APLICAÇÃO DE REFATORAÇÃO EM UMA BASE DE DADOS

Cristiano Ramiris Diniz da COSTA[1]
Jayrson Sousa PARANA[2]
André Magno Costa de ARAUJO[3]
Cássio Cipriano NOGUEIRA[4]
Eliseu José dos SANTOS[5]

RESUMO: Os projetos de softwares evoluíram com bastante frequência ao longo do seu desenvolvimento, surgindo várias versões. A refatoração de banco de dados torna-se uma importância técnica aplicável à base de dados dessas aplicações, viabilizando a correta integração entre os softwares e essas mesmas bases de dados. As técnicas aplicadas às bases de dados podem ser em níveis estruturais, semânticos e comportamentais, tomando o cuidado para que as suas semânticas sejam preservadas. Vários benefícios podem ser adquiridos, tais como a qualidade da informação dos dados e diminuição de custos em manutenção e melhoria na velocidade de acesso aos dados. Durante a elaboração deste artigo, foi obtido uma diminuição no tamanho da base e aumento do nível de normalização através da aplicação das técnicas que são abordadas ao longo do texto.

Palavras Chave: Refatoração. Banco de Dados. Normalizar.

REFLECTION APPLICATION ON A DATABASE

ABSTRACT: Software projects have evolved quite frequently throughout their development, with several versions emerging. The database refactoring becomes a technical importance applicable to databases of these applications, making possible the correct integration between the software and these same databases. The techniques applied to databases can be at structural, semantic and behavioral levels, taking care that their semantics are preserved. Several benefits can be gained, such as the quality of the data information and the reduction of costs in maintenance and improvement in the speed of access to the data. During the elaboration of this article, a decrease in the size of the base and increase of the level of normalization was obtained through the application of the techniques that are approached throughout the text.

Keywords: Refactoring. Database. Normalize.

[1] Especialista em MBA de Administração de Banco de Dados; FAHESA/ITPAC. Email: ramiris.diniz@gmail.com.
[2] Docente do curso de Tecnológo em Análise e Desenvolvimento de Sistemas e Especialista em MBA de Administração de Banco de Dados; Faculdade de Ciências do Tocantins - FACIT; Araguaína - TO. Email: jayrsonjsp@gmail.com.
[3] Orientador, Mestre em Ciência da Computação pela Universidade Federal de Pernambuco. E-mail: amcaraujo@gmail.com
[4] Professor do curso de Análise e Desenvolvimento de Sistemas (ADS) da FACIT – Faculdade de Ciências do Tocantins; Graduado em Sistemas de Informação e especializado em MBA em Gestão de Tecnologias da Informação pela FAHESA/ITPAC – Instituto Tocantinense Presidente Antônio Carlos. E-mail: profcassiocipriano@outlook.com.
[5] Professor da Faculdade de Ciências do Tocantins FACIT. Curso Análise e Desenvolvimento de Sistemas. E-mail: admejs@gmail.com.

Introdução

As informações contidas dentro de uma base de dados são valiosas, e devido às constantes alterações de sistemas sem a devida preocupação na estrutura dessas bases, acarretam inconsistências como: a perda de desempenho e uma má qualidade dos dados armazenados. Poucos pesquisadores se propõe a discussão devido ás dificuldades de se aplicar a refatoração devido a acoplação dessas bases. Porém existe um catálogo que detalha setenta refatorações de banco de dados que são uma fonte valiosa na pesquisa e aprofundamento das técnicas atualmente disponíveis.

Na refatoração são realizadas aprimorações nos esquemas do banco, auxiliando na reparação de inconsistências existentes no projeto de banco de dados, e gerando um feedback ao desenvolvimento para evolução da base.

Propor a refatoração nos esquemas de banco de dados, melhorando seu projeto tendo como alvo o avanço do desempenho dessas informações, mantendo a semântica dos dados, no intuito de solucionar problemas com o desempenho desses sistemas e a qualidade dos dados que estão sendo armazenados no banco de dados.

Nesse sentido apresentamos esse trabalho que tem como objetivo realizar uma análise estrutural de uma base de dados legada propondo melhorias em sua estrutura. Através da análise dos relacionamentos e fluxo das informações, propopomos uma melhor compreensão dos elementos estruturais, otimizando a qualidade do banco de dados de forma segura.

Esse artigo está organizado da seguinte forma: Além da seção introdutória, a seção 1 aborda conceitos de Refatoração de Banco de Dados demonstrando suas técnicas e aplicações. Na seção 2 discutimos o contexto atual da base de dados a ser avaliada. Na Seção 3 encontra-se a análise e aplicação da refatoração na base de dados legada. Em seguida trazemos as concideraçõe finais.

1. Fundamentação Teórica: Refatoração de Banco de Dados

A refatoração de banco de dados pode ser vista como uma simples mudança no esquema do banco de dados produzindo melhorias para o seu projeto, preservando características de comportamento e semântica dos dados.

Segundo Ambler (2003, *apud* Araújo e Dalpra, 2012, p.2), a refatoração de um banco de dados consiste em uma melhoria junto à sua estrutura, onde se aprimora o projeto sem a adição de novas funcionalidades e sem alteração das funcionalidades já existentes ou da semântica informacional.

Todavia, refatorar não é um processo simples, pois é necessário um conhecimento aprofundado sobre os relacionamentos e os fluxos das informações da base proposta a qual se propõe refatorar.

Figura 1. Acesso ao banco com acoplamento simples.
Fonte: Ambler e Sadalage, 2006. - Adaptado

A Figura 1 é um exemplo de acoplamento simples. Quanto mais complexos forem os relacionamentos, mais oneroso será o processo de refatoração, como demonstrado na Figura 2, e em muitos casos o esforço não e recomendado pelas equipes de desenvolvimentos e DBAs.

Após o levantamento dessas informações é possível se determinar a viabilidade ou não de aplicar uma refatoração de banco de dados. Um esquema de banco de dados inclui tanto os aspectos estruturais como: definições de tabelas, tipos de dados, relacionamentos e restrições; como também aspectos funcionais, stored procedures e triggers. Porém, não é um processo simples, devido ao acoplamento do banco e suas aplicações. Quanto mais acoplado o banco, mais oneroso será esse procedimento.

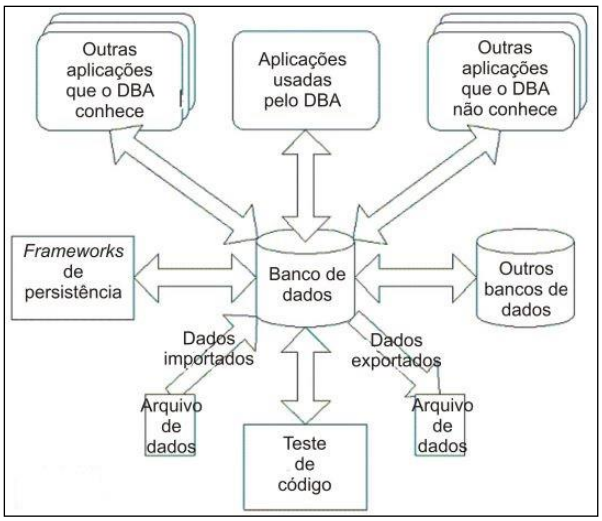


Figura 2. Acesso ao banco fortemente acoplado.
Fonte: Ambler e Sadalage, 2006. - Adaptado

É necessário preservar a semântica dos dados, pois as funcionalidades devem ser mantidas mesmo após a refatoração. Ou seja, todas as informações mantidas não podem perder o seu valor, elas devem continuar disponíveis sem que nenhuma consulta ou funcionalidade seja afetada.

