FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
CENTRO UNIVERSITÁRIO “EURÍPIDES DE MARÍLIA” – UNIVEM
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RENATA SARA DE SOUZA

APLICAÇÃO DE TÉCNICAS DE TESTE FUNCIONAL EM UM SISTEMA DE BANCO DE DADOS

MARÍLIA
2009

RENATA SARA DE SOUZA

APLICAÇÃO DE TÉCNICAS DE TESTE FUNCIONAL EM UM SISTEMA DE BANCO DE DADOS

Monografia apresentada ao Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha, para obtenção do Título de Bacharel em Ciência da Computação.

Orientador:
Prof. Ms. Paulo Augusto Nardi

MARÍLIA

2009

SOUZA, Renata Sara de

Aplicação de técnicas de teste funcional em um sistema de banco de dados/ Renata Sara de Souza; orientador: Prof. Ms. Paulo Augusto Nardi. Marília, SP: [s.n.], 2009.

51f.

Trabalho de Conclusão de Curso (Graduação em Bacharel em Ciência da Computação) – Curso de Bacharelado em Ciência da Computação, Fundação de Ensino "Eurípides Soares da Rocha", mantenedora Centro Universitário Eurípides de Marília – UNIVEM, Marília, 2009.

1. Engenharia de *Software* 2. Teste de *Software* 3. *Banco de Dados*

CDD: 005.74

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Renata Sara de Souza

APLICAÇÃO DE TÉCNICAS DE TESTE FUNCIONAL EM UM SISTEMA DE BANCO DE DADOS

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

Nota: 7,0 (sete)

Orientador: Paulo Augusto Nardi

1°. Examinador: Elvis Fusco

2°. Examinador: Fabio Lucio Meira

Marília, 02 de dezembro de 2009.

AGRADECIMENTOS

À DEUS, o meu melhor amigo, por me dar a vida, por me amar incondicionalmente, por permitir-me atingir mais um objetivo e me dar a vitória.

À minha Família abençoada por Deus.
Meus pais, Benedito e Eunice, meus irmãos, Wellington e Ellen, minha cunhada, Elisangela pelo amor, apoio, incentivo, confiança, cuidado e carinhos incessantes, e minha sobrinha Anne Caroline que com sua chegada deu mais alegria e sentido à minha vida.

Aos meus amigos da graduação, pelos momentos de descontração, sorrisos, expectativas, ansiedades e tensões vividas todos esses anos juntos.

Ao meu professor e orientador Paulo Augusto Nardi pelo incentivo, ensinamento e orientação.

À todos os professores que ao longo da graduação contribuíram de várias formas para a minha formação profissional.

Ao Centro Universitário Eurípides de Marília – UNIVEM, por fazer parte dessa formação.

"Porque a sabedoria deste mundo é loucura diante de Deus; porquanto está escrito: Ele apanha os sábios na própria astúcia deles".
1 Coríntios 3:19.

"Ninguém é tão grande que não possa aprender, nem tão pequeno que não possa ensinar".
Voltaire.

SOUZA, Renata Sara de. **Aplicação de Técnicas de Teste Funcional em um Sistema de Banco de Dados.** 2009 51f. Monografia (Graduação em Ciência da Computação) - Centro Universitário Eurípides de Marília - Fundação de Ensino Eurípides Soares da Rocha, Marilia, 2009.

RESUMO

Uma das fases para a obtenção da qualidade de um software é a execução do teste de software, pois podem existir ocorrências de falhas em relação à informação, que geram custos. Uma proposta de obtenção da qualidade é aplicar técnicas de teste funcional com base nas dependências de dados e integridades das informações geradas ao longo de toda aplicação. Neste trabalho foi aplicado um estudo de caso que aplica a técnica funcional de teste de software com base nas características do Banco de Dados Relacional. São apresentados os conceitos básicos do teste, resultados obtidos, bem como uma análise de resultados.

Palavras-Chave: Engenharia de *Software.* Teste de *Software*. Banco de Dados.

SOUZA, Renata Sara de. **Aplicação de Técnicas de Teste Funcional em um Sistema de Banco de Dados.** 2009 51f. Monografia (Graduação em Ciência da Computação) - Centro Universitário Eurípides de Marília - Fundação de Ensino Eurípides Soares da Rocha, Marilia, 2009.

