do banco de dados (DOMINGUES, 2014).

Uma refatoração de banco de dados de arquitetura é uma mudança que melhora a maneira global em que programas externos interagem com um banco de dados, de modo a melhorar o seu projeto de banco de dados sem alterar sua semântica (AGILEDATA, 2015).

- Adicione métodos CRUD

No exemplo da figura 47 adicionar métodos crud, no esquema original tem-se o banco de dados com procedures, no esquema resultante foram adicionados métodos de crud ao banco para leitura, alteração, exclusão e criação de dados no banco de dados.

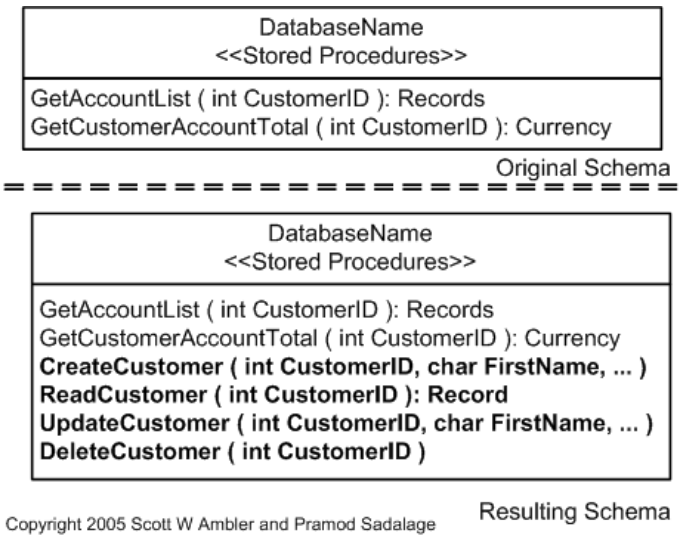

Figura 47. Adicione métodos CRUD

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogArchitectural.html

- Introduzir tabela espelho

No exemplo da figura 48 introduzir tabela espelho, tem-se a tabela Customer (cliente), no esquema resultante a tabela Customer (cliente) é copiada em outro banco de dados.

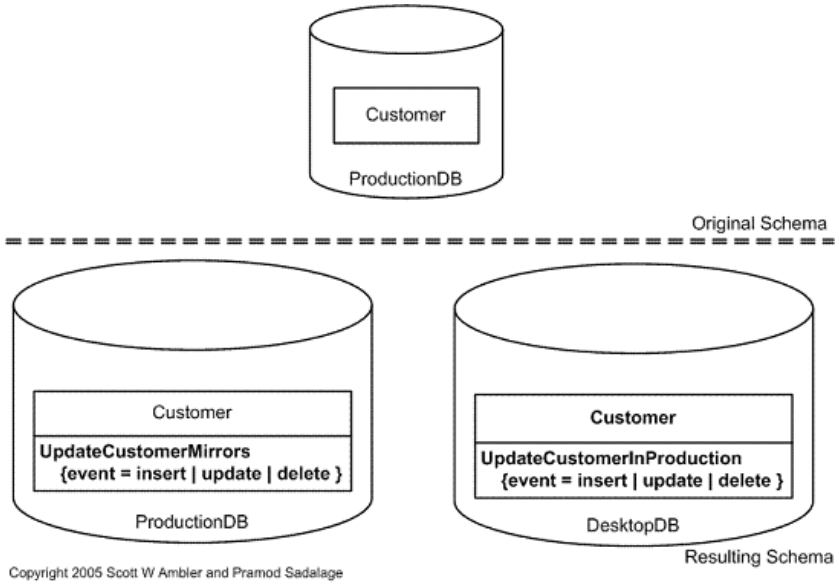

Figura 48. Introduzir tabela espelho

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogArchitectural.html

- Adicionar método de leitura

No exemplo da figura 49 adicionar método de leitura, no esquema original tem-se o banco de dados com procedures, no esquema resultante é introduzido um método para a recuperação dos dados do banco.

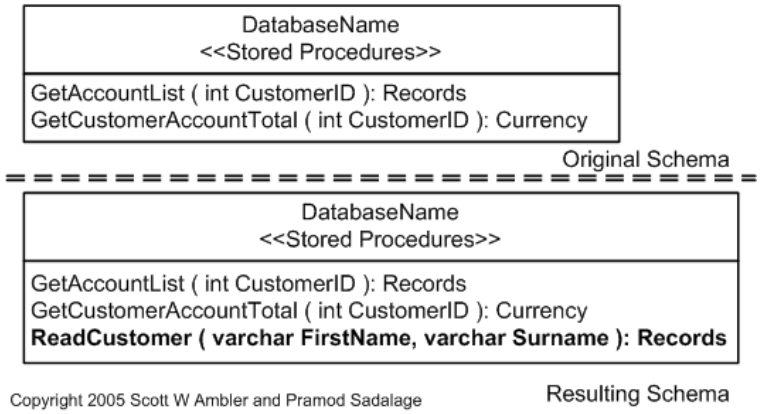

Figura 49. Adicionar método de leitura

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogArchitectural.html

- Encapsular tabela com uma visão

No exemplo da figura 50 encapsular tabela com uma visão, no esquema original tem-se a tabela Customer (cliente), no esquema resultante é criada uma view (visão) de acesso para a tabela Customer (cliente).

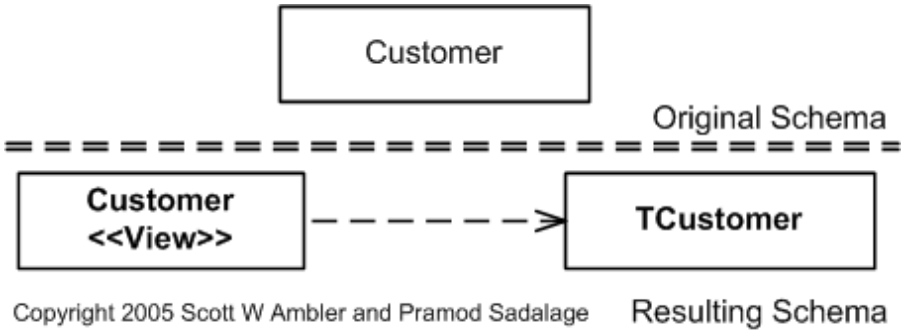

Figura 50. Encapsular tabela com uma visão

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogArchitectural.html

- Introduzir método para cálculo

No exemplo da figura 51 introduzir método para cálculo, no esquema original tem-se a tabela Customer (cliente) no banco de dados, no esquema resultante é a introdução de um método de cálculo que faz cálculos com os dos dados contidos do banco.

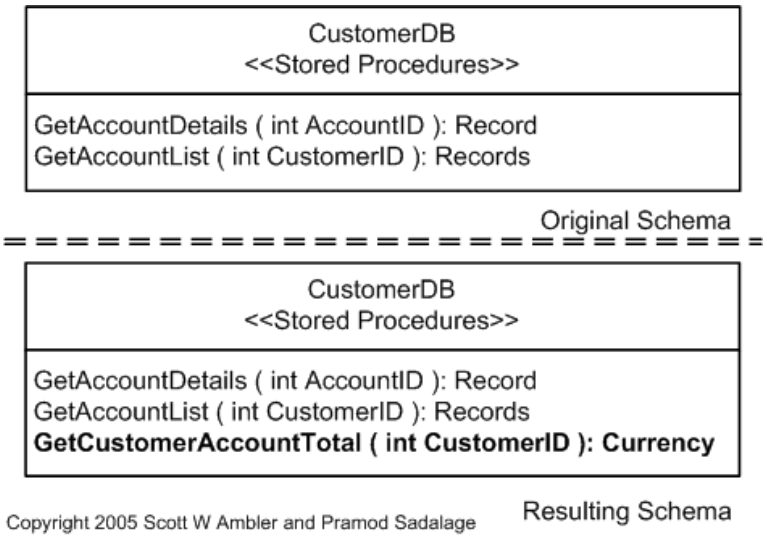

Figura 51. Introduzir método para cálculo

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogArchitectural.html

- Introduzir Índice

No exemplo da figura 52 introduzir índice, no esquema original tem-se a tabela Customer (cliente) que possui quatro colunas, sendo que a coluna Customer ID (id cliente) está denominada como chave primária, no esquema resultante é introduzido um novo índice a tabela Customer (cliente).

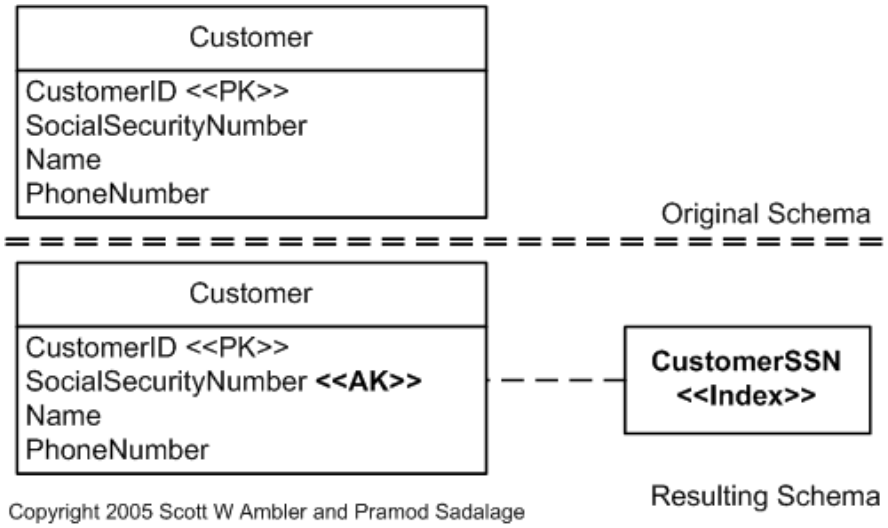

Figura 52. Introduzir Índice

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogArchitectural.html

- Introduzir somente tabela de leitura

No exemplo da figura 53 introduzir somente tabela de leitura, no esquema original têm-se três tabelas Customer (cliente), Account (conta) e Insurance (seguro), a tabela Customer (cliente) possui três colunas, sendo que a coluna Customer ID (id cliente), está denominada como chave primária, a tabela Account (conta) possui quatro colunas, sendo que a coluna Account ID (id conta) está denominada como chave primária, enquanto as duas colunas Customer ID (id cliente) está denominada como chave estrangeira, e a coluna Account Type (tipo de conta) também está denominada como chave estrangeira, a tabela Insurance (seguro) possui seis colunas, sendo que a coluna Policy ID (id política) está denominada como chave primária, enquanto a coluna Customer ID (id cliente) está denominada como chave estrangeira.

No esquema final foi introduzido um portfólio de armazenamento de dados, para leitura dos dados das tabelas existentes no banco de dados.