Tudo isso se faz necessário para solucionar inconsistências ocasionadas devido às constantes mudanças que os softwares sofrem durante todo o seu processo de desenvolvimento e ao longo do seu ciclo de vida. Por ser um sistema complexo, essas alterações devem ser feitas durante um pequeno período de tempo, afim de corrigir as falhas das quais as bases não foram projetadas na fase de desenvolvimento ou mesmo por alguma melhoria proposta nas aplicações para aprimoramento de funcionalidades ou novas funções.

Essas alterações devem ser realizadas pelas equipes de desenvolvimento juntamente com os DBAs. Algumas características devem ser observadas e avaliadas antes de se optar por uma refatoração, são elas:

- Nível de acoplamento entre a base e a aplicação;
- Complexidade;
- Custo;
- Tempo;
- Conhecimento das equipes responsáveis;
- Alterações culturais da Empresa;

1.1. Uso da Refatoração em Banco de Dados

Neste contexto existem duas razões básicas onde é interessante realizar os conceitos de refatoração de banco de dados:

- Evoluir de forma segura os bancos de dados em produção.
- Apoiar o desenvolvimento de sistemas através de metodologias ágeis.

No primeiro caso, os softwares e sistemas evoluem ao longo do tempo surgindo várias versões a cada evolução, contudo os bancos de dados dessas aplicações necessitam acompanhar estas alterações para dar suporte a esta evolução, a qual não é uma preocupação da equipe de desenvolvimento, para solucionar e atender essa necessidade a refatoração de banco de dados é a forma segura de se realizar esta alteração mantendo a qualidade dos dados sem alterar a semântica dos mesmos.

Para o segundo caso, conforme Dias Neto (2011) os processos atuais de desenvolvimento de software, tais como RUP, XP e SCRUM, trabalham de forma evolutiva, e muitas vezes ágeis.

As bases de dados devem acompanhar a evolução dessas novas metodologias, necessitando adotar técnicas para trabalhar de tal maneira, portanto sendo uma opção de propor essa melhoria na base para que o banco possa acompanhar a evolução desses softwares, pois cada alteração pode modificar o desempenho e as necessidades de arquivamentos das informações, aumentando exponencialmente o tamanho dos dados da mesma, ou alterando a forma como os relacionamentos e fluxos dessas informações são interpretados pelo software. Essa preocupação se deve a maior necessidade de atender a modelagem das regras do negócio para qual o aplicativo foi desenvolvido.

1.2 Tipos de Refatoração

Segundo Ambler e Sadalage (2006), a refatoração em banco de dados pode ser dividida em seis categorias principais, a saber: Refatoração Estrutural, Refatoração de Qualidade de Dados, Refatoração de Integridade Referencial, Refatoração Arquitetural, Refatoração de Método e Refatoração de Transformação.

Com características bem definidas descritas a seguir:

- **Refatoração Estrutural**: consiste em alterações de um ou mais elementos. Podendo ser: Dividir Coluna, Dividir Tabela, Excluir Tabela, Excluir Visão, Introduzir Chave Substituta por Chave Natural, Substituir Coluna, Substituir campo complexo por Tabela e Substituir um para muitos com Tabela Associativa.

- **Refatoração de Qualidade de Dados:** baseia-se nas mudanças que aprimoram a qualidade da informação contidas no banco de dados, ou seja, buscam melhorar e/ou assegurar a coerência dos dados armazenados na base de dados. Sendo: Adicionar Tabela de Pesquisa, Aplicar Códigos Padrão, Aplicar Tipo Padrão, Consolidar a Estratégia-Chave, Excluir Restrição de Coluna, Excluir Valor Padrão, Excluir Restrição Não-Anulável, Introduzir Restrição de Coluna, Introduzir Formato Comum, Introduzir Valor Padrão, Fazer Coluna Não-Nula, Mover Dados, Substituir o Código do Tipo de Propriedade com bandeiras.
- **Refatoração de Integridade Referencial:** buscam garantir a remoção adequada de um registro em uma tabela que seja referenciado dentro de outra ou que uma linha que a existência de uma tupla ou linha que não seja mais necessária. Sendo: Adicionar Restrição de Chave Estrangeira, Adicionar gatilho para Coluna Calculada, Excluir Restrição de Chave Estrangeira, Introduzir Cascata, Introduzir Exclusão Rígida, Introduzir Exclusão Maleável, Introduzir Gatilhos para Histórico.
- **Refatoração Arquitetural:** proporcionam melhorias de uma forma global no banco. Propondo a substituição de programas ou scripts externos que interagem com a base de dados por programações internas ao banco, por exemplo stored procures. Podendo ser: Adicionar Métodos CRUD (Criar), Recuperar, Atualizar, Excluir, Adicionar Espelho de Tabela, Adicionar Leitura de Método, Encapsular Tabela com Visão, Introduzir Método de Cálculo, Introduzir Índice, Apresentar Tabela Somente Leitura, Migrar Método de Banco de Dados, Método de Substituição com Visão, Visão Substituir Método, Utilizar Fonte de Dados Oficiais.
- **Refatoração de Método:** consistem em buscar melhorar a qualidade de stored procedures, function ou trigger. Exemplo: Adicionar Parâmetro.
- **Refatoração de Transformação:** são mudanças que alteram a semântica do esquema do banco de dados, adicionando novos recursos a ele, por exemplo a inserção de uma nova coluna a tabela existente. Outros Tipos: Inserir Dados, Introduzir Nova Tabela, Introduzir Visão, Atualizar Dados.

1.3 Vantagens e Desvantagens

A aplicabilidade da refatoração de banco de dados pode proporcionar alguns aspectos de vantagens e desvantagens. A figura 3 demonstra algumas delas.

VANTAGEM	DESVANTAGEM
• Melhorar de forma significativa um projeto mal elaborado, impedindo que este deteriore ao longo do tempo. • Proporciona uma melhoria na qualidade e integridade dos dados. • Torna mais fácil e clara a tarefa de analise e entendimento do banco de dados, consequentemente pode-se ter maior velocidade de desenvolvimento e manutenção a menores custos.	• Tarefa difícil devido a problemas estruturais da base e aos próprios dados existentes no banco. • Base de dados em produção, proporcionando maior risco a integridade dos dados. • Quanto maior for o acoplamento entre a a base de dados e o software, script, framework, mais complexa será o processo de refatoração desta base.

Figura 3. Tabela Vantagens versus Desvantagens de Refatoração

Conforme Araújo e Miguel (2011, p. 3), ao se aplicar a refatoração em banco de dados, deve-se conhecer bem essa técnica e os benefícios alcançados com sua utilização, para que a mesma possa produzir realmente resultados satisfatórios.