ABSTRACT

One of the stages to obtain the quality of software is the implementation of software testing, because there can be instances of failures in relation to information which generates costs.A proposal for achieving quality is to apply techniques of functional testing based on data dependencies and integrity of information generated throughout the entire application. At this work we applied a case study that applies the technique of functional test software based on the characteristics of a relational database. We present the basics of testing, the obtained results as well as an analysis of results.

Keywords: *Software* Engineering. *Software* Testing. Database.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABDR	Aplicação de Banco de Dados Relacional
BD	Banco de Dados
BDR	Banco de Dados Relacional
DDL	Linguagem de definição de dados
DML	Linguagem de manipulação de dados
ER	Elementos Requeridos
IBM	International Business Machines
OO	Orientação a Objetos
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
SQL/DS	Structured Query Language/Data System
UML	Unified Modeling Language (Linguagem de Modelagem Unificada)
VV&T	Validação, verificação e teste

SUMÁRIO

INTRODUÇÃO

A Engenharia de Software aprimora a praticidade, organização e qualidade de um software, pois possui fases, métodos, modelos, padrões, entre outras características de desenvolvimento, que auxiliam o desenvolvedor na construção de um software (PEDRYCZ; PETERS,2001)

A garantia para que um software seja desenvolvido com qualidade é haver um conjunto de técnicas aplicadas no seu desenvolvimento. A VV&T (validação, verificação e teste de software) são algumas dessas técnicas utilizadas para se garantir que um software obtenha qualidade, pois objetiva a diminuição de erros (MALDONADO, 2004). Essas atividades devem estar contidas no desenvolvimento de um sistema, para que não haja persistência de defeitos e para que sejam corrigidos os erros antes do uso do produto (DELAMARO, 2007).

O teste de software surgiu com o crescimento dos sistemas de software e com os custos relacionados às falhas que os mesmos possuem. Gastar de 30 a 50% do tempo de um projeto em teste é comum para os desenvolvedores de software, pois existem sistemas em que é obrigatório ter uma precisão nas suas informações, como em sistemas de aviões, lançamentos de foguetes, caldeiras, entre outros que envolvem riscos para as pessoas (PRESSMAN, 1995).

No desenvolvimento de um sistema são utilizados critérios, técnicas e diversos meios para se garantir que um software não contenha erros, mas realizar a atividade de teste de software é necessário para se eliminar os testes que são persistentes (MALDONADO, 1991).

O desenvolvimento de um software fica submetido a erros ou falhas, pois depende da prototipação, interpretação de quem constrói o sistema (DELAMARO, 2007).

O teste funcional é uma das técnicas que é mais praticável tendo em vista que muito software é levado para teste em fases que já se encontram em execução pelo cliente ou já foram implantados. Muitas vezes os desenvolvedores não disponibilizam o código fonte, sendo uma opção, realizar o teste a partir da técnica funcional (baseado na especificação). Muitas tentativas de autores em melhorar o número de critérios e tentativas de exercitar classes diferentes de erros vêm diretamente melhorando o nível de teste funcional em diferentes tipos de software.

Existem trabalhos já desenvolvidos que os assuntos abordam o teste funcional, como Gonçalves (2003) que apresenta as técnicas de testes incluindo o teste funcional nas ABDR.

Gonçalves (2003) propõe uma aplicação dos critérios de teste funcional para ser aplicado em Sistemas que acessam Banco de Dados.

Modesto (2006) aborda a técnica de Teste Funcional, criando casos de testes baseados nos diagramas da Linguagem de Modelagem Unificada (UML), visando explorar os diagramas de classes, casos de usos e interação para extrair elementos de testes funcionais para serem aplicados em um programa de Orientação a Objetos (OO).

Souza (2008) aborda as técnicas de Teste Funcional em Aplicação de Banco de Dados Relacional, executando casos de testes baseados no diagrama da UML, introduzindo os critérios de Teste Funcional Intra-classe e Teste Funcional Inter-classe.

Neste trabalho serão testados os tipos de restrições de integridades no Banco de Dados apresentadas e especificadas no Capítulo 2, em que serão inseridos comandos Structured Query Language (SQL) testando se esses tipos de restrições existem no Banco de Dados.