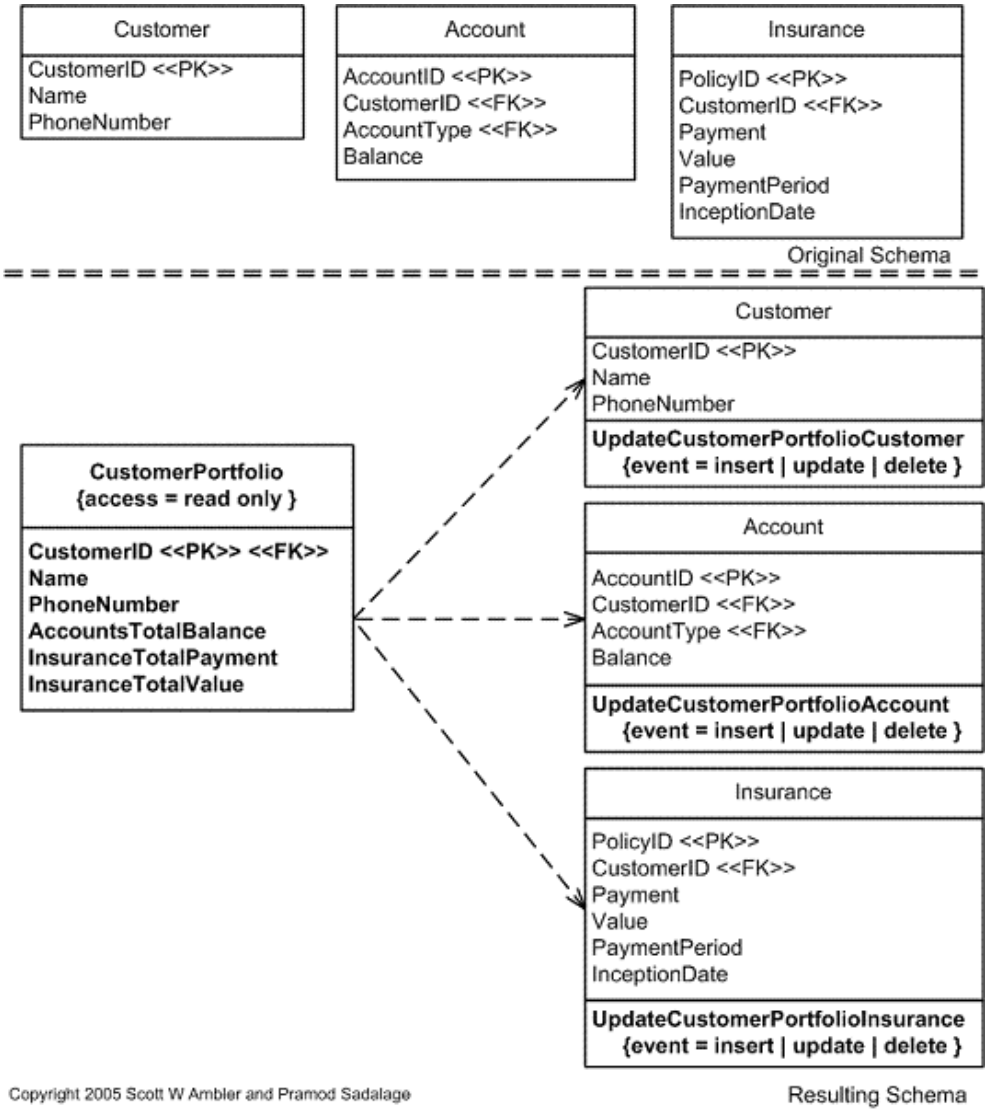

Figura 53. Introduzir somente tabela de leitura

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogArchitectural.html

- Migrar método para banco de dados

No exemplo da figura 54 migrar método para banco de dados, no esquema original têm-se algumas aplicações como CRM (gestão de relacionamento com o cliente), e Service, Policy Administration (política de administração), ligados ao banco de dados Customer (cliente), logo após o período de transição (refatoração) um método, ou procedimento é migrado do banco para umas aplicações do banco, a aplicação Policy Adminstration (política de administração) para a invocação da aplicação existente no banco de dados.

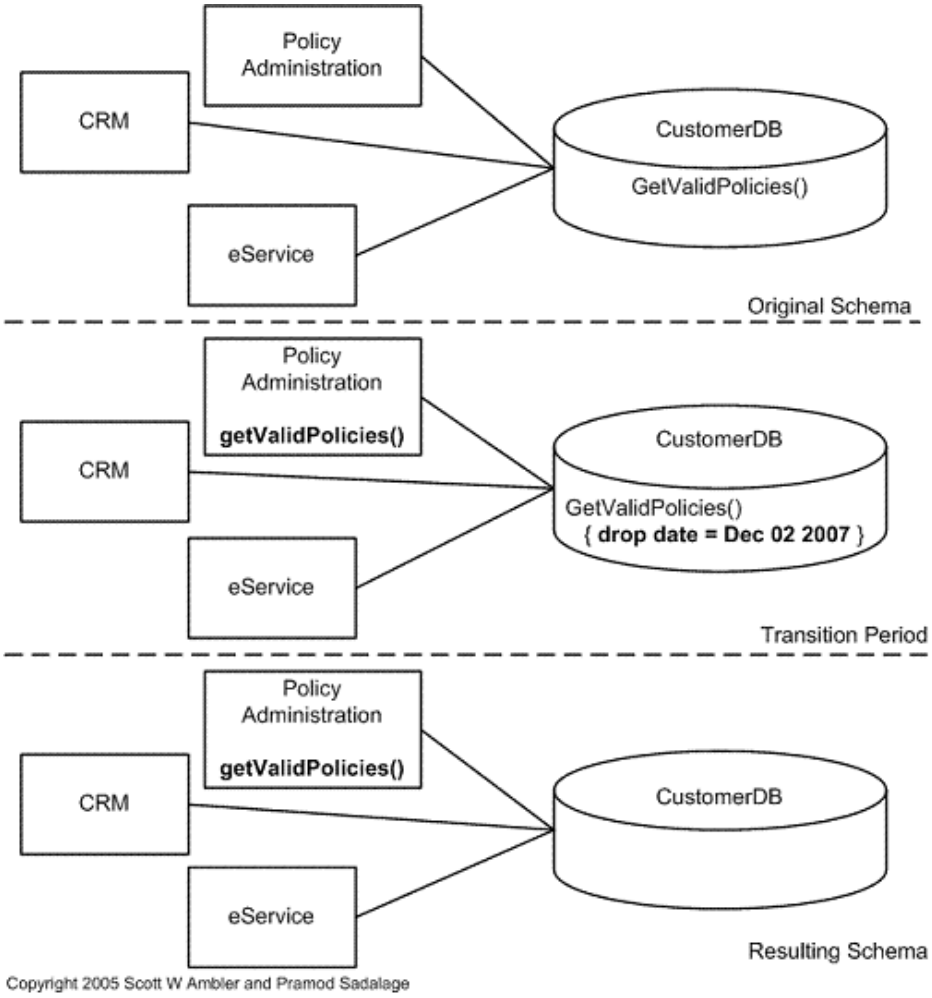

Figura 54. Migrar Método para banco de dados

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogArchitectural.html

- Migrar método de banco de dados

No exemplo da figura 55 migrar método de banco de dados, no esquema original têm-se novamente aplicações como: CRM (gestão de relacionamento com cliente), Policy Administration (política de administração) com o método de identificação de clientes, e a aplicação Customer Credit (crédito de cliente) com método de listagem de clientes como ilustra a figura abaixo, logo após o período de transição (refatoração) os métodos de identificação de clientes da aplicação Policy Administration (política de administração) e o método de listagens de clientes da aplicação Customer Credit (crédito do cliente) são migrados para o banco com a mesma lógica no banco de dados.

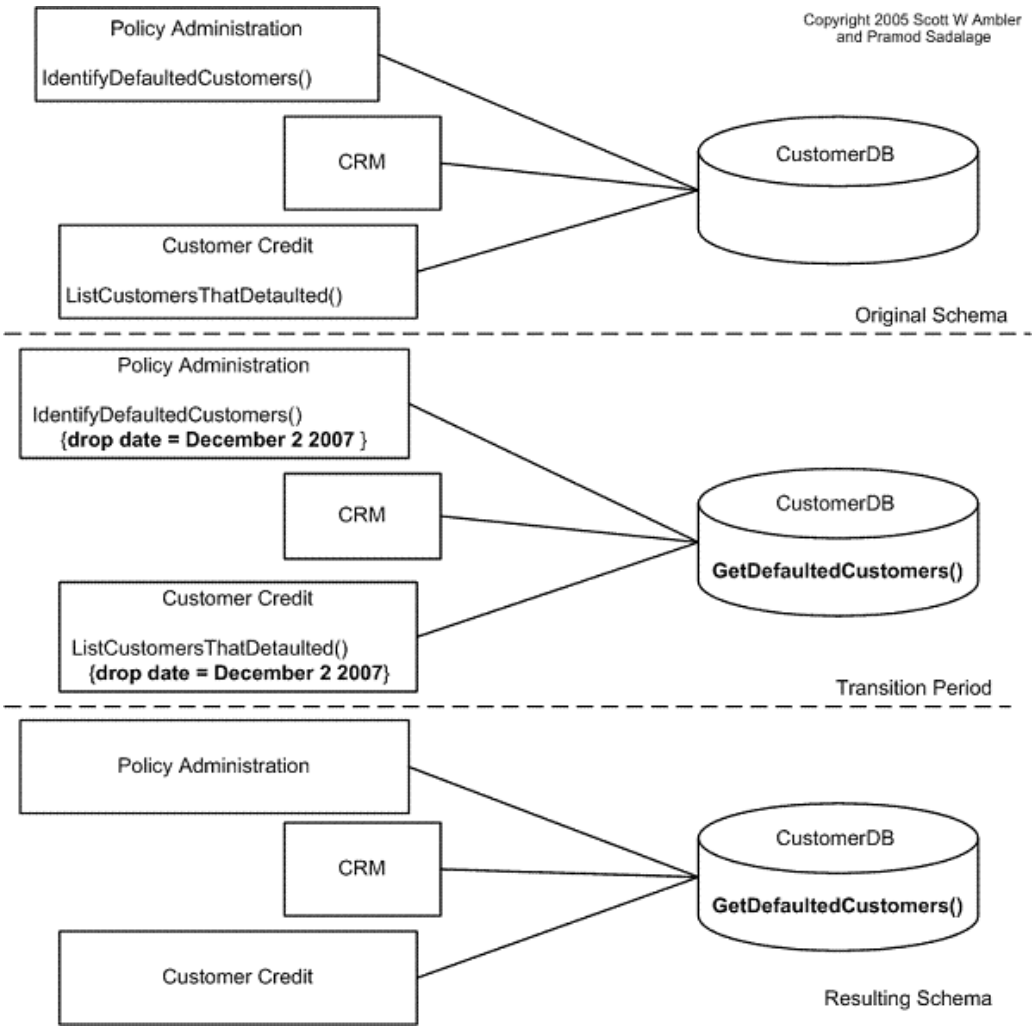

Figura 55. Migrar método de banco de dados

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogArchitectural.html

- Trocar método (s) por visão

No exemplo da figura 56 trocar método (s) por visão, no esquema original tem-se o banco de dados com seus métodos e procedures (procedimentos), logo após o período de transição (refatoração) o método Get Customer Account List (pegar a lista de conta dos clientes) foi substituído por uma view (visão) Customer Account List (listagem de conta do cliente).

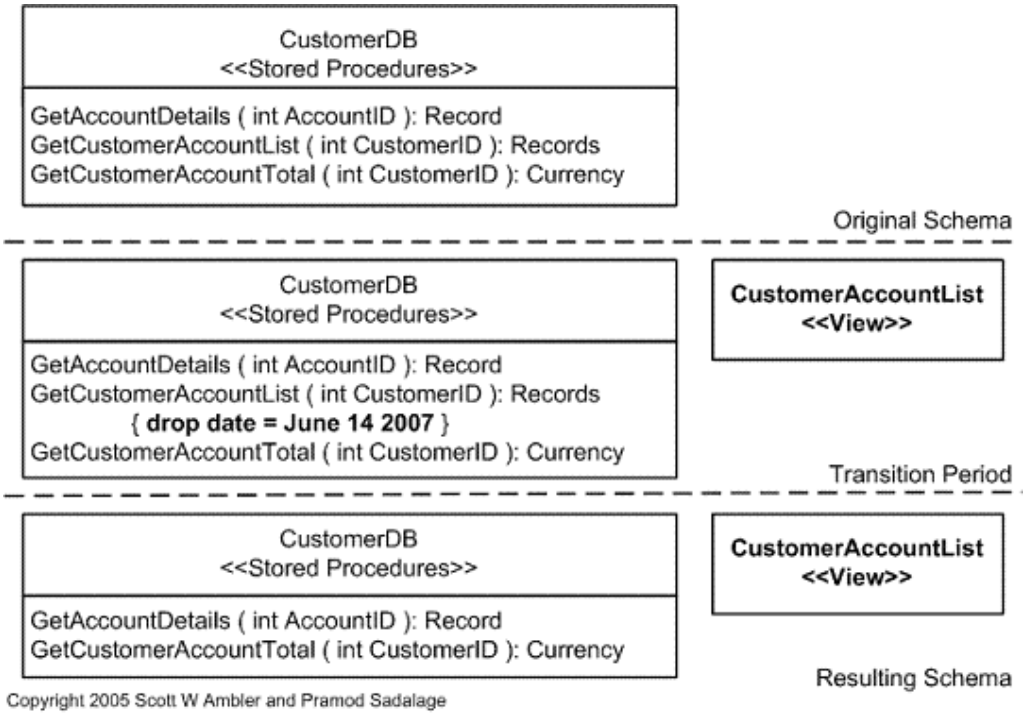

Figura 56. Trocar método (s) por visão

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogArchitectural.html

- Trocar visão por método (s)

No exemplo da figura 57 trocar visão por método (s), no esquema original tem-se o banco de dados com suas views (visões) e procedures (procedimentos), logo após o período de transição (refatoração) a view (visão) Customer Transactions History (histórico de transições do cliente) foi trocado por um método chamado Get Customer Transactions (obter transações do cliente).