Outros fatores também devem ser levados em conta, dominar as técnicas para refatorar e um passo fundamental para que se possa obter resultados satisfatórios. O nível de qualidade da refatoração pode ser expressa pela qualidade do projeto, quanto pior foi o desenvolvimento da base, melhor pode ser o resultado obtido através da refatoração, caso não haja inconsistência, o sucesso pode ficar pouco evidente. Mesmo com projetos altamente elaborados, pode ainda ser necessário que ela sofra alterações. Pois sabemos que os sistemas são vivos, e sofrem alterações, mudanças das quais não foram previstas a nível de projeto. Sendo assim justificável o uso das técnicas de refatoração para solucionar ou sanar inconsistências adquiridos sofridas pelas as alterações.

Os próprios dados armazenados dentro dos bancos, podem justificar aplicar uma das técnicas de refatoração de base de dados. Além de proporcionar um melhor controle, também permite um melhor aprimoramento da estrutura e um entendimento mais aprofundado da mesma. Proporcionando um planejamento mais adequado para as futuras necessidades que o sistema irá adotar.

Em um contexto geral, as vantagens são imensas, desde que atenda aos requisitos mencionados anteriormente. Alteração de dados, melhoria na velocidade das requisições ao banco, melhorias na estrutura do banco e permitir alterações futuras pelas aplicações.

1.4 O que não podemos considerar como Refatoração

É necessário comentar o que não estaria dentro do contexto de refatoração de base de dados. Segundo Dias Neto (2011, p. 3), uma pequena transformação em um esquema de banco de dados para estendê-lo, tal como a adição de uma nova coluna ou tabela, não é uma refatoração de banco de dados porque a mudança estende seu projeto.

Porém existe um tipo de refatoração chamada de Refatoração de Transformação que altera o esquema do banco de dados. Entretanto para ser considerada como refatoração, essa adição de nova coluna ou tabela precisa trazer dados de uma outra tabela ou coluna já existente na base para ser considerada como uma refatoração.

Tendo em vista que é uma pequena alteração no esquema do banco para aumentar a capacidade do mesmo por que essa alteração apenas muda a extensão do seu projeto inicial. Pois não visa melhorar seu projeto, mas adapta-lo a uma extensão. Pois a semântica inicial é alterada devido a essas adições de colunas ou tabelas. Alterar algumas tabelas podem até ser um processo similar, porém não seriam consideradas refatoração de base de dados.

2. Contexto Atual da Base de Dados

A base de dados utilizada neste trabalho atende uma empresa de departamentos tendo o seu nicho de mercado a venda de eletrodomésticos e eletroeletrônicos em geral, atualmente com um quadro de 300 colaboradores. Utilizando a tecnologia de SQL Server para a sua base de dados. Com os mais de 10 anos em atividade essa base tem como problemática o crescimento dos dados em volume, colunas sem valores atribuídos e colunas sem usabilidade.

A aplicabilidade proposta neste trabalho limita-se às tabelas de Clientes, Produtos, onde foi observado uma maior necessidade de aplicar os conceitos de refatoração de banco de dados.

3. Análise e Aplicação da Refatoração na Base de Dados Legada

3.1. Ciclos de vida

É possível estabelecer etapas gerais na execução das refatorações através de um processo que chamamos de ciclo de vida de refatoração. Em suma, os passos para uma aplicação de uma refatoração em banco de dados podem ser descritos como:

- Detectar a real necessidade de uma base de dados a ser refatorada;
- Identificar quais as refatorações que devem ser aplicadas e onde;

● Realizar as refatorações;

● Realizar testes.

Seguindo o modelo do ciclo de vida apresentado na Figura 4 (Ciclo de Vida da Refatoração).

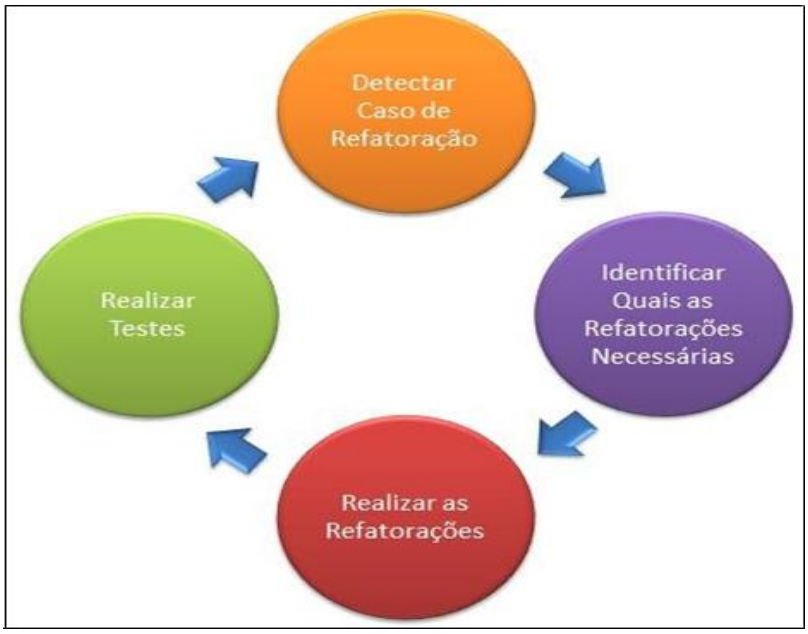


Figura 4. Ciclo de Vida da Refatoração. Fonte: Touré & Mens, 2003 – Adaptado

O ciclo garante que todos os fatores necessários, foram identificados antes do início da refatoração. Nos sub-tópicos a seguir será demonstrado todo o processo de refatoração

3.2. Tabela clientes

3.2.1. Cenário Tabela Cliente

A tabela clientes é utilizada para armazenar todos os dados referentes aos clientes da empresa, tais como filiação, telefones, cônjuge, endereço residencial, profissional e etc. A tabela possui 107 colunas sem nenhum relacionamento com outra entidade do banco. O grande número de colunas possibilita uma visualização confusa dos dados, impossibilitando à análise dos mesmos de forma clara e objetiva.

3.2.2. Problemas Identificados

A partir da análise realizada na tabela Clientes foram identificados alguns pontos dos quais poderíamos efetuar alterações que melhorasse a estrutura da tabela, como demonstrado na Figura 5.