No Capítulo 2 são apresentados e descritos as restrições de integridades que um banco de dados pode possuir e os conceitos de SQL, apresentando o conceito de Linguagem de manipulação de dados (DML).

No Capítulo 3 é apresentado os conceitos de teste de banco de dados em Aplicação de Banco de Dados Relacional (ABDR) e os critérios de teste funcional utilizado neste trabalho.

O Capítulo 4 mostra o estudo de caso realizado a partir de um Banco de Dados (BD) e seus elementos requeridos.

O Capítulo 5 aborda os casos de testes aplicados e a análise dos resultados obtidos pelo testes exercitados. Após esse capítulo é apresentada a conclusão.

CAPÍTULO 1 - Conceitos de Teste de Software

Existe um modelo de processo de teste de Software (Figura 1), em que para a realização de testes é preciso criar casos de testes especificando as entradas de dados que devem ser implantadas no sistema, cada entrada gera uma determinada saída com declarações do que está sendo testado no sistema e os dados de testes são as entradas geradas automaticamente. A saída é analisada e comparada com o resultado esperado ou previsto por quem entende o que o sistema executa. Se apontar a presença de defeitos é depurado e feito correção do mesmo (SOMMERVILLE, 2008).

Sommerville (2008) relata que para realizar um projeto de casos de teste é necessário especificar os casos de entradas e saídas esperadas que será testado no sistema, com o objetivo de criar um conjunto de casos de teste que encontre erros no programa e que verifique se o sistema atende aos requisitos. Já Spoto (2000), diz que realizar um projeto de casos de teste é projetar um dado de entrada para um programa e a saída esperada para a determinada entrada, com o objetivo de se ter uma grande probabilidade de encontrar a maioria dos erros ainda não revelados com um tempo e um esforço mínimo.

Segundo Delamaro (2007), o teste de software é uma atividade que executa o programa utilizando entradas e verifica se o programa está reagindo como o esperado, se algo não condiz com o esperado, quer dizer que algo foi encontrado, como erros ou defeitos.

Nardi (2006) cita:

> "Geralmente é o próprio testador que a partir dos requisitos do software do estudo do programa, ou trecho de programa, consegue determinar previamente qual a saída esperada para determinado dado de entrada."

Segundo Pressman (1995) teste de software é um elemento crítico para se garantir a qualidade de um sistema e representa a última revisão de especificação, projeto e codificação.

Segundo (DOMINGUES, 2002 apud VICENZI, 2000), não se é possível através do teste, mostrar que um sistema está correto, mas sim garantir que o sistema esteja desempenhando as funções especificadas e mostrar que o produto tende a ter qualidade.

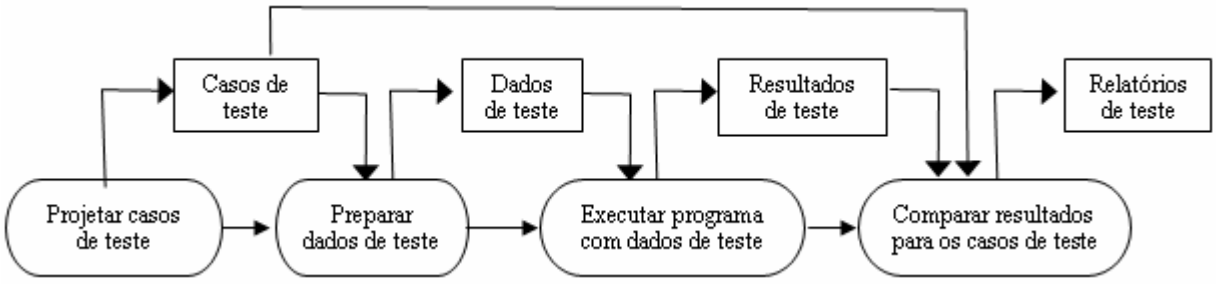


Figura 1 – Modelo de Processo de testes de Software (SOMMERVILLE, 2008).

1.1 Fases do Teste

Segundo Delamaro (2007) para a realização das atividades avaliadoras de teste, existe uma divisão em três fases distintas: teste de unidade, teste de integração ou incremental e teste de sistema.

Segundo Nardi (2006) a aplicação dos testes pode ser feita com os procedimentos separadamente (refere ao teste de unidade), com procedimentos relacionados (teste de integração) ou no sistema todo (teste de sistema).