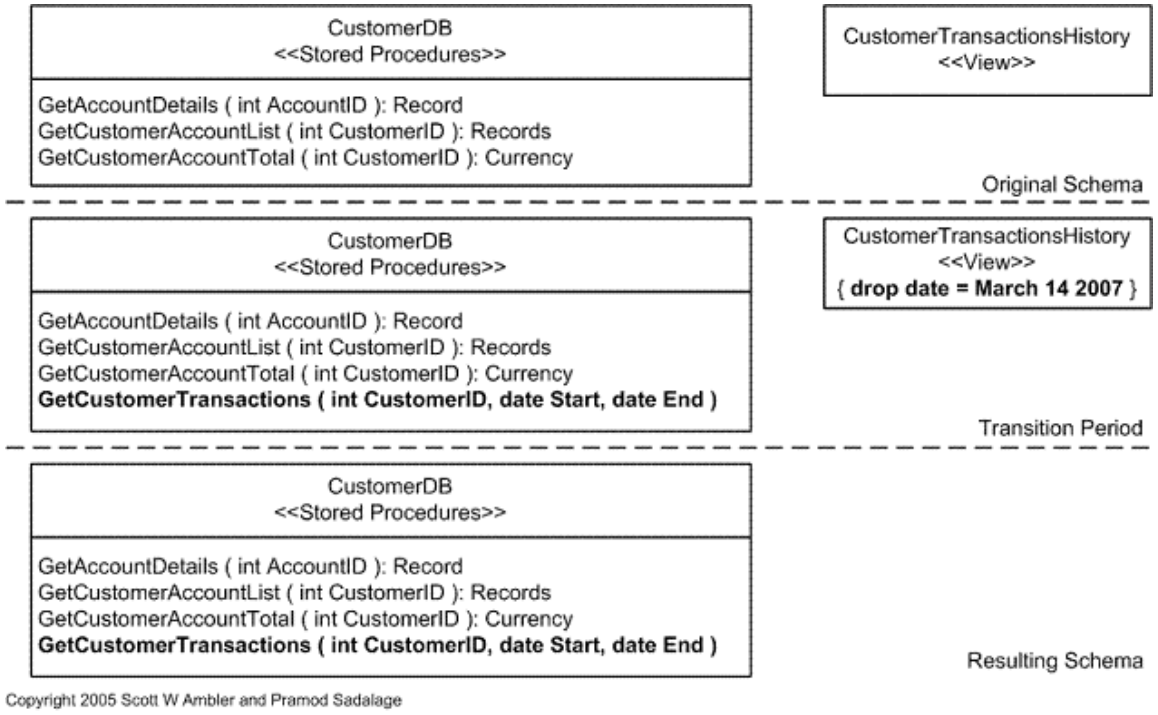

Figura 57. Trocar visão por método (s)

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogArchitectural.html

- Uso oficial da fonte de dados (diretamente)

No exemplo da figura 57 uso oficial da fonte de dados (diretamente), no esquema original tem-se uma aplicação, uma base de dados com duas tabelas Account (conta) e Customer (cliente) que estão diretamente ligadas á aplicação e outra base de dados Customer (cliente) com a tabela Customer (cliente), logo após o período de transição (refatoração) é utilizada a fonte de dados oficial Customer (cliente) da tabela Customer (cliente) do que a fonte que estava sendo usada anteriormente.

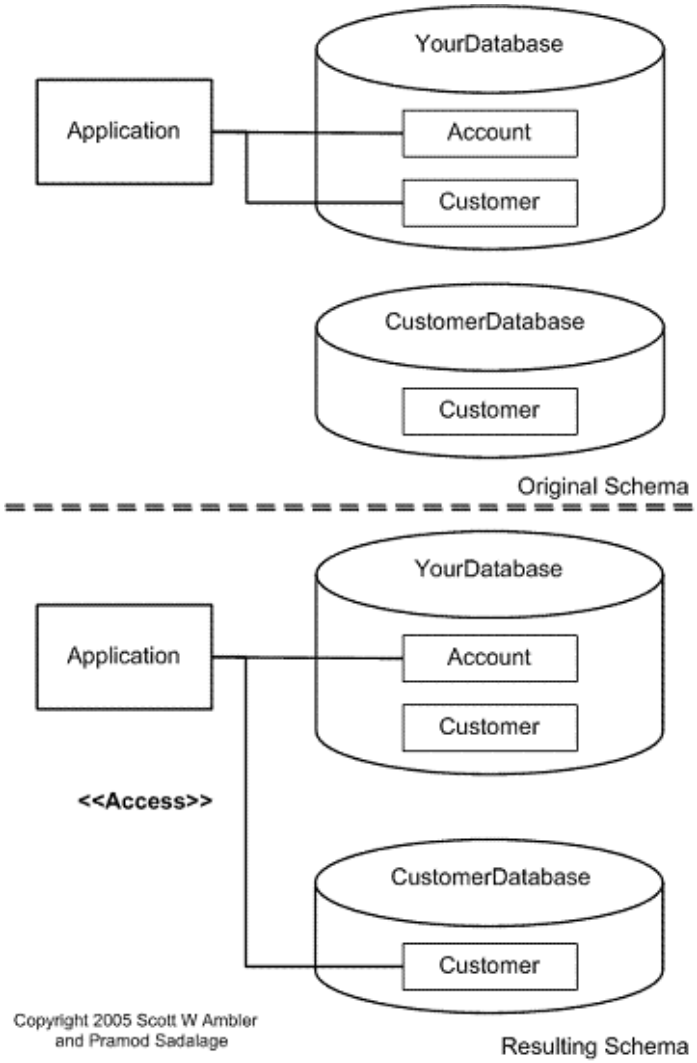

Figura 58. Uso oficial da fonte de dados diretamente

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogArchitectural.html

- Uso oficial fonte de dados (via replicação)

No exemplo da figura 58 uso oficial da fonte de dados via replicação, no esquema original tem-se uma base de dados que possui duas tabelas que são: a tabela Account (conta), e a tabela Customer (cliente), ambas as tabelas ligadas á uma aplicação, e outra fonte de dados isolada Customer (cliente) com a tabela Customer (cliente), logo após o período de transição (refatoração) é feita replicação entre a base de dados Customer (cliente) com a tabela Customer (cliente) e a outra tabela Customer (cliente) da outra base de dados existente.

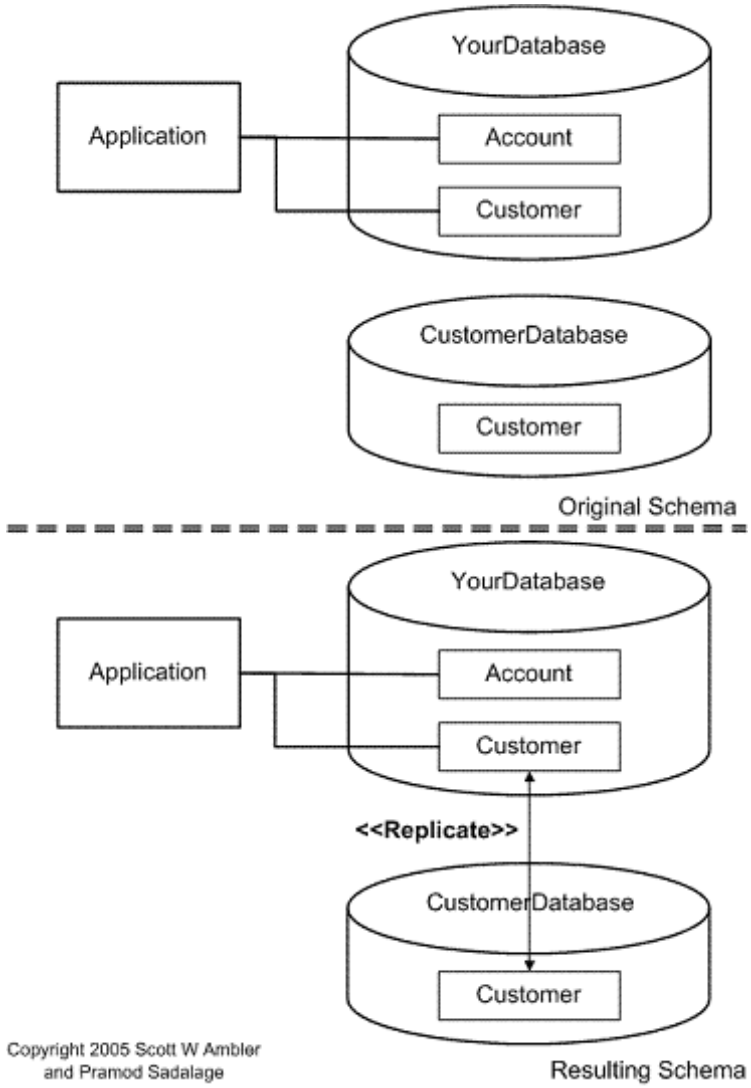

Figura 59. Uso oficial fonte de dados via replicação

Fonte: http://www.agiledata.org/essays/databaseRefactoringCatalogArchitectural.html

4.5 Considerações Finais Sobre o Capítulo

Neste capítulo foram abordados exemplos sobre as quatro categorias de refatoração de banco de dados e suas respectivas técnicas contidas em cada categoria. Foram analisados e explicados pelo autor deste projeto, todos os exemplos foram retirados do catálogo de refatoração de banco de dados, de Scott W Ambler que se encontra disponível no endereço eletrônico abaixo: http://www.agiledata.org/essays/databaseRefactoringCatalog.html.

No capítulo seguinte será apresentado o processo de refatoração de banco de dados, também serão abordados, além do processo de refatoração, problemas no processo proposto por Ambler retomado na Tese apresentada por Márcia Beatriz Pereira Domingues, que referencia Ambler em seu projeto, será apresentado em figuras o melhor e pior cenário para se aplicar a refatoração de banco de dados, como refatorar o banco de dados, e o passo a passo para o mesmo, material que será retirado e referenciado pelo endereço eletrônico logo abaixo: http://www.agiledata.org/essays/databaseRefactoring.html.

5 PROCESSO DE REFATORAÇÃO EM BANCO DE DADOS

Neste capítulo será apresentado e contextualizado o processo de refatoração de banco de dados, serão descritos as etapas que uma refatoração em banco de dados deve conter, levando em conta a preservação da semântica e comportamento do banco de dados em questão.

Na tese de doutorado apresentada por Márcia Beatriz Pereira Domingues, foi apresentado um novo processo para a refatoração em banco dados. A tese também apresentou uma análise dos problemas no processo proposto por Ambler, de adaptações no processo e vantagens no mesmo, que também serão abordadas neste capítulo além também de fazer uma mescla com outra base de dados em questão sobre o processo de refatoração, que como já explicado é muito mais trabalhoso do que uma refatoração de código devido ao acoplamento, em que muitas aplicações, frameworks e diversas fontes acessam e compartilham dados e informações do mesmo banco de dados.

Neste capítulo como já havia dito será explanado todo o processo de refatoração em partes para melhor entendimento sobre o processo como um todo, e os problemas do processo proposto por Ambler.

5.1 Por que refatorar o banco de dados

Há duas razões fundamentais para querer refatorar o banco de dados, são essas razões: (AGILEDATA, 2015).

1. **Para corrigir segurança em bases de dados legadas existentes**. A linha inferior mostra que bancos de dados legados não irão corrigir a si mesmos, e que a partir de um ponto técnico de refatoração de banco de dados vista, é uma forma segura, simples de melhorar os dados e o banco de dados, a qualidade ao longo do tempo (AGILEDATA, 2015).
2. **Para apoiar o desenvolvimento** evolutivo. Processos de desenvolvimento de software modernos, como o RUP, XP, e AUP, todo o trabalho em um evolutivo se não forma ágil.

5.1.1 Preservando a Semântica

Semântica informativa refere-se ao significado da informação no banco de dados a partir do ponto de vista dos usuários dessa informação. Para preservar a semântica informativa implica que quando você mudar os valores dos dados armazenados em uma coluna os clientes de que a informação não deve ser afetada pela melhoria. Da mesma forma, em relação à semântica comportamental o objetivo é manter a funcionalidade da caixa preta o mesmo código, fonte que trabalha com os aspectos de mudanças de seu esquema de banco de dados, que deve ser reformulado para realizar a mesma funcionalidade que antes (AGILEDATA, 2015).