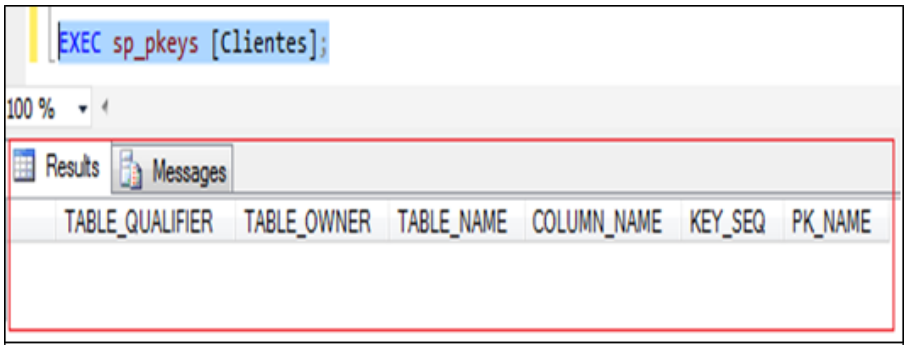
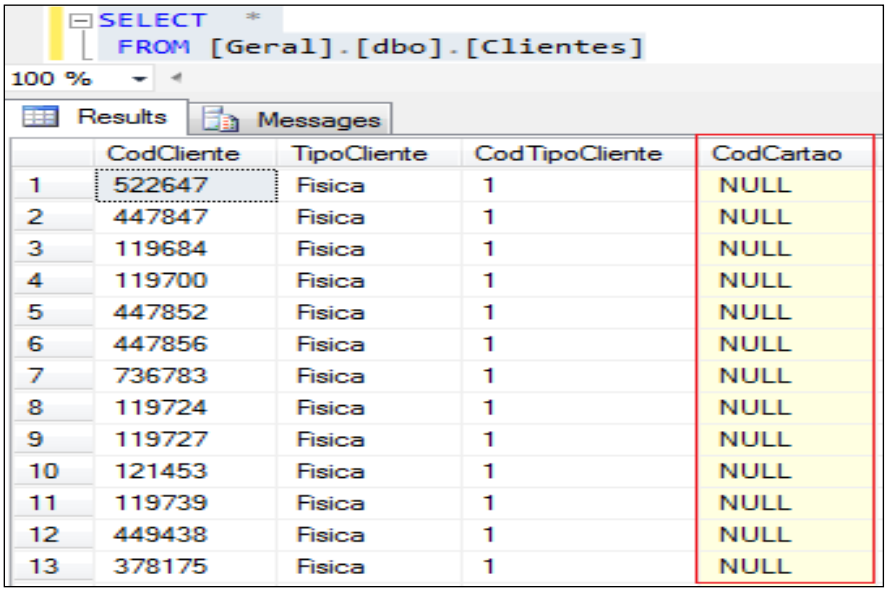


Figura 5. Tabela Clientes. Ausência de Chave Primária

Foi constatado a inexistência de primary key para a tabela Clientes, neste caso não seria possível garantir a unicidade de uma tupla.



	CodCliente	TipoCliente	CodTipoCliente	CodCartao
1	522647	Fisica	1	NULL
2	447847	Fisica	1	NULL
3	119684	Fisica	1	NULL
4	119700	Fisica	1	NULL
5	447852	Fisica	1	NULL
6	447856	Fisica	1	NULL
7	736783	Fisica	1	NULL
8	119724	Fisica	1	NULL
9	119727	Fisica	1	NULL
10	121453	Fisica	1	NULL
11	119739	Fisica	1	NULL
12	449438	Fisica	1	NULL
13	378175	Fisica	1	NULL

Figura 6. Tabela Clientes – Coluna CodCartao

Verificamos que em todas as 391118 tuplas, a coluna CodCartao apresenta valores do tipo NULL ou vazio. Conforme demonstrado na Figura 6.

```
SELECT  CodCliente, DescontinuadoMotivo,
 DescontinuadoData
 FROM [Geral].[dbo].[Clientes]
```

100 %

Results | Messages

	CodCliente	DescontinuadoMotivo	DescontinuadoData
1	522647	NULL	NULL
2	447847	NULL	NULL
3	119684	NULL	NULL
4	119700	NULL	NULL
5	447852	NULL	NULL
6	447856	NULL	NULL
7	736783	NULL	NULL
8	119724	NULL	NULL
9	119727	NULL	NULL
10	121453	NULL	NULL
11	119739	NULL	NULL
12	449438	NULL	NULL
13	378175	NULL	NULL

Figura 7. Tabela Clientes. Colunas Descontinuadas.

Conforme a Figura 7, percebemos a existência de colunas rotuladas como descontinuados, entretanto as mesmas ainda estavam presentes na base de dados contendo valores NULL ou vazios.

```
SELECT TOP 1000 [CodCliente]
      ,[FONE1]
      ,[TipoFone1]
      ,[FONE2]
      ,[TipoFone2]
      FROM [Geral].[dbo].[Clientes]
```

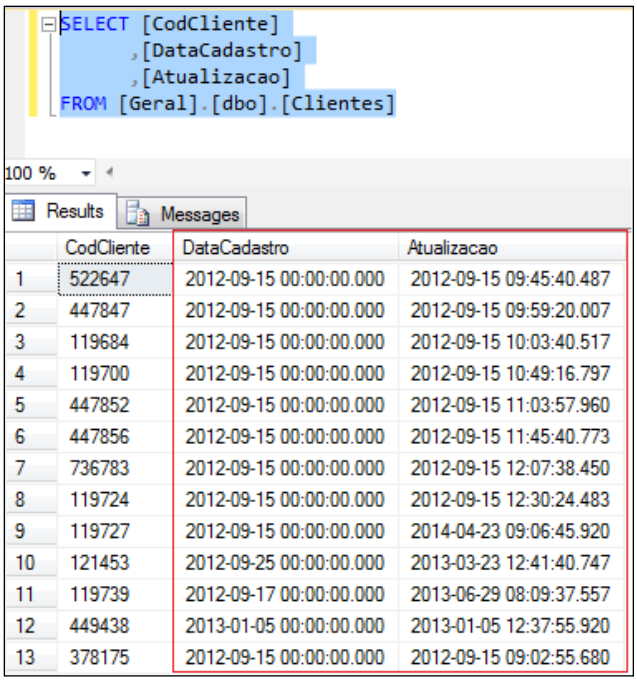
100 %

Results | Messages

	CodCliente	FONE1	TipoFone1	FONE2	TipoFone2
1	522647	NULL	0	NULL	0
2	447847	(63)3225-7573	0		0
3	119684		0		0
4	119700	(63)3415-4249	0		0
5	447852	(63)9968-2908	0	NULL	0
6	447856	NULL	0	NULL	0
7	736783	(63)3312-0309	0	(63)8406-8877	0
8	119724	(63)9257-3055	0		0
9	119727	(63)9208-9589	2	(63)9279-7866	2
10	121453	(63)3602-4091	0	(63)9244-3954	0
11	119739	(63)9995-4611	0	(63)9203-2384	0
12	449438	(63)8424-8299	0	NULL	0
13	378175	(63)8462-4207	0	(63)9276-1106	0

Figura 8. Tabela Clientes. Colunas FONE1, TipoFone1, fone2, tipofone2.

Conforme demonstrado na Figura 8, as colunas FONE1, TipoFone1, fone2, tipofone2 que armazenam dados referentes aos contatos dos clientes possuem tuplas NULL e vazias.



```
SELECT [CodCliente]
      ,[DataCadastro]
      ,[Atualizacao]
FROM [Geral].[dbo].[Clientes]
```

100 %

Results | Messages

	CodCliente	DataCadastro	Atualizacao
1	522647	2012-09-15 00:00:00.000	2012-09-15 09:45:40.487
2	447847	2012-09-15 00:00:00.000	2012-09-15 09:59:20.007
3	119684	2012-09-15 00:00:00.000	2012-09-15 10:03:40.517
4	119700	2012-09-15 00:00:00.000	2012-09-15 10:49:16.797
5	447852	2012-09-15 00:00:00.000	2012-09-15 11:03:57.960
6	447856	2012-09-15 00:00:00.000	2012-09-15 11:45:40.773
7	736783	2012-09-15 00:00:00.000	2012-09-15 12:07:38.450
8	119724	2012-09-15 00:00:00.000	2012-09-15 12:30:24.483
9	119727	2012-09-15 00:00:00.000	2014-04-23 09:06:45.920
10	121453	2012-09-25 00:00:00.000	2013-03-23 12:41:40.747
11	119739	2012-09-17 00:00:00.000	2013-06-29 08:09:37.557
12	449438	2013-01-05 00:00:00.000	2013-01-05 12:37:55.920
13	378175	2012-09-15 00:00:00.000	2012-09-15 09:02:55.680

Figura 9. Tabela Cliente. Coluna DataCadastro.