5.1.2 Por que refatoração de banco de dados é difícil

Acoplamento, em bancos de dados relacionais é uma medida do grau de dependência entre dois itens, ao acoplar duas coisas se tem a chance de que ocorra a mudança em um que consequentemente exigirá uma mudança no outro. O acoplamento é a "raiz de todo mal", quando se trata de refatoração de banco de dados, se as coisas que estão em seu esquema de banco de dados são acopladas ao que são mais difíceis, os esquemas de banco de dados relacionais são automaticamente acopladas a uma grande variedade de coisas, que geralmente são: (AGILEDATA, 2015).

- Seu código fonte da aplicação
- Outro código fonte da aplicação
- Código-fonte de carregamento de dados
- Código-fonte extrair dados
- Frameworks de persistência / camadas
- Seu esquema do banco
- Scripts de migração de dados
- Código de ensaio
- Documentação

A figura 60 ilustra o melhor cenário para a refatoração em banco de dados, isso quando existe apenas uma aplicação interligada diretamente ao banco de dados existente.

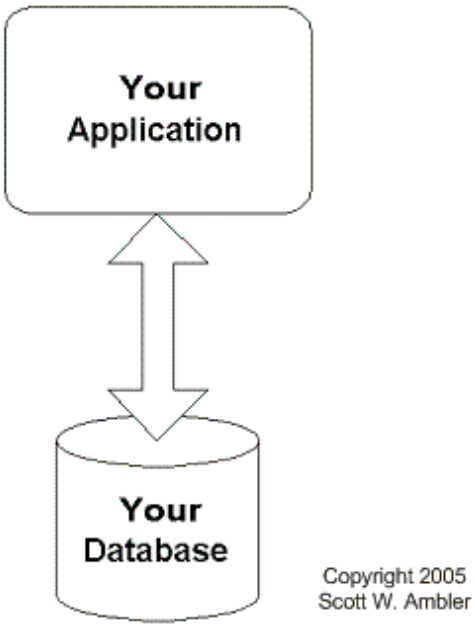

Figura 60. O melhor cenário

Fonte: http://agiledata.org/essays/databaseRefactoring.html

A figura 61 demonstra o pior cenário de refatoração em banco de dados, devido ao acoplamento.

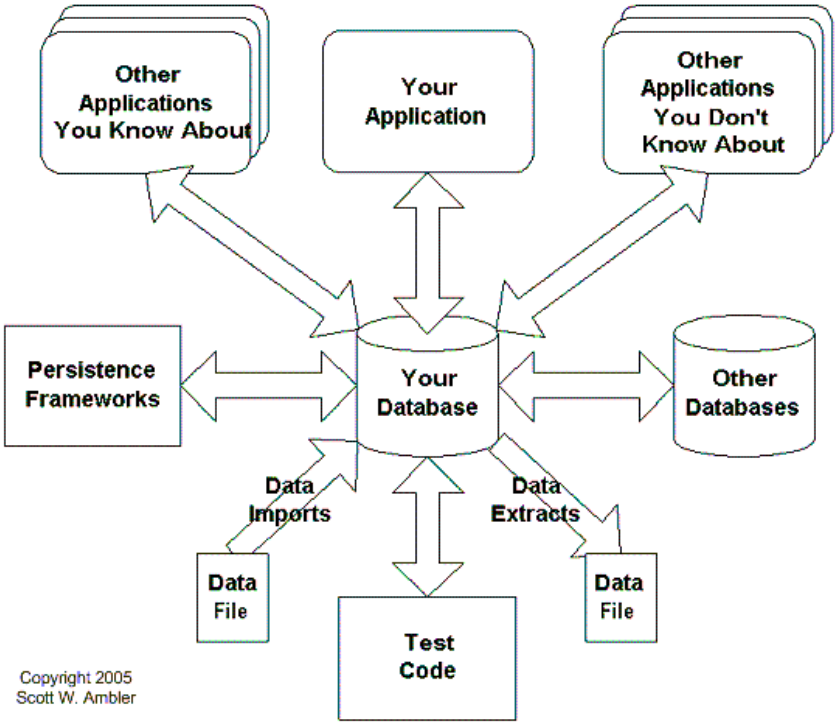

Figura 61. O cenário de pior caso

Fonte: http://agiledata.org/essays/databaseRefactoring.html

5.1.3 O que são refatoração de banco de dados e o que não são refatoração

Uma pequena transformação em seu esquema de estendê-lo, tal como a adição de uma nova coluna ou tabela, não é uma refatoração de banco de dados porque a mudança aumenta seu design (AGILEDATA, 2015).

Um grande número de pequenas mudanças simultaneamente aplicadas ao seu esquema de banco de dados, como a renomeação de dez colunas, não é considerado uma refatoração de banco de dados, porque esta não é uma única e pequena mudança. Refatorações de banco de dados são pequenas alterações em seu esquema de banco de dados que melhora a sua concepção, preservando a semântica comportamental e informativa (AGILEDATA, 2015).

5.2 Como Refatorar o Banco de Dados

Antes de descrever os passos para refatorar um banco de dados, é preciso destacar uma questão crítica: O cenário simples destacado na figura 60 implicará a fazer coisas diferentes do que o cenário altamente acoplado da Figura 61. O processo fundamental por si, só permanece o mesmo, apesar da dificuldade de programar refatoração de banco de dados, se

os acoplamentos do banco de dados aumentarem também. (NETO, 2011).

Caso esteja em um cenário simples, não é necessário fazer o trabalho descrito abaixo, pode- se simplesmente refatorar o esquema de banco de dados e o código da aplicação em paralelo e implantá-los simultaneamente. Aqueles que se encontrar em um cenário mais complexo não terão este privilégio (NETO, 2011).

Esta seção assume que os ambientes técnicos e culturais estão organizados para apoiar a refatoração do banco de dados.

A refatoração de banco de dados pode ser vista como um processo de 3 passos: (NETO, 2011).

1. Começar em seu ambiente de desenvolvimento.
2. Programar em seu ambiente de integração.
3. Instalar em ambiente de produção.

5.2.1 Passo 1: Comece pelo ambiente de desenvolvimento

O ambiente de desenvolvimento é um ambiente técnico onde o software, incluindo o código da aplicação e esquema de banco de dados, é desenvolvido e testado (testes de unidade). A necessidade de refatorar um esquema de banco de dados é tipicamente identificada por um técnico que está tentando implantar um novo requisito ou que está corrigindo um defeito (NETO, 2011).

O DBA e o desenvolvedor tipicamente irão seguir alguns ou todos os passos para implantar uma ou um conjunto de refatorações em banco de dados, que serão descritas nas seções seguintes (NETO, 2011).

- Verificar que uma refatoração de banco de dados é necessária;
- Escolher a refatoração de código mais apropriada;
- Depreciar o esquema original;
- Escrever testes de unidade;
- Modificar o esquema do banco de dados;
- Migrar os dados de origem (que estavam na versão original);
- Atualizar os programas de acesso externo;

- Atualizar os scripts de migração dos dados;
- Executar testes de regressão;
- Divulgar a refatoração.

5.2.2 Verificar que a refatoração do banco de dados é necessária

A primeira coisa que o DBA faz é tentar determinar se a refatoração de banco de dados de fato precisa ocorrer e se ela é a escolha correta a ser realizada. A segunda coisa é avaliar internamente a probabilidade que a mudança é realmente necessária. Isso normalmente é decidido baseando-se nas experiências anteriores do DBA e do desenvolvedor. A próxima atividade que o DBA faz é avaliar o impacto geral da refatoração (NETO, 2011).

5.2.3 Escolher a refatoração mais adequada ao banco de dados

Uma habilidade importante que os DBAs necessitam, é o entendimento de que você normalmente tem várias opções para a implementação de novas estruturas de dados, e nova lógica dentro de um banco de dados. (AGILEDATA, 2015).

5.2.4 Fazer testes de unidade

Como refatoração de código, refatoração de banco de dados, está habilitada pela existência de um conjunto de testes abrangente no qual você sabe que pode mudar seu esquema de banco de dados com segurança se você constatar que o banco de dados ainda funciona após a mudança (AGILEDATA, 2015).

5.2.5 Depreciar o esquema original

Uma técnica efetiva de refatoração é usar um período de depreciação para a porção original do esquema que está sendo alterado. É observado que você pode simplesmente fazer a mudança em seu esquema de banco de dados instantaneamente, que em vez de precisar trabalhar com os esquemas velho e novo em paralelo por um período para prover um tempo para que equipes de desenvolvedores de outras aplicações refatorem e reimplantem seus sistemas (NETO, 2011).

A Figura 62 ilustra como essa ideia funcionaria quando aplicada o Substituir da coluna zip code (CEP) por refatoração de banco de dados para post code (código postal). Observe as mudanças entre o esquema original e o esquema durante o período de transição (AGILEDATA, 2015).

O código postal foi adicionado como uma coluna, a coluna Zip code (CEP) foi marcada como obsoleta, porque uma data de remoção foi atribuída à mesma usando uma UML chamada variável. Um gatilho também foi introduzido para manter os valores contidos nas duas colunas sincronizados, partindo do princípio de que o novo código do aplicativo irá trabalhar com código postal, mas não deve se esperar manter a coluna Zip code (CEP) até à data, e que o código do aplicativo mais antigo que não foi reformulado para usar o novo esquema não vai saber manter código postal até à data. Esse gatilho é um exemplo de código de andaimes banco de dados, código simples e comum que é necessário para manter seu banco de dados "colados um ao outro", este código foi atribuído à mesma data de remoção como do CEP, como ilustra a figura 62 (AGILEDATA, 2015).

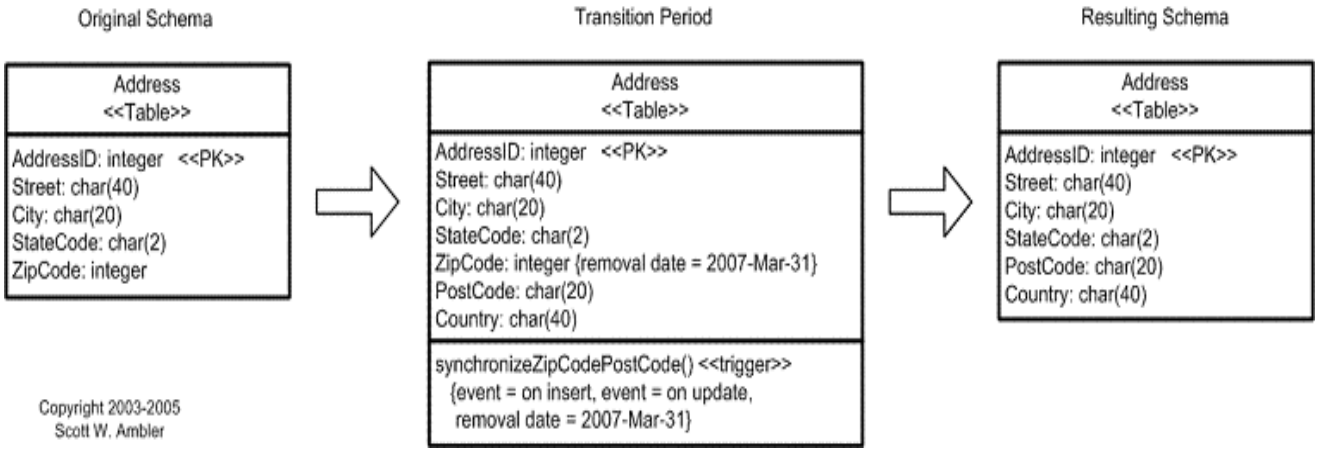

Figura 62. Refatorando tabela de endereço

Fonte: http://agiledata.org/essays/databaseRefactoring.html

5.2.6 Modificar o esquema do banco

O desenvolvedor e DBA trabalham juntos para fazer as mudanças no ambiente de desenvolvimento. A estratégia é iniciar cada refatoração de forma simples, realizando a refatoração no ambiente de desenvolvimento.

Para fazer isso, precisamos atualizar dois logs (assumindo que você não possua uma ferramenta de refatoração de banco de dados que faz isso automaticamente) (NETO, 2011).

1. Log de mudanças no banco de dados. Este log contém os códigos fontes que programam todas as mudanças no esquema do banco de dados na ordem em que elas ocorrem

ao longo do curso de um projeto. Quando estamos implementando uma refatoração de banco de dados, incluímos apenas as mudanças imediatas neste log.

2. Log de atualização. Este log contém o código fonte para mudanças futuras que serão implantadas no esquema de banco de dados que serão executadas após o período de transição da refatoração.

5.2.7 Migrar os dados

Muitas refatorações requerem migração ou cópia dos dados a partir da versão antiga do esquema para o novo. O log de migração dos dados contém a linguagem de manipulação de dados (*Data Manipulation Language* – DML) para reformatar os limpar os dados de origem ao longo do projeto (NETO, 2011).

5.2.8 Atualizar programas externos

Os programas que acessam a parte do esquema de banco que está sendo refatorado devem ser atualizados para trabalhar com a nova versão do esquema do banco de dados. Todos esses programas devem ser reformulados e em seguida, implantados em produção antes do término do período de transição (AGILEDATA, 2015).

5.2.9 Executar testes de regressão

Uma vez que as mudanças no código da aplicação e no esquema do banco de dados foram realizadas, precisa-se então executar os testes de regressão. Como testes realizados com sucesso descobrem problemas, precisa-se reavaliar os itens alterados até que estes estejam corretos.

Uma vantagem significativa das refatorações de banco de dados serem pequenas mudanças é que se os testes falham, é mais simples de identificar a origem do problema. Quanto maior for a mudança, mais difícil será identificar o problema que originará um eventual problema, e portanto o esforço aumentará. Isso confirma que desenvolvimento em passos pequenos e incrementais funciona muito bem na prática (NETO, 2011).

5.2.10 Divulgar as mudanças que fez

Como seu banco de dados é um recurso compartilhado (minimamente ele é compartilhado pela equipe de desenvolvimento da aplicação, isso se não for por várias outras aplicações), o DBA precisa comunicar às mudanças que foram feitas. Se ainda não foi feito, então devemos atualizar o modelo de dados físico do banco de dados. Uma alternativa interessante é usar ferramentas que a partir do modelo físico geram os scripts para alteração do banco de dados (NETO, 2011).

5.2.11 Controlar a Versão de trabalho

A habilidade crítica para desenvolvedores ágeis é o hábito de colocar todo o seu trabalho sobre gerenciamento de configurações de controle, verificando-o em uma ferramenta de controle de versão. No caso do banco de dados de refatoração isso inclui qualquer DDL (Data Definition Language) que você criou scripts de alteração, scripts de migração de dados, dados de teste, casos de teste, teste de código de geração de dados, documentação e modelos. Isto é, além de os artefatos centrados em aplicações que normalmente versão tratar o seu artefato orientado para o banco de dados da mesma maneira que você iria tratar outros artefatos de desenvolvimento você deve estar ok (AGILEDATA, 2015).

5.3 Passo 2: Implementação do ambiente de integração

Após vários dias de trabalho estaremos prontos para implantar a refatoração de banco de dados no ambiente de integração. A razão pela qual precisamos aguardar para fazer isso é que precisamos dar tempo às equipes de outros sistemas refatorarem seus códigos para usarem o novo esquema de banco de dados (NETO, 2011).

As equipes que escolherem encapsular o acesso ao banco de dados usando um framework de persistência julgarão simples o tratamento às alterações no esquema de banco de dados e, portanto descobrirão que poderão encurtar o período entre a implementação da refatoração do banco de dados no ambiente de desenvolvimento e em seu ambiente de integração do projeto. Isso ocorre pelo fato do esquema do banco de dados ser representado em metadados, portanto muitas mudanças no esquema do banco de dados irão requerer ajustes apenas nos metadados e não no código fonte atual (NETO, 2011).

5.4 Passo 3: Instalação em produção

Instalar em produção é a parte mais difícil do processo de refatoração de banco de dados, mais ainda no cenário complexo da Figura 61. Geralmente não implantaremos refatorações de banco de dados apenas para o nosso sistema, em vez disso iremos implantá-las como parte do desenvolvimento geral de um ou vários sistemas (NETO, 2011).

A implantação é mais fácil quando temos apenas uma aplicação e um banco de dados para atualizar, e este cenário ocorre na prática, mas de fato precisamos considerar o cenário onde iremos implantar vários sistemas e várias fontes de dados ao mesmo tempo. A Figura 63 apresenta uma visão geral dos passos para implantação da refatoração em um ambiente de produção (NETO, 2011).

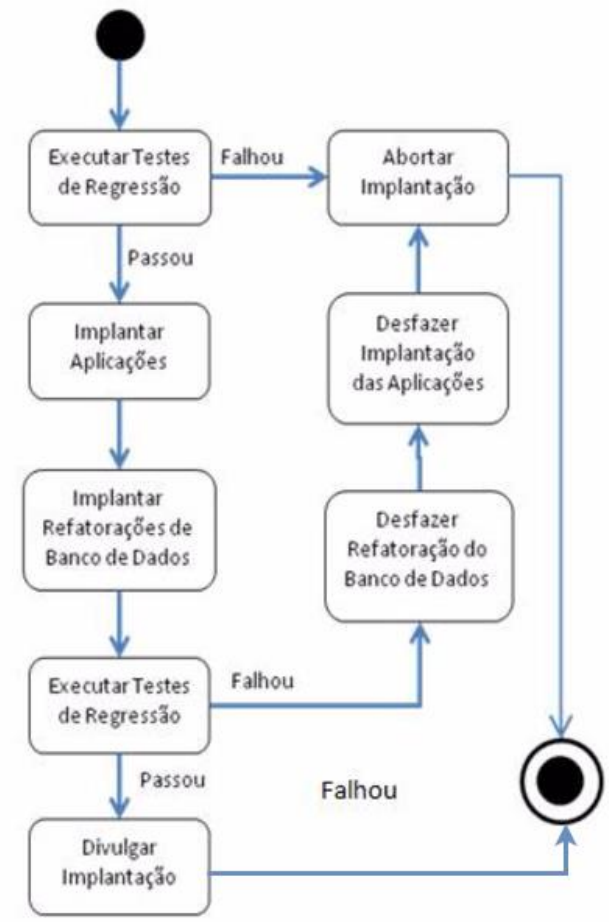

Figura 63. Os passos para implantação de refatoração de banco de dados

Fonte: http://www.devmedia.com.br/o-processo-de-refatoracao-de-banco-de-dados-artigo-revista-sql-magazine-86/20400

5.5 Problemas no Processo Proposto por Ambler

Nesta seção serão abordados problemas no processo original proposto por Scott W. Ambler (2006) na refatoração de banco de dados, onde a autora da Tese Márcia Beatriz Pereira Domingues (2014), enfatiza no capítulo 5 cerca de nove problemas no modelo de processo que Ambler propôs para a refatoração de banco de dados, cada parte do processo de refatoração é apontado um problema diferente, como será especificado nesta seção, o modelo original do processo de refatoração de banco de dados está ilustrado na figura 64 e, já foram contextualizados o processo original nas seções anteriores, os problemas com o mesmo logo em seguida, após a ilustração da figura 64.

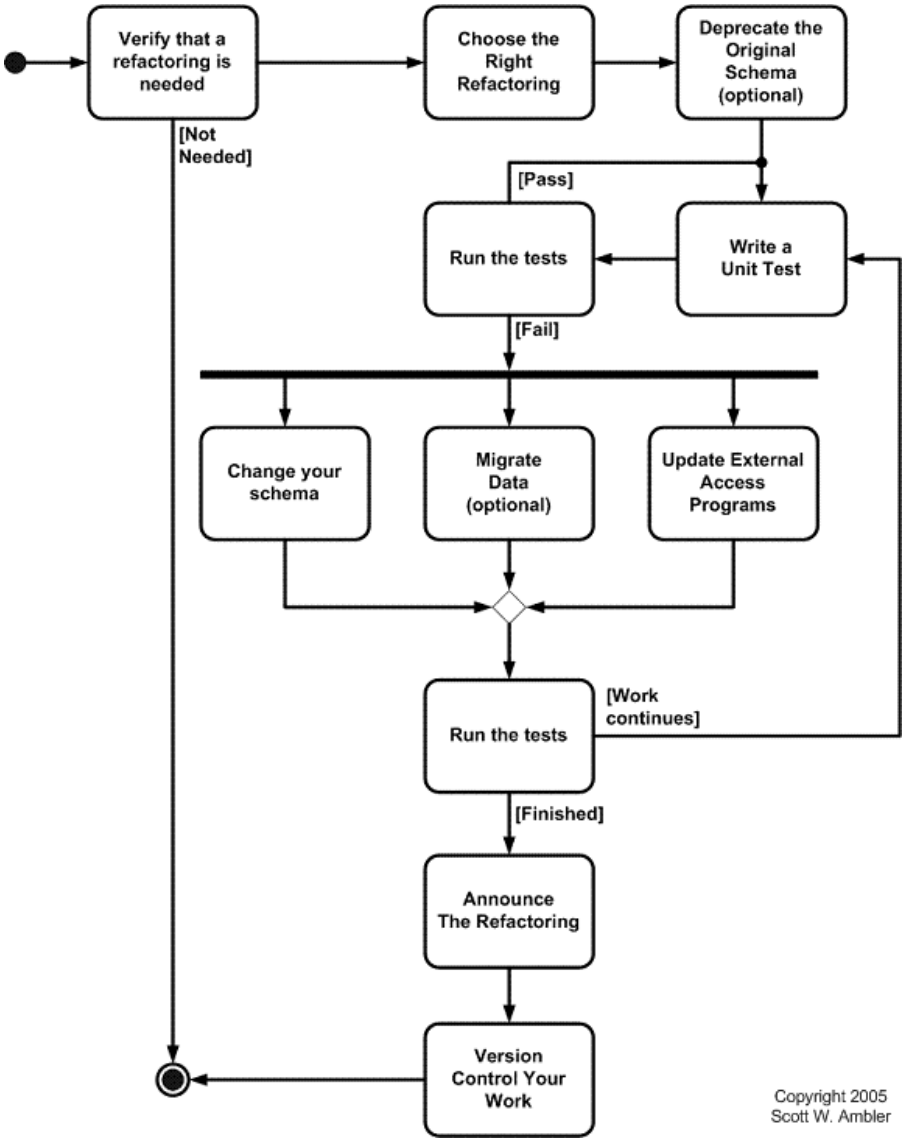

Figura 64. O processo de refatoramento base de dados

Fonte: http://agiledata.org/essays/databaseRefactoring.html

Os problemas neste modelo de processo de refatoração de banco de dados proposto por Scott W. Ambler (2006), enfatizados por Márcia Beatriz Pereira Domingues em sua tese, um novo processo para refatoração de banco de dados (2014), estão contextualizados abaixo.

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 (DOMINGUES, 2014).

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 (DOMINGUES, 2014).

3. Tarefa: Tornar o esquema atual obsoleto (opcional). Basicamente essa tarefa tem o objetivo de avisar todo o desenvolvedor o que será feito no banco de dados, para que se prepararem 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 (DOMINGUES, 2014).

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 (DOMINGUES, 2014).

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 (DOMIGUES, 2014).

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 (DOMINGUES, 2014).

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 se 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 (DOMINGUES, 2014).

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 (DOMINGUES, 2014).

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 (DOMINGUES, 2014).

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.

(DOMINGUES, 2014).

5.6 Considerações Finais Sobre o Capítulo

Neste capítulo foi discutido o processo de refatoração de banco de dados proposto por Scott W. Ambler (2006) autor do livro Refactoring Databases: Evolutionary Database Design juntamente com Pramond J. Sadalage, que disponibiliza além de um catálogo de refatorações de banco de dados abordados no capítulo anterior, como também o processo de refatoração de banco de dados em um blog cujo endereço eletrônico é http://agiledata.org/essays/databaseRefactoring.html, e foi feito uma mescla com um artigo da Devmedia publicado em 2011 por Arilo Cláudio Dias Neto, sobre o processo de refatoração de banco de dados.

Foram estudados e apontados problemas sobre o processo original proposto e disponibilizado por Ambler (2006) na tese de doutorado de Márcia Beatriz Pereira Domingues (2014), neste capítulo foi debatido um problema que torna a refatoração de banco de dados mais difícil que é o acoplamento, quanto mais acoplado estiver o banco de dados em questão mais difícil se torna a refatoração, porque ao refatorar o banco de dados, os programas externos interligados a ele precisam ser atualizados para corresponder às mudanças que foram feitas no banco de dados.