A Figura 9 apresenta a existência de algumas colunas da tabela Clientes utilizadas para armazenar dados do tipo *datetime* estavam com valores de tempo (hh:mm:ss:ms) zerados, ou seja, neste caso a hora não estava sendo utilizada como parâmetro(valor) relevante.

3.2.3. Solução Proposta

Para a tabela Clientes propõem-se utilizar os tipos de refatoração estrutural, qualidade de dados e integridade referencial, onde podemos empregar alguns de seus vários tipos, abaixo veremos quais foram os utilizados: Introduzir restrição de coluna: Este tipo de refatoração de qualidade de dados é empregada com a finalidade de garantir a integridade referencial da tabela Cliente identificando as tuplas da relação através de uma chave primária de conforme a Figura 10.

```
/*
Aplicando Refatoração de Integridade Referencial
Tabela Clientes.
*/
ALTER TABLE clientes
ADD CONSTRAINT pk_Clientes PRIMARY KEY(CodCliente);
```

Figura 10. Refatoração Qualidade de Dados.

Abaixo, na Figura 11, encontra-se a aplicação deste tipo de refatoração que favorece a implantação de relacionamento com outras tabelas e garantia de integridade relacional.

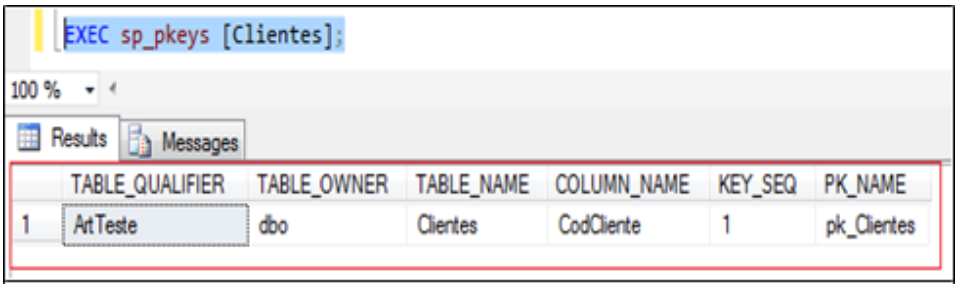


Figura 11. Refatoração Qualidade de Dados. Resultado com Chave Primaria

Excluir Coluna: A exclusão de coluna é um tipo de refatoração estrutural que consiste em remover uma coluna de uma tabela existente. Segundo Araújo e Dalpra (2012, p. 5) a principal razão para utilizar a exclusão de coluna é refatorar uma tabela de banco de dados, no caso da referida coluna não ser mais utilizada. No esquema de dados utilizado percebe-se que a coluna CodCartao da tabela Clientes não contém nenhum valor diferente de NULL ou vazio em seus 391188 registros. Conforme o procedimento executado na Figura 12.

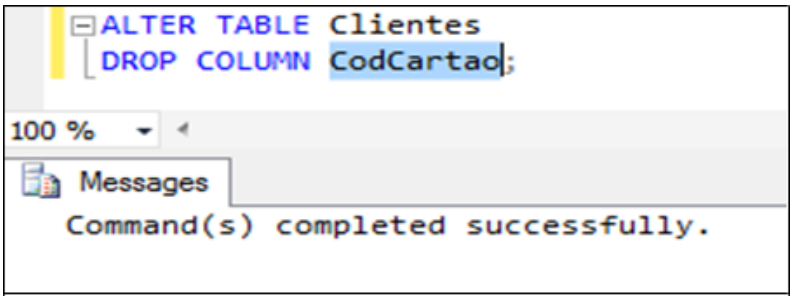


Figura 12. Refatoração – Excluir Coluna.

O procedimento Excluir Coluna também foi adotado para as colunas DescontinuadoData, DescontinuadoMotivo.

Mesclar Coluna: Esta técnica de refatoração estrutural de banco de dados consiste em mesclar duas ou mais colunas em uma única tabela. Conforme Araújo e Dalpra(2012, p.6) a técnica pode ser aplicada ao identificar colunas idênticas, as quais podem ser provenientes do fato de desenvolvedores distintos terem criado as colunas sem o conhecimento da duplicidade ou então quando os metadados que descrevem o esquema de tabela não estão disponíveis. Podemos observar esta ocorrência para as colunas FONE1, TipoFone1, fone2,

tipofone2. Conforme Figura 13, segue os procedimentos adotados para a execução desta refatoração.

```
-- 1. CRIAR TABELA telefonesClientes:

      CREATE TABLE telefonesClientes(
      idFone INT PRIMARY KEY IDENTITY (1,1),
      idcliente INT,
      telefone VARCHAR(14),
      TipoFone SMALLINT
      )

-- 2. CRIAR RELACIONAMENTO ENTRE Clientes E
     TelefonesClientes:

      ALTER TABLE telefonesClientes
      ADD CONSTRAINT fk_idCliente FOREIGN
      KEY(idCliente)
      REFERENCES Clientes (CodCliente);

-- 3. COPIAR TODOS OS DADOS DE Clientes PARA
     telefonesClientes

      INSERT INTO telefonesClientes
      (idcliente, telefone,TipoFone)
      SELECT CodCliente, fone1, TipoFone1
      FROM Clientes WHERE FONE1 is not null
      and fone1 != '';

      INSERT INTO telefonesClientes
      (idcliente, telefone,TipoFone)
      SELECT CodCliente, fone2, TipoFone2
      FROM Clientes WHERE FONE2 is not null
      and fone2 != '';

-- 4. CRIAR VIEW PARA telefonesClientes.

      CREATE VIEW vwTelefonesClientes
      AS select
      c.CodCliente as 'Codigo Cliente',
      tc.telefone as 'Telefone',
      tc.TipoFone as 'Tipo de Telefone'
      from Clientes c, telefonesClientes tc
      where c.CodCliente = tc.idcliente;
```

Figura 13. Refatoração Mesclar Coluna.

Foi criado uma nova tabela chamada de TelefonesClientes referenciando a tabela Clientes através de relacionamento por chave estrangeira, em seguida copiou-se os dados referente a FONE1, TipoFone1, fone2, tipofone2 para a nova tabela. Veja a tabela TelefonesClientes, Conforme Figura 14.

```
SELECT * FROM vwTelefonesClientes;
```

100 %

Results | Messages

	Codigo Cliente	Telefone	Tipo de Telefone
1	1	(63)9297-0134	1
2	2	(63)3448-1416	1
3	3	(63)(63)(063)	4
4	4	(63)3420-1154	4
5	6	(63)3414 4979	1
6	8	(63)9242-7665	3
7	9	(63)9239-5438	3
8	11	(63)3415-3485	2
9	12	(63)415 -5177	NULL
10	13	(63)(63)(063)	NULL
11	14	(063)414 -1873	NULL
12	16	(06)33414-1203	NULL
13	17	(63)(63)(063)	NULL

Figura 14. Tabela TelefonesClientes.

3.3. Tabela Produtos

3.3.1. Cenário Tabela Produtos

A tabela produtos é utilizada para armazenar dados referentes aos produtos oferecidos pela empresa, tais como código do produto, código de barras, preço, data de cadastro, etc., totalizando 152 colunas.