No capítulo seguinte serão contextualizados os casos de uso (exemplos) sobre as técnicas e ou refatorações de banco de dados, que são divididos em seis categorias que como já dito são: estrutural, qualidade de dados, integridade referencial, arquitetural, método, e transformações. Nestas categorias no total se encontram setenta refatorações de banco de dados, mas nos trabalhos correlatos são citadas e comentadas apenas quatro que são: estrutural, qualidade de dados, integridade referencial, e arquitetural, nos casos de uso. No capítulo seguinte serão disponibilizadas algumas tabelas (exemplos) com as seis categorias e todas as refatorações que Ambler disponibilizou em seu livro e também em seu blog.

No próximo seguimento serão apresentados também, os scripts utilizados para se aplicar o conceito de refatoração, e as ilustrações (figuras) sobre as refatorações, como foram apresentados no capítulo quatro às figuras sobre as refatorações de banco de dados.

6 ESTUDO COMPARATIVO DE TÉCNICAS DE REFATORAÇÃO DE BANCO DE DADOS

Nesse capítulo serão contextualizadas tabelas com as seis categorias de refatoração de banco de dados, e suas técnicas de refatoração, ao todo são cerca de 70 técnicas de refatoração de banco de dados presente nas tabelas abaixo.

Tabela 1. Refatorações Estruturais

Técnicas de Refatoração		Autor 1	Autor 2	Autor 3
Categoria 1	Refatoração	Scott W. Ambler Autor do Livro de Refatoração em Banco de Dados	Márcia Beatriz Pereira Domingues	Helves Humberto Domingues
Estrutural	Eliminar Coluna	X	X	X
Estrutural	Eliminar Tabela	X		X
Estrutural	Eliminar Visão	X		X
Estrutural	Introduzir Coluna Calculada	X		X
Estrutural	Introduzir chave de identificação	X	X	X
Estrutural	Unir Colunas	X	X	X
Estrutural	Unir Tabelas	X		X
Estrutural	Mover Coluna	X		X
Estrutural	Renomear Coluna	X		X
Estrutural	Renomear Tabela	X		X
Estrutural	Renomear Visão	X		X
Estrutural	Trocar Coluna	X		X
Estrutural	Trocar Coluna Complexa por Tabela	X		X
Estrutural	Trocar um para muitos por tabela associativa	X		X
Estrutural	Trocar chave de identificação por chave natural	X		X
Estrutural	Dividir Coluna	X		X
Estrutural	Dividir Tabela	X		X
Total de Refatorações = 17		Total = 17	Total = 3	Total = 17

Fonte: Próprio Autor

A tabela 1 demonstra a categoria de refatoração estrutural, com 17 técnicas de refatoração, às mesmas discutidas no livro e no catálogo web de refatorações de Scott W. Ambler. Serão mostradas também, as técnicas de refatoração abordadas pelos autores nos trabalhos correlatos, todas marcadas com x, e o total de técnicas de refatoração que cada autor abordou.

Tabela 2. Técnicas de Refatoração Qualidade de Dados

Técnicas de Refatoração		Autor 1	Autor 2	Autor 3
Categoria 2	Refatoração	Scott W. Ambler Autor do Livro de Refatoração em Banco de Dados	Márcia Beatriz Pereira Domingues	Helves Humberto Domingues
Qualidade de dados	Adicionar Tabela Descritiva	X	Não Utilizou	X
Qualidade de dados	Aplicar Código Padrão	X	Não Utilizou	X
Qualidade de dados	Aplicar Tipo Padrão	X	Não Utilizou	X
Qualidade de dados	Consolidar Estratégias de Chaves	X	Não Utilizou	X
Qualidade de dados	Remover Restrição de Coluna	X	Não Utilizou	X
Qualidade de dados	Remover Valor Padrão	X	Não Utilizou	X
Qualidade de dados	Remover Restrição de não Nulo	X	Não Utilizou	X
Qualidade de dados	Introduzir Coluna de Restrição	X	Não Utilizou	X
Qualidade de dados	Introduzir formato Comum	X	Não Utilizou	X
Qualidade de dados	Introduzir Valor Padrão	X	Não Utilizou	X
Qualidade de dados	Alterar Coluna para não Nula	X	Não Utilizou	X
Qualidade de dados	Mover Dados	X	Não Utilizou	X
Qualidade de dados	Trocar Código de Tipo por Propriedades Sinalizadoras	X	Não Utilizou	X
Total de Refatorações = 13		TOTAL = 13	Total = 0	TOTAL = 13

Fonte: Próprio Autor

A tabela 2 demonstra a categoria de refatoração qualidade de dados, com 13 técnicas de refatoração, às mesmas contextualizadas no livro e no catálogo web de Scott W. Ambler. Também estão evidenciadas na tabela, as técnicas de refatoração que foram abordadas pelos

autores nos trabalhos correlatos, e, o total de técnicas de refatorações que cada autor abordou.

Tabela 3. Refatorações de Integridade Referencial

Técnicas de Refatoração		Autor 1	Autor 2	Autor 3
Categoria 3	Refatoração	Scott W. Ambler Autor do Livro de Refatoração em Banco de Dados	Márcia Beatriz Pereira Domingues	Helves Humberto Domingues
Integridade Referencial	Adicionar Restrição de Chave Estrangeira	X	X	X
Integridade Referencial	Adicionar Trigger para Coluna Calculada	X		X
Integridade Referencial	Remover restrição de chave estrangeira	X		X
Integridade Referencial	Introduzir exclusão em Cascata	X		X
Integridade Referencial	Introduzir exclusão Lógica	X		X
Integridade Referencial	Introduzir exclusão Física	X		X
Integridade Referencial	Adicionar Trigger para Histórico	X		X
Total de Refatorações = 7		Total = 7	Total = 1	Total = 7

Fonte: Próprio Autor

A tabela 3 demonstra a categoria de refatoração integridade referencial, com 7 técnicas de refatoração, as mesmas abordadas no livro e no catálogo web de Scott W. Ambler. Outra amostragem são as técnicas de refatoração que foram abordadas pelos autores nos trabalhos correlatos, como demonstrado na tabela 3 marcada com x, e o total de técnicas de refatoração que cada autor abordou.

Tabela 4. Refatorações Arquiteturais

Técnicas de Refatoração		Autor 1	Autor 2	Autor 3
Categoria 4	Refatoração	Scott W. Ambler Autor do Livro de Refatoração em Banco de Dados	Márcia Beatriz Pereira Domingues	Helves Humberto Domingues
Arquitetural	Adicionar Métodos Crud	X		X
Arquitetural	Introduzir Tabela Espelho	X		X
Arquitetural	Adicionar Método de Leitura	X		X
Arquitetural	Encapsular tabela com uma visão	X		X
Arquitetural	Introduzir Método para Cálculo	X		X
Arquitetural	Introduzir Índice	X	X	X
Arquitetural	Introduzir tabela somente de leitura	X		X
Arquitetural	Migrar método para banco de dados	X		X
Arquitetural	Migrar método de banco de dados	X		X
Arquitetural	Trocar método por Visão	X		X
Arquitetural	Trocar visão por Método	X		X
Arquitetural	Usar Fonte de dados Oficial	X		X
Total de Refatorações = 12		Total = 12	Total = 1	Total = 12

Fonte: Próprio Autor

A tabela 4 demonstra a categoria de refatoração arquitetural, com 12 técnicas de refatoração, contextualizadas no livro e no catálogo web de refatorações de Scott W. Ambler. Também as técnicas de refatoração que foram elucidadas nos trabalhos correlatos desta categoria de refatoração, e o total de técnicas de refatoração abordadas por cada autor.

Tabela 5. Refatorações de Método

Técnicas de Refatoração		Autor 1	Autor 2	Autor 3
Categoria 5	Refatoração	Scott W. Ambler Autor do Livro de Refatoração em Banco de Dados	Márcia Beatriz Pereira Domingues	Helves Humberto Domingues
Método	Adicionar parâmetro	X	Não é Citado	Não é Citado
Método	Consolidar Expressão Condicional	X	Não é Citado	Não é Citado
Método	Decompor Condicional	X	Não é Citado	Não é Citado
Método	Extrair Método	X	Não é Citado	Não é Citado
Método	Introduzir Variável	X	Não é Citado	Não é Citado
Método	Parametrizar Métodos	X	Não é Citado	Não é Citado
Método	Remover Sinalização de Controle	X	Não é Citado	Não é Citado
Método	Remoção Intermediária	X	Não é Citado	Não é Citado
Método	Remover Parâmetro	X	Não é Citado	Não é Citado
Método	Renomear Método	X	Não é Citado	Não é Citado
Método	Reordenar Parâmetros	X	Não é Citado	Não é Citado
Método	Substituir Literal Com tabela de pesquisa	X	Não é Citado	Não é Citado
Método	Substituir Expressão Aninhada Com Cláusulas Guarda	X	Não é Citado	Não é Citado
Método	Substituir parâmetro com métodos específicos	X	Não é Citado	Não é Citado
Método	Dividir Variável Temporária	X	Não é Citado	Não é Citado
Método	Algoritmo substituto	X Visualização web Indisponível	Não é Citado	Não é Citado
Total de Refatorações = 16		Total = 16	Total = 0	Total = 0

Fonte: Próprio Autor

A tabela 5 demonstra a categoria de refatoração de método, com 16 técnicas de refatoração, contextualizadas no livro e no catálogo web de refatorações de Scott W. Ambler. Nessa categoria apenas o autor do livro e do catálogo web de refatorações Scott W. Ambler abordou o assunto, e suas respectivas técnicas de refatoração de banco de dados.

Tabela 6. Refatorações Transformações-Não Refatorações

Técnicas de Refatoração		Autor 1	Autor 2	Autor 3
Categoria 6	Refatoração	Scott W. Ambler Autor do Livro de Refatoração em Banco de Dados	Márcia Beatriz Pereira Domingues	Helves Humberto Domingues
Transformações – Não Refatoração	Inserir dados	X	Não é Citado	Não é Citado
Transformações – Não Refatoração	Introduzir nova coluna	X	Não é Citado	Não é Citado
Transformações – Não Refatoração	Introduzir Nova Tabela	X	Não é Citado	Não é Citado
Transformações – Não Refatoração	Introduzir Vista	X	Não é Citado	Não é Citado
Transformações – Não Refatoração	Atualizar Dados	X	Não é Citado	Não é Citado
Total de Refatorações = 5		Total = 5	Total = 0	Total = 0

Fonte: Próprio Autor

A tabela 6 demostra a categoria de refatoração Transformações – não refatoração, com 5 técnicas de refatoração, todas contextualizadas no livro e no catálogo web de Scott W. Ambler. Nessa categoria apenas o autor do livro e do catálogo web de refatorações as abordaram e mostraram suas respectivas técnicas de refatoração.

6.1 Considerações Finais Sobre o Capítulo

Neste capítulo foram apresentadas tabelas sobre as seis categorias de refatoração de banco de dados que como já ditos são: estrutural, qualidade de dados, integridade referencial, arquitetural, método e, transformações não refatoração. Além das categorias de refatoração de banco de dados, este capítulo apresentou todas as 70 técnicas de refatoração de banco de dados que estão no livro de Scott W. Ambler, e no catálogo web que estão dividas nas seis categorias de refatoração de banco de dados.

Nas tabelas em questão estão marcadas com x as técnicas de refatoração de banco de dados que cada autor utilizou em seu trabalho, seja um estudo caso, como também em um caso de uso (exemplo), Ambler utilizou todas as mesmas em seu livro Refactoring Databases: Evolutionary Database Design juntamente com Sadalage, e que o mesmo também se encontra disponível no endereço eletrônico http://agiledata.org/essays/databaseRefactoring.html, na tese Domingues (2011) foram utilizadas 49 técnicas de refatoração de banco de dados de quatro categorias como estão contextualizadas nas tabelas 1, 2, 3 e, 4, e demarcadas com x em casos de uso (exemplos), já Domingues (2014) utilizou em sua tese 5 técnicas de refatoração de banco de dados de três categorias como demostra as tabelas 1, 3 e, 4 demarcadas com x as técnicas de refatoração que foram utilizadas em seu estudo de caso.