Percebe-se que com o intuito de facilitar a programação em localizar os dados de produtos em apenas uma tabela do banco consegue-se criar um cenário, onde os dados sugerem para uma inconsistência e má qualidade dos mesmos.

3.3.2. Problemas Identificados

A tabela produtos proporciona uma má compreensão dos dados devido ao grande número de registros com valores NULL ou vazio, o que dificulta o trabalha de análise e manipulação dos mesmos, além de apresentar definição de dados de forma a proporcionar o aumento da base em armazenamento. Um grande número de tuplas com valores vazios é apresentado nas colunas referente a códigos de barras dos produtos cadastrados. Nota-se que alguns produtos apresentam mais de um código de barra, porém para a maioria o mesmo não acontece conforme a Figura 15.

```
SELECT * FROM Produtos;
```

100 %

Results | Messages

	CodProd	Barra1	Barra2	Barra3	CodGrp1
1	000003	3	9,33		3
2	000004	000004			112
3	000010	0063			163
4	000015	15			163
5	000016	16			163
6	000019	19			2
7	000021	21			94
8	000022	0087			94
9	000023	0088			94

Figura 15. Tabela Produtos. Colunas Barra1, Barra2 e Barra3.

A definição do tipo de dados se faz de suma importância, uma vez que esta pode de certa forma onerar o banco de dados com relação ao armazenamento. No caso, observa-se que as colunas DataCadastro e Atualizado utilizam tipo de dados datetime, este tipo de dado tem armazenamento fixo de 8 bytes, por outro lado temos o tipo date, que tem armazenamento fixo de 3 bytes e atenderia muito bem neste cenário, pois o parâmetro de hora não é levado em questão. Veja os dados na Figura 16.

```
select CodProd, DataCadastro, Atualizado
from Produtos;
```

100 %

Results | Messages

	CodProd	DataCadastro	Atualizado
1	000003	2004-09-01 00:00:00.000	2012-12-08 00:00:00.000
2	000004	2004-09-01 00:00:00.000	2013-12-11 00:00:00.000
3	000010	2004-09-01 00:00:00.000	2013-07-12 00:00:00.000
4	000015	2004-09-01 00:00:00.000	2014-05-20 00:00:00.000
5	000016	2004-09-01 00:00:00.000	2014-05-21 00:00:00.000
6	000019	2004-09-01 00:00:00.000	2012-06-16 00:00:00.000
7	000021	2004-09-01 00:00:00.000	2013-04-30 00:00:00.000
8	000022	2004-09-01 00:00:00.000	2009-05-19 00:00:00.000
9	000023	2004-09-01 00:00:00.000	2009-05-19 00:00:00.000
10	000026	2004-09-01 00:00:00.000	2008-03-27 00:00:00.000
11	000028	2004-09-01 00:00:00.000	2014-05-17 00:00:00.000
12	000031	2004-09-01 00:00:00.000	2012-04-30 00:00:00.000
13	000035	2004-09-01 00:00:00.000	2004-09-01 00:00:00.000

Figura 16. Tabela Produtos. Colunas DataCadastro, Atualizado.

Ainda analisando as colunas CodGrp1, CodGrp2, CodGrp3, CodGrp4 da tabela Produtos fazem referência as tabelas SubGrupo1, SubGrupo2, SubGrupo3, SubGrupo4 respectivamente, observe a Figura 17.

```
SELECT CodProd, CodGrp1, CodGrp2, CodGrp3, CodGrp4
FROM Produtos;
```

100 %

Results | Messages

	CodProd	CodGrp1	CodGrp2	CodGrp3	CodGrp4
1	000003	3	39	52	2
2	000004	112	72	76	2
3	000010	163	40	44	2
4	000015	163	4	22	2
5	000016	163	4	22	2
6	000019	2	60	64	2
7	000021	94	2	20	2
8	000022	94	2	20	2
9	000023	94	2	20	2

Figura 17. Tabela Produtos – Colunas CodGrp1, CodGrp2, CodGrp3, CodGrp4.

Entretanto percebe-se a inexistência de relacionamento entre as tabelas, como pode-se verificar na Figura 18.

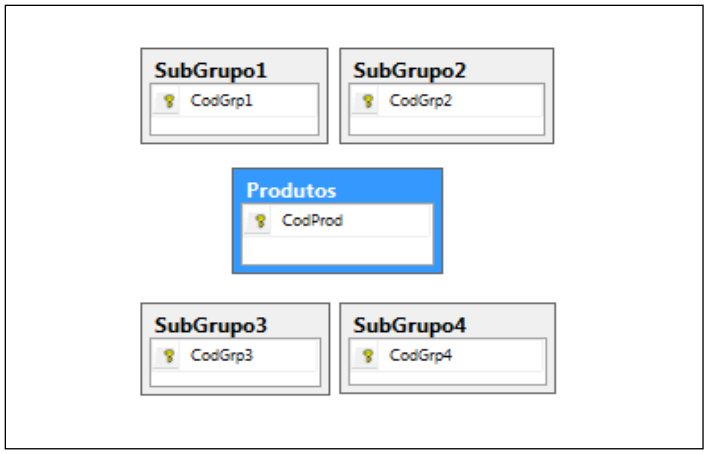


Figura 18. Tabelas Produtos, SubGrupo1, SubGrupo2, SubGrupo3, SubGrupo4.

3.3.3. Solução Proposta

Após a apresentação de alguns problemas é proposto a realização de uma refatoração estrutural e de integridade referencial:

- Mesclar Coluna: Propõem-se utilizar a mescla de coluna nesta tabela para as colunas Barra1, Barra2, Barra3. Estas colunas podem ser exportadas para uma nova tabela (codBarra) e realizar relacionamento desta com a tabela produtos, segue escopo para a refatoração conforme a Figura 19.

```
-- 1. CRIAR TABELA codigoBarra:

      CREATE TABLE codigoBarra(
      idBarra int primary key identity(1,1),
      codProduto varchar(6),
      codBarra varchar(25)
      )

-- 2. CRIAR RELACIONAMENTO ENTRE produtos e
--    codigoBarra:

      ALTER TABLE codigoBarra
      ADD CONSTRAINT fk_codigoBarra FOREIGN
      KEY(codProduto)
      REFERENCES produtos(codProd);

-- 3. COPIAR TODOS OS DADOS DE produtos PARA
     codigoBarra:

      INSERT INTO codigoBarra
      (codProduto, codBarra)
      select codProd, Barra1
      from Produtos where Barra1
      is not null and Barra1 != '';

      INSERT INTO codigoBarra
      (codProduto, codBarra)
      select codProd, Barra2
      from Produtos where Barra2
      is not null and Barra2 != '';

      INSERT INTO codigoBarra
      (codProduto, codBarra)
      select codProd, Barra3
      from Produtos where Barra3
      is not null and Barra3 != '';

-- 4. CRIAR VIEW PARA codigoBarra:

      CREATE VIEW vwCodigoBarras
      AS select
      p.CodProd as 'Codigo Produto',
      cb.codBarra as 'Codigo de Barras'
      from produtos p, codigoBarra cb
      where p.CodProd = cb.codProduto;
```

Figura 19. Refatoração Mesclar Coluna

Na Figura 20 apresenta a tabela codigoBarra através da *view* criada pela refatoração, observa-se de imediato que não há presença de registros com valores *NULL* ou vazios.

```
SELECT * FROM vwCodigoBarras;
```

100 %

Results | Messages

	Codigo Produto	Codigo de Barras
1	000003	3
2	000004	000004
3	000010	0063
4	000015	15
5	000016	16
6	000019	19
7	000021	21
8	000022	0087
9	000023	0088
10	000026	0100

Figura 20. View da Tabela CodigoBarra.

- Substituir Coluna: A substituição de coluna se faz necessário para realização da alteração do tipo de dados das colunas DataCadastro e Atualizado que foram definidos como datetime que ocupa um valor fixado em 8 bytes ao invés de date que ocupa valor fixado em 3 bytes, ou seja, uma redução de 62,5%. Segue o processo de alteração para coluna DataCadastro, conforme Figura 21.

```
-- 1.CRIAR COLUNA DataCadastro_new:

      ALTER TABLE PRODUTOS
      ADD DataCadastro_new DATE;

-- 2.COPIAR TODOS OS DADOS DE DataCadastro
--   PARA DataCadastro_new:

      Update produtos
      SET DataCadastro_new =  DataCadastro;

-- 3.RENOMEAR A COLUNA DataCadastro
--   PARA DataCadastro_old

      EXECUTE sp_RENAME
      'produtos.DataCadastro',
      'DataCadastro_old' , 'COLUMN';

-- 4.RENOMEAR A COLUNA DataCadastro_new
--   PARA DataCadastro:

      EXECUTE sp_RENAME
      'produtos.DataCadastro_new',
      'DataCadastro' , 'COLUMN';

-- 5.EXCLUIR A COLUNA DataCadastro_old:

      ALTER TABLE produtos
      DROP COLUMN DataCadastro_old;
```

Figura 21. Refatoração Substituir Coluna.

O processo similar foi adotado para a coluna Atualizado, na Figura 22 apresenta-se as colunas DataCadastro e Atualizado com os novos tipos de dados definidos pela refatoração:

```
select CodProd, DataCadastro, Atualizado
from Produtos;
```

100 %

Results | Messages

	CodProd	DataCadastro	Atualizado
1	000003	2004-09-01	2012-12-08
2	000004	2004-09-01	2013-12-11
3	000010	2004-09-01	2013-07-12
4	000015	2004-09-01	2014-05-20
5	000016	2004-09-01	2014-05-21
6	000019	2004-09-01	2012-06-16
7	000021	2004-09-01	2013-04-30
8	000022	2004-09-01	2009-05-19
9	000023	2004-09-01	2009-05-19
10	000026	2004-09-01	2008-03-27
11	000028	2004-09-01	2014-05-17
12	000031	2004-09-01	2012-04-30
13	000035	2004-09-01	2004-09-01

Figura 22. Tabela Produtos. Colunas DataCadastro e Atualizado.

- Introduzir Restrição de Chave Estrangeira: A principal razão de se aplicar esta refatoração é garantir que nas colunas CodGrp1, CodGrp2, CodGrp3, CodGrp4 da tabela Produtos persistirão valores válidos, ou seja, é uma forma de se aplicar regras de validação de dados. Neste caso adicionará relacionamento de chave estrangeira entre a tabela Produtos e as tabelas: SubGrupo1, SubGrupo2, SubGrupo3, SubGrupo4. Na Figura 23, segue o processo para execução desta refatoração.

```
-- 1.CRIAR RELACIONAMENTO ENTRE TABELA PRODUTOS E SubGrupo1
    ALTER TABLE PRODUTOS
    ADD CONSTRAINT FK_SUBGRUPO1 FOREIGN KEY(CodGrp1)
    REFERENCES SUBGRUPO1(CODGRP1)

-- 2.CRIAR RELACIONAMENTO ENTRE TABELA PRODUTOS E SubGrupo2
    ALTER TABLE PRODUTOS
    ADD CONSTRAINT FK_SUBGRUPO2 FOREIGN KEY(CodGrp2)
    REFERENCES SUBGRUPO2(CODGRP2)

-- 3.CRIAR RELACIONAMENTO ENTRE TABELA PRODUTOS E SubGrupo3
    ALTER TABLE PRODUTOS
    ADD CONSTRAINT FK_SUBGRUPO3 FOREIGN KEY(CodGrp3)
    REFERENCES SUBGRUPO3(CODGRP3)

-- 4.CRIAR RELACIONAMENTO ENTRE TABELA PRODUTOS E SubGrupo4
    ALTER TABLE PRODUTOS
    ADD CONSTRAINT FK_SUBGRUPO4 FOREIGN KEY(CodGrp4)
    REFERENCES SUBGRUPO4(CODGRP4)
```

Figura 23. Refatoração de Introduzir Restrição de Chave Estrangeira.

Na Figura 24, pode-se verificar o efeito da refatoração aplicada as tabelas envolvidas.

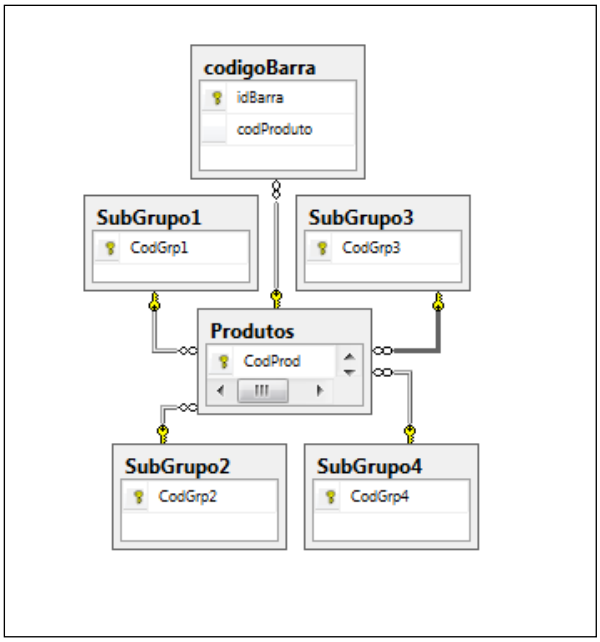


Figura 24. Resultado da Refatoração de Introduzir Restrição de Chave Estrangeira.

3.4. Resultados Obtidos

Os resultados obtidos com a aplicação da refatoração das tabelas Clientes e Produtos foram satisfatórios. Possibilitou a redução do espaço em disco através da alteração dos tipos de dados e exclusão de colunas sem utilização, redução de registros NULL e com valores vazios e implantação de integridade referencial.

A tabela Clientes após a realização das alterações propostas, obteve como resultado uma redução de aproximadamente 2,158%, ou seja, o espaço armazenado por esta tabela diminuiu 9,12 MB.

A Figura 25 apresenta o espaço ocupado pela tabela Clientes antes da aplicação da refatoração, observa-se que o espaço ocupado pelos dados era de 422440KB, veja abaixo.

```
USE Geral;
EXEC sp_spaceused clientes;
```

100 %

Results | Messages

	name	rows	reserved	data	index_size	unused
1	Clientes	391118	447448 KB	422440 KB	8 KB	25000 KB

Figura 25. Tabela Clientes - Espaço armazenado antes da refatoração.

Conforme apresenta a Figura 26 verificamos a redução do espaço ocupado pelos dados da tabela clientes após a refatoração.