No capítulo seguinte serão contextualizados casos de uso (exemplos) sobre quatro categorias de refatoração de banco de dados, que são: estrutural, qualidade de dados, integridade referencial, e, arquitetural, com seis técnicas de refatoração, que foram abordas nos trabalhos correlatos, que são: renomear coluna, renomear tabela, e, eliminar coluna, ambas da categoria de refatoração de banco de dados estrutural, introduzir valor comum que pertence à categoria de refatoração de banco de dados qualidade de dados, adicionar restrição de chave estrangeira que pertence a categoria de refatoração integridade referencial e, introduzir índice que pertence a categoria de refatoração de banco de dados arquitetural e, os scripts SQL que foram utilizados em cada caso de uso (exemplo) com explicações sobre os respectivos casos de uso (exemplos), e ilustrações (figuras) sobre as mesmas.

7 CASOS DE USO

Neste capítulo serão contextualizados casos de uso (exemplos) sobre as técnicas de refatorações de banco de dados, serão comentados e exemplificados as categorias de refatoração estrutural, qualidade de dados, integridade referencial, e arquitetural, que corrige erros detectados no esquema do banco de dados mantendo a semântica e a integridade dos dados, as categorias de refatoração de banco de dados e suas respectivas técnicas de refatoração estão nas seções seguintes, com os casos de uso (exemplos) e os scripts SQL que foram utilizados em cada caso de uso (exemplo).

7.1 Refatoração Estrutural Exemplos

No caso de uso (exemplo) da figura 65 renomear coluna, tem- se a tabela cliente com cinco colunas sendo que a coluna P Nome não tem um significado especifico logo após o período de transição (refatoração) como demonstra à imagem a coluna P Nome é renomeada, e no esquema resultante tem-se a coluna renomeada para Primeiro Nome que se dá um entendimento melhor da coluna em questão e os dados que nesta coluna podem ser armazenados.

O script de banco de dados utilizado para renomear a coluna PNome foi:

EXEC sp_RENAME [Cliente.PNome], [PrimeiroNome], 'COLUMN';

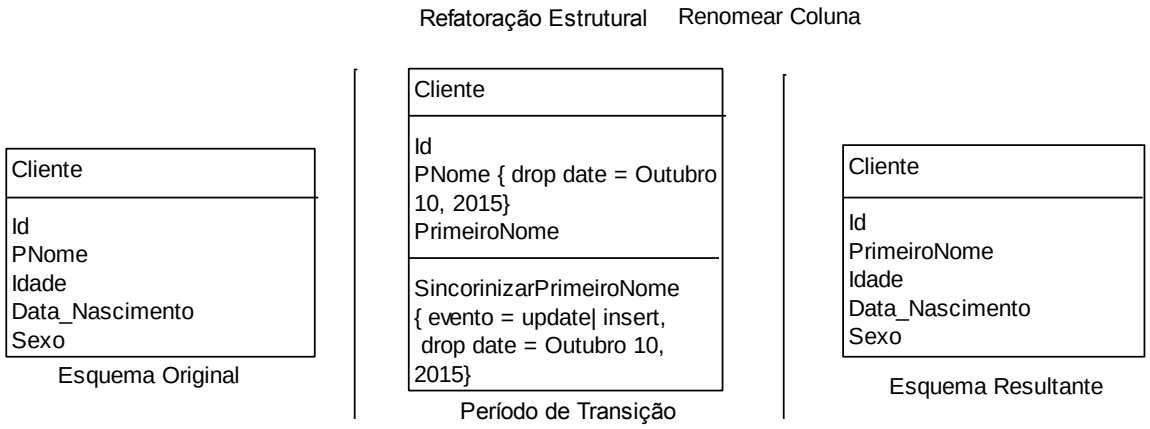

Figura 65. Refatoração Estrutural Renomear Coluna

Fonte: Próprio Autor

No caso de uso (exemplo) da figura 66 tem-se a tabela Empr. com cinco colunas,

logo após o período de transição (refatoração) o nome da tabela foi reformulado de Empr. para Empregado, onde demonstrou o objetivo da tabela em questão, e os dados que a mesma deveria armazenar.

O script de banco de dados utilizado para renomear a tabela Empr. foi:

EXEC sp_rename [Empr.], [Empregado];

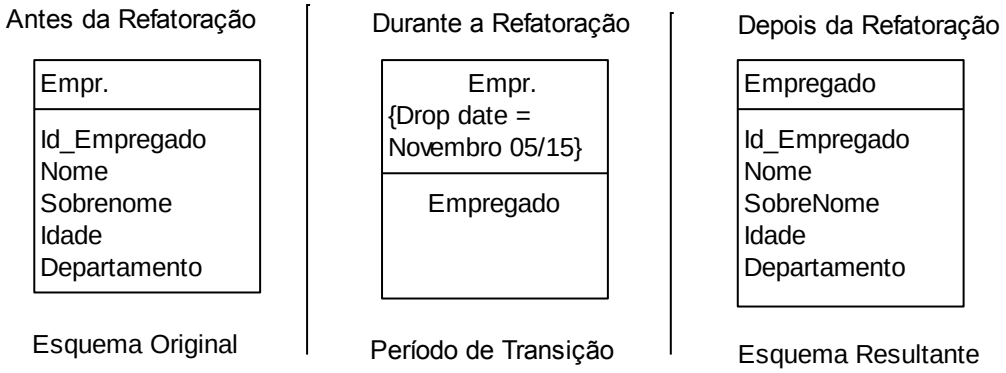

Figura 66. Refatoração Estrutural Renomear Tabela

Fonte: Próprio Autor

No caso de uso (exemplo) da figura 67 eliminar coluna, tem-se a tabela Empregado com nove colunas sendo que a coluna ID_Empregado está denominada como chave primária da tabela em questão, logo após o período de transição (refatoração) a coluna Função foi eliminada, o motivo da eliminação da coluna aconteceu porque a mesma não estava sendo utilizada e se apresentava repetitiva. Isto porque outra coluna da tabela já se enquadrava na informação que a coluna Função armazenava, ou porque o script de banco de dados já tinha sido utilizado para a eliminação da coluna Função.

O script de banco de dados utilizado para eliminar a coluna Função foi:

ALTER TABLE Empregado DROP COLUMN Função;

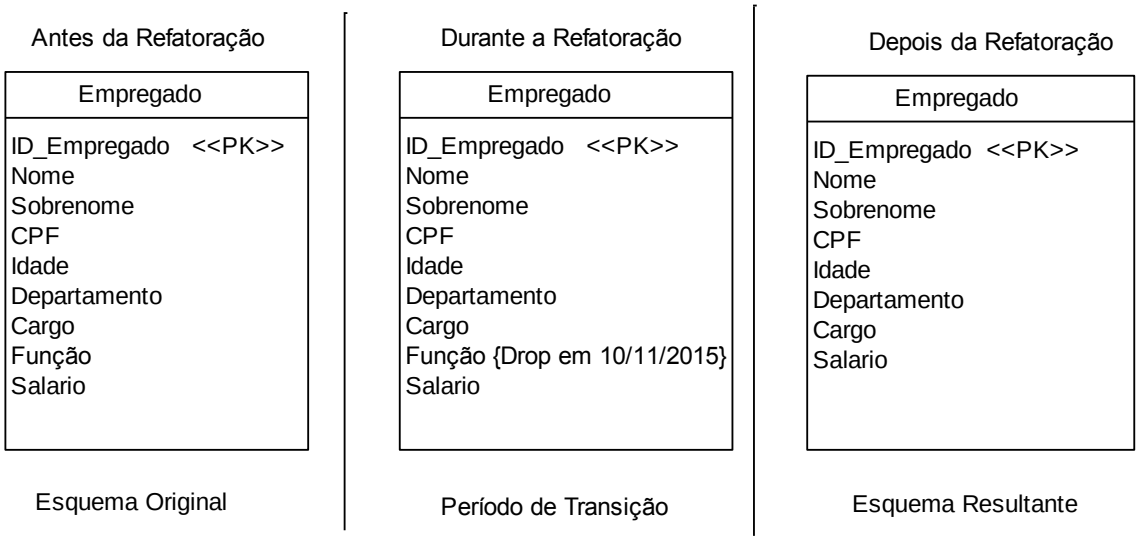

Figura 67. Refatoração Estrutural Eliminar Coluna

Fonte: Próprio Autor

7.2 Refatoração Qualidade de dados Exemplo

No caso de uso (exemplo) da figura 68, introduzir valor comum, tem-se a tabela funcionário, com 7 colunas, sendo que a coluna Id_Funcionário foi denominada como chave primária, logo após o período de transição (refatoração) foi adicionada uma restrição de valor na coluna Salário.

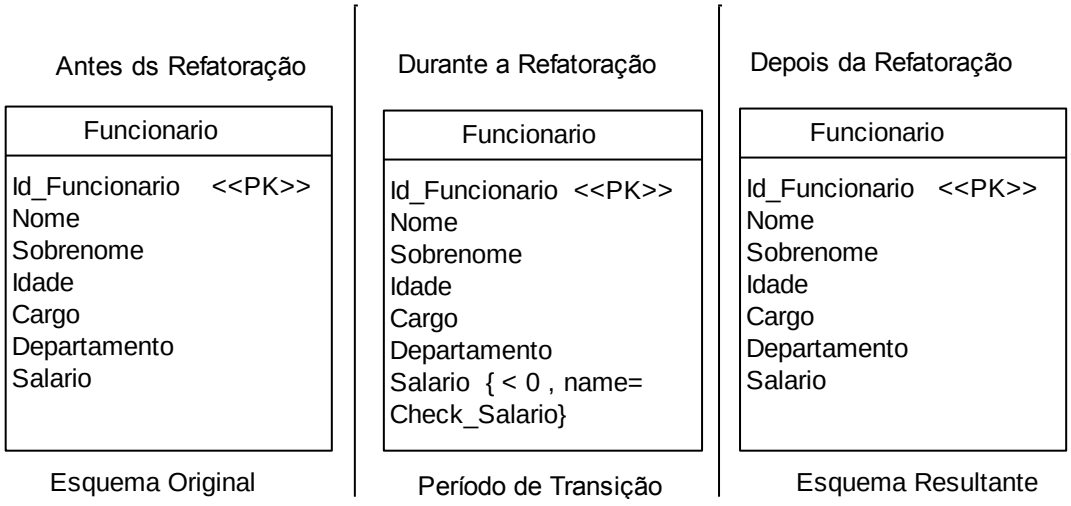

Figura 68. Refatoração de Qualidade de dados Introduzir valor comum

Fonte: Próprio Autor

7.3 Refatoração Integridade Referencial Exemplo

No caso de uso (exemplo) da figura 69 adicionar restrição de chave estrangeira, têm-se duas tabelas: empregado e dependentes. No esquema original, a tabela empregada possuía oito colunas: coluna ID_EMPREGADO chamada de chave primária, coluna CPF denominada chave estrangeira, a tabela dependentes com sete colunas, sendo a Id_Dependentes denominada como chave primária. No esquema final foi adicionada uma restrição de chave estrangeira na coluna ID_EMPREGADO. A tabela dependente, também está interligada a tabela empregada.

O script de banco de dados que foi utilizado para adicionar uma restrição de chave estrangeira para a coluna ID_EMPREGADO na tabela Dependentes foi:

ALTER TABLE Empregado ADD CONSTRAINT FK_Empregado_Dependentes FOREIGN KEY(Id_Empregado) REFERENCES Empregado;

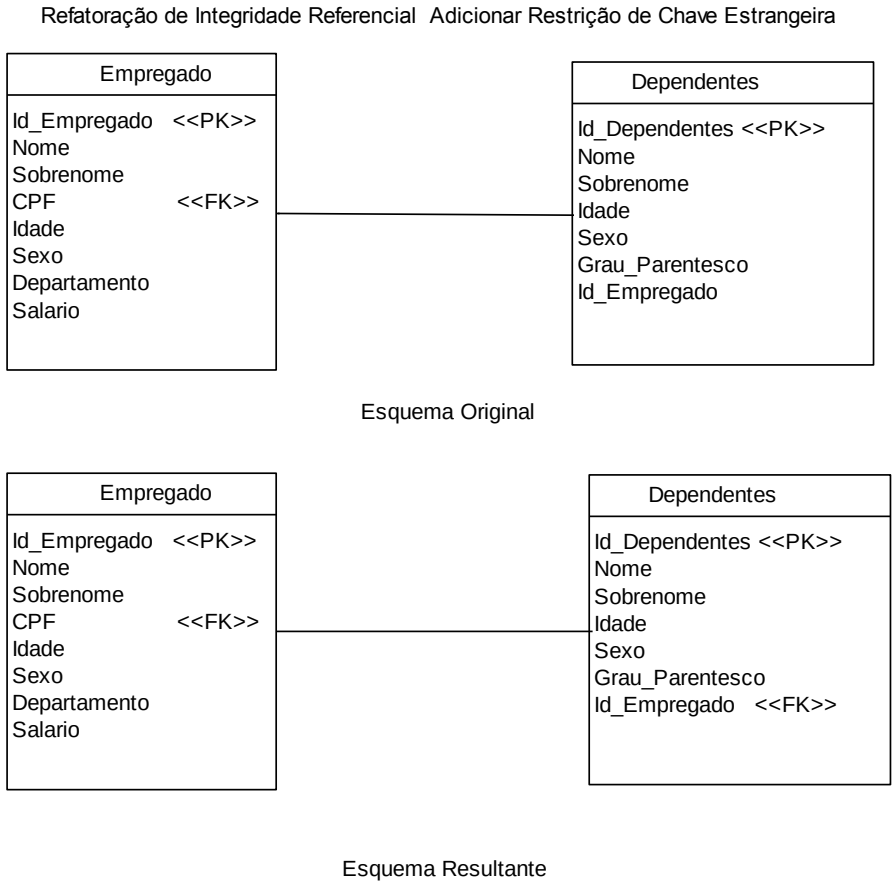

Figura 69. Refatoração de Integridade Referencial Adicionar Restrição de Chave Estrangeira

Fonte: Próprio Autor

7.4 Refatoração Arquitetural Exemplo

No caso de uso (exemplo) da figura 70 introduzir índice, tem-se a tabela Funcionário com oito colunas. A coluna Id_Funcionário foi denominada como chave primária da tabela no esquema original, no esquema final foi introduzido um índice para a coluna CPF, para a mesma ser um campo único como demonstra o script utilizado, e a representação da tabela ilustrado na Figura 70.