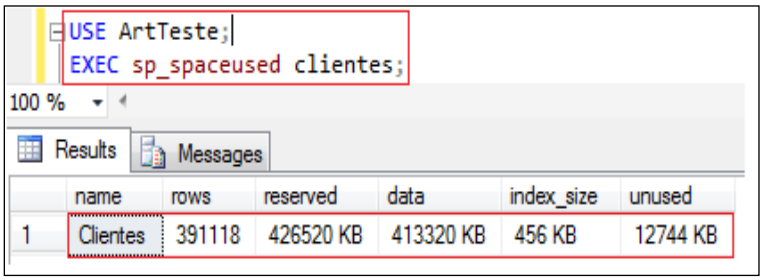


Figura 26. Tabela Clientes - Espaço armazenado após a refatoração.

Para o esquema de dados a refatoração proporcionou várias melhorias, tornando mais fácil o entendimento e o uso da base de dados, a implementação da mescla de coluna permitiu a criação de novas tabelas e consequentemente a integridade relacional entre elas, que antes se tratava apenas tabelas soltas sem relacionamentos, conforme demonstrado na Figura 27.

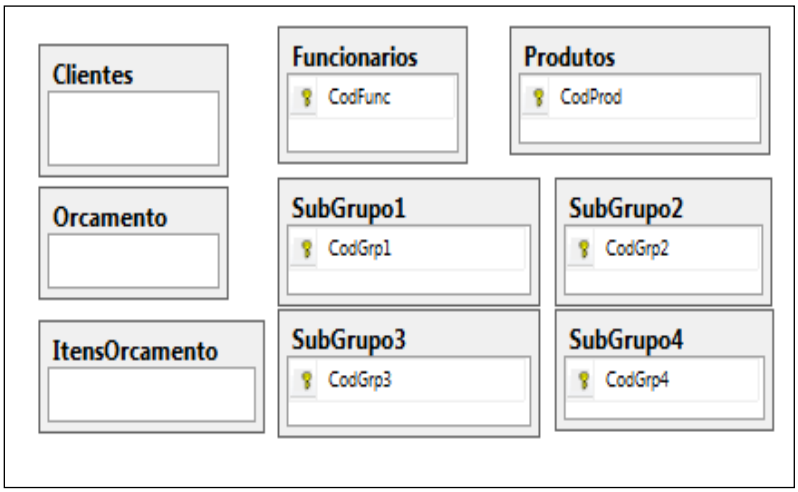


Figura 27. Esquema de dados antes da refatoração.

A Figura 28 apresenta o esquema de dados após a criação dos relacionamentos entre as tabelas propostas nas refatorações, permitindo assim visualizá-los de forma a perceber a normalização e um melhor entendimento do esquema de dados.

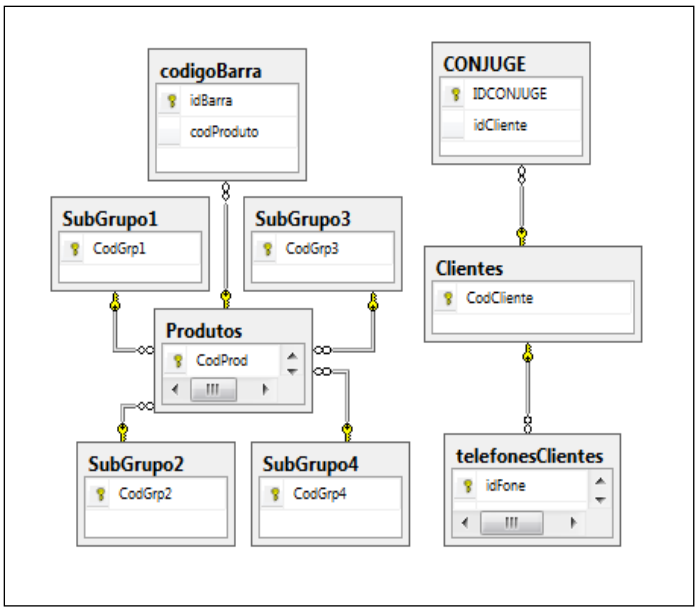


Figura 28. Esquema de dados após da refatoração.

A incidência de registros com valores NULL ou vazios foram reduzidos, minimizando a inconsistência e redundância dos dados, conforme demonstrado com a Figura 29.

```
SELECT TOP 1000 [idBarra]
      ,[codProduto]
      ,[codBarra]
  FROM [ArtTeste].[dbo].[codigoBarra]
```

100 %

Results | Messages

	idBarra	codProduto	codBarra
1	1	000003	3
2	2	000004	000004
3	3	000010	0063
4	4	000015	15
5	5	000016	16
6	6	000019	19
7	7	000021	21
8	8	000022	0087
9	9	000023	0088
10	10	000026	0100
11	11	000028	28
12	12	000031	0118
13	13	000035	0125

Figura 29. Tabela codigoBarra.

4. CONCLUSÃO

A refatoração de banco de dados é uma técnica que tem como objetivo prover melhorias as bases de dados em produção, pois possibilita as alterações desses esquemas de forma gradual e atender aos novos requisitos das aplicações.

Buscou-se neste artigo aplicar os conceitos estudados de algumas técnicas de refatoração de banco de dados, foi possível verificar que a refatoração é uma técnica viável e capaz de solucionar inconsistências de bases com um mal planejamento, assim provendo suporte para a implementação de recursos dos quais não foram projetados durante o processo evolutivo da base.

Através dela podemos normalizar as bases e os dados, reduzindo o tamanho das mesmas aplicando pequenas alterações, preservando a semântica das informações e provendo a segurança dos dados.

Ao final da aplicação das refatorações propostas na base de estudo, notou-se uma normalização da base, uma normalização na qualidade dos dados e uma redução no tamanho da mesma. Podemos assim então observar que houve uma melhoria na base, indo de encontro com o objetivo principal do uso de refatorações.

Dessa forma, mostra que a utilização de Refatorações de Banco de Dados prove melhorias nas bases, elevando o nível de segurança e qualidade dos bancos de dados.

5. REFERÊNCIAS

AMBLER, S. W.; SADALAGE, P. J. Refactoring Databases: Evolutionary Database Design. Boston: Editora: Addison-Wesley Professional, 2006. 41p.

ARAÚJO, M. A. P.; DALPRA, H. L. O. Técnicas de refatoração aplicadas a bancos de dados. Artigo Revista Engenharia de Software Magazine 46, 2012. Disponível em: http://www.devmedia.com.br/websys.5/webreader_print.asp?cat=48&artigo=4404&revista=impressao_46#a-4404.

ARAÚJO, M. A. P.; MIGUEL, M. A. Refatoração em Banco de Dados. Revista SQL Magazine 84, 2011. Disponível em: http://www.devmedia.com.br/websys.5/webreader.asp?cat=2&artigo=3223&revista=sqlmagazine_84#a-3223.

DIAS NETO, A. C. O processo de refatoração de banco de dados. Artigo Revista SQL Magazine 86, 2011. Disponível em: http://www.devmedia.com.br/websys.5/webreader.asp?cat=2&artigo=3461&revista=sqlmagazine_86#a-3461

TOM TOURWÉ & TOM MENS. Identifying Refactoring Opportunities Using Logic Meta Programming.Proceedings of the Seventh European Conference on Software Maintenance and Reengineering (CSMR ’03), 2003.