O script de banco de dados utilizado para introduzir um índice ao campo CPF da tabela Funcionário foi:

CREATE UNIQUE INDEX FuncionarioCPF ON Funcionario(CPF);

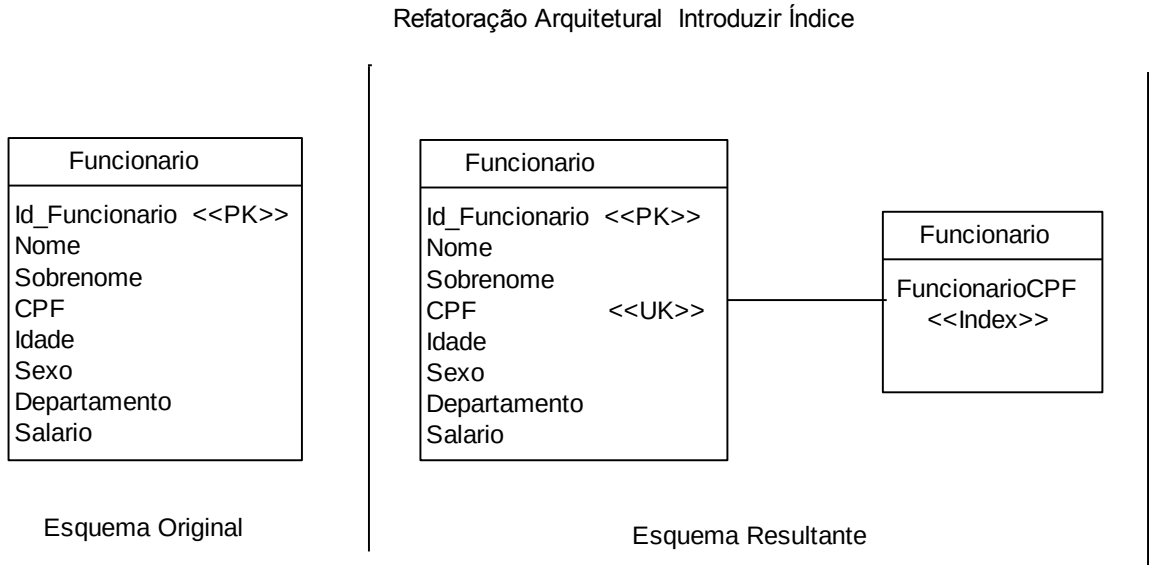

Figura 70. Refatoração Arquitetural Introduzir Índice

Fonte: Próprio Autor

7.5 Considerações Finais Sobre o Capítulo

Neste capítulo foram mostrados os casos de uso (exemplos) sobre as técnicas e ou refatorações de banco de dados, com suas respectivas ilustrações, também foram elucidados scripts utilizados para refatorar os exemplos, e as explicações sobre os mesmo. Na sequência foram demonstradas tabelas com os autores de trabalhos correlatos que utilizaram e citaram as categorias de refatoração de banco de dados e as refatorações. Scott W. Ambler juntamente com Sadalage, abordaram 70 técnicas de refatorações de banco de dados em seu livro "Refactoring Databases: Evolutionary Database Design" e também demonstradas através do

endereço eletrônico http://www.agiledata.org/essays/databaseRefactoringCatalog.html, e do processo de refatoração de banco de dados que também é disponibilizado no endereço eletrônico http://agiledata.org/essays/databaseRefactoring.html.

CONCLUSÃO

Sobre o trabalho conclui-se que o objetivo do tema em questão era o de apresentar os conceitos de refatoração de banco de dados, aplicando as respectivas técnicas de refatoração de banco de dados em casos de uso (exemplos), de quatro categorias de refatoração que são: estrutural, qualidade de dados, integridade referencial e, arquitetural, que tem por objetivo corrigir erros nos esquemas de banco de dados, melhorando a estrutura e o modelo dos mesmos.

A refatoração de banco de dados é composta por seis categorias que contêm 70 técnicas de refatoração, todas apontadas por Scott W. Ambler e Pramond J. Sadalage (2006), que no presente trabalho foram utilizadas seis técnicas de refatoração que são: eliminar coluna, eliminar tabela, renomear tabela, que pertencem á categoria estrutural, introduzir valor comum que pertence á categoria qualidade de dados, adicionar restrição de chave estrangeira que pertence á categoria integridade referencial, e, introduzir índice que pertence á categoria arquitetural, todas referenciadas nos trabalhos correlatos.

No capítulo 1 foram apresentados os conceitos sobre modelagem de dados e suas diversas etapas como: modelagem conceitual, lógica, e física. Todas discutidas por Chu Shao Yong, Peter Chen e demais autores pesquisados na Web. O trabalho apresentou os conceitos de refatoração refletidos e descritos por Martin Fowler (1999) em seu livro “Aperfeiçoando o projeto de código” juntamente com Kent Beck, John Brant e outros autores no capítulo 2.

No capítulo 3 foram apresentados os conceitos de refatoração de banco de dados descritas nos trabalhos correlatos, e também referenciados por outros autores que discutem o assunto em voga, no qual, apresentam as vantagens da refatoração, e as diferenças entre refatorar código fonte e código de banco de dados.

Em seguida no capítulo 4 foram apresentados exemplos sobre quatro categorias de refatoração de banco de dados que são: estrutural, qualidade de dados, integridade referencial, e arquitetural, e suas respectivas técnicas no capítulo 4 todas as mesmas disponíveis no livro de Scott W. Ambler e Pramond J. Sadalage, e, também se encontra disponível no endereço eletrônico: http://www.agiledata.org/essays/databaseRefactoringCatalog.html, os exemplos foram analisados e explicados pelo autor desse presente trabalho.

Na sequência no capítulo 5 foi contextualizado o processo de refatoração de banco de dados proposto por Scott W. Ambler em seu livro e que se encontra disponível no endereço eletrônico: http://agiledata.org/essays/databaseRefactoring.html, e o que torna a refatoração de banco de dados mais difícil que é o acoplamento, além disso, também são contextualizados

problemas sobre o processo de refatoração, apresentados na tese de Domingues (2014).

No capítulo 6 foram contextualizas seis tabelas contendo todas 70 técnicas de refatoração de banco de dados em suas respectivas categorias, os autores que as utilizaram em seus trabalhos correlatos, e também a quantidade de técnicas que cada autor utilizou em seu trabalho, sendo em estudo de caso, ou casos de uso (exemplos).

Foram demonstrados casos de uso (exemplos) no sétimo e último capítulo sobre as técnicas de refatoração de banco de dados de quatro categorias de refatoração que são: estrutural, qualidade de dados, integridade referencial, e arquitetural, os casos de usos (exemplos) os mesmos foram referenciados e utilizados nos trabalhos correlatos, os casos de uso (exemplos) foram implementados sem nenhuma ferramenta evolutiva ou de refatoração de banco de dados, foi utilizado o conceito de refatoração para a implementação e contextualização dos presentes casos de uso (exemplos), além do próprio banco de dados SQL Server 2012 para a visualização e implementação dos scripts SQL, para refatoração de banco de dados nos seguintes exemplos anteriormente implementados e exemplificados, se conclui que as técnicas de refatoração de banco de dados corrigem os erros no esquema de dados mantendo a semântica informacional e comportamental do banco de dados e informações contidas no mesmo, algumas das mesmas possui o conceito de normalização apresentado no capítulo 1 para manter os dados atualizados, como as técnicas de refatoração da categoria estrutural renomear coluna e renomear tabela.

Os casos de uso (exemplos) foram escolhidos, implementados e exemplificados por meio dos trabalhos correlatos, onde os mesmo são exemplificados e implementados utilizando casos de uso (exemplo) e estudo de caso, com as categorias de refatoração estrutural, qualidade de dados, integridade referencial e, arquitetural, e as respectivas técnicas de refatoração de banco de dados, principalmente as técnicas de refatoração de banco de dados do presente trabalho contextualizados nos casos de uso (exemplos) do capítulo 7.

REFERÊNCIAS

Alexandruk, M. (2011) Apostila de Modelagem de banco de dados, Disponível em: < http://pt.slideshare.net/fernandammachado14/apostila-modelagem-de-banco-de-dados>. Acesso em: 5 de Nov de 2015.

Catálogo de Refatorações de Banco de Dados, Disponível em: < http://www.agiledata.org/essays/databaseRefactoringCatalog.html>. Acesso em: 14 de Ago de 2015.

Chen, P. (1990) Modelagem de dados: A abordagem entidade-relacionamento para projeto lógico.

Dalpra, H. L. O; Araújo, M. A. P. (2012) Refatoração aplicadas a bancos de dados, Disponível em: < http://www.devmedia.com.br/refatoracao-aplicadas-a-bancos-de-dados-revista-engenharia-de-software-magazine-46/23815>. Acesso 10 em: Set de 2015.

Domingues, H. H. **Replicação assíncrona em bancos de dados evolutivos.** 2011. 150 f. Tese (Doutorado) Instituto de Matemática e Estatística da Universidade de São Paulo, 2011.

Domingues, M.B.P. **Um novo processo para refatoração de banco de dados**. 2014. 115 f. Tese (Doutorado) Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais, 2014.

Fowler, M; Beck, K; Brant, J; Opdyke, W; Roberts, D. (1999) Refatoração: Aperfeiçoando o projeto de código existente.

Introdução à Normalização de dados: um banco de dados "Melhores Práticas", Disponível em: < http://agiledata.org/essays/dataNormalization.html>. Acesso em: 25de Out de 2015.

Luis. (2008) Modelagem de dados: modelo conceitual, modelo lógico e físico, Disponível em: <http://www.luis.blog.br/modelagem-de-dados-modelo-conceitual-modelo-logico-e-fisico.aspx >. Acesso em 10 de Out de 2015.

Macedo, D. (2011) Modelagem Conceitual, Lógica e Física de Dados, Disponível em: < http://www.diegomacedo.com.br/modelagem-conceitual-logica-e-fisica-de-dados/>. Acesso em: 10 de Out de 2015.

Miguel, M. A; Araújo, M. A. P. (2011) Refatoração em Banco de Dados, Disponível em: < http://www.devmedia.com.br/refatoracao-em-banco-de-dados-sql-magazine-84/19065>. Acesso em: 20de Set de 2015.

O processo de refatoração de banco de dados: Estratégias para Melhorar a Qualidade de Banco de Dados, Disponível em: < http://agiledata.org/essays/databaseRefactoring.html>. Acesso em 12 de Set de 2015.

Neto, A. C. D. (2011) O processo de refatoração de banco de dados, Disponível em: < http://www.devmedia.com.br/o-processo-de-refatoracao-de-banco-de-dados-artigo-revista-sql-magazine-86/20400>. Acesso em: 13 de Set de 2015.

Yong, C. S. (1990) Banco de Dados: Organização sistemas e administração.