Segundo Pressman (2006) o Teste de Unidade busca encontrar defeitos em unidades do sistema, ou seja, procedimentos, funções ou métodos, em cada módulo. De acordo com Delamaro (2007), essa fase do teste objetiva menores unidades de um sistema, como sendo, procedimentos, classes, funções ou métodos. Já Myers (2004), o teste de unidade é feito de modo individual em sub-rotinas, procedimentos ou subprogramas de um programa, analisando que é preciso existir um controle sobre cada parte do programa.

Na fase de teste de integração Sommerville (2004) define que após ser testados os componentes do programa de forma individual, deve haver uma integração para se criar um sistema completo, ou seja, devem–se fazer testes a partir das interações de componentes disponíveis. A principal dificuldade é a localização da origem dos erros que são descobertos. De acordo com Myers (2004), o teste de integração seria como uma fase que testa o programa a partir dos testes unitários, combinando-se os módulos.

O teste de sistema, segundo Maldonado (2004), é uma etapa após a fase de integração, que tem por objetivo a identificação de erros de funções e características de desempenho. De acordo com Pressman (2006), é a fase de verificação e/ou validação do sistema, identificando requisitos funcionais e requisitos não funcionais.

1.2 Técnicas e Critérios de Teste

Testes são feitos através de várias técnicas como o teste funcional (teste de caixa preta), o teste estrutural (teste de caixa branca) e outras técnicas não apresentadas neste trabalho. Os testes com seus dados de teste visam analisar as saídas do software e seu comportamento perante aos dados inseridos, mostrando se o resultado está sendo compatível com o esperado (SOMMERVILLE, 2004). No teste de caixa preta ou teste funcional são realizados testes no banco de dados com o programa em execução, através dos requisitos funcionais em que condições de entradas serão inseridas para obtenções de resultados.

Segundo (DOMINGUES, 2002), o que faz diferença das técnicas apresentadas é onde a informação se origina, pela qual é utilizada para os conjuntos de casos de teste. E os critérios são utilizados para avaliar e gerar esses conjuntos de casos de teste.

Apresentam-se nesta seção 1.2.1, o teste estrutural e na seção 1.2.2 o teste funcional.

1.2.1 Teste Estrutural

O teste de caixa branca, também chamado de teste estrutural, é uma técnica realizada diretamente na estrutura interna do programa, ou seja, no código fonte onde os testes serão feitos em cada procedimento ou função da estrutura do programa (PRESSMAN,1995).

No teste de caixa preta, para o testador não é necessário possuir a parte lógica do sistema, mas no teste de caixa branca o testador deve possuir a estrutura do sistema, ele deve ter uma transparência do sistema para desenvolver seu trabalho (MYERS, 2004).

Segundo Nardi (2006), a criação de representações gráficas, grafos de programa é uma maneira de se analisar o fluxo de controle do código de forma mais explícita e também são utilizados como base para os critérios de teste estrutural.

Segundo Sommerville (2004), os testes estruturais são aplicados em unidades de programas como sub-rotinas, onde o testador tem acesso e conhecimento da estrutura do programa e de sua implementação, pode analisá-lo possibilitando utilizar os conhecimentos sobre o código para derivar dados, podendo analisar quantos casos de teste podem ser gerados. A figura 2 mostra o fluxograma do teste estrutural.

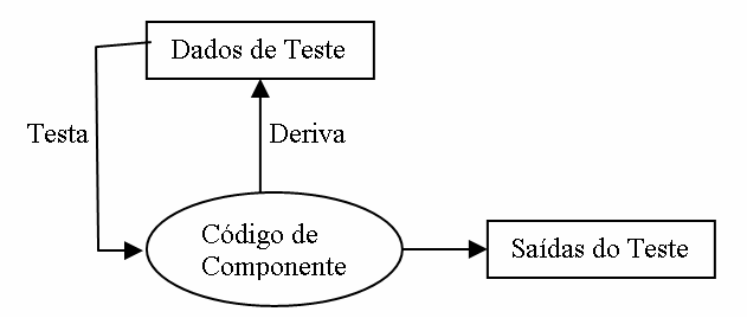


Figura 2 – Teste Estrutural (SOMMERVILLE, 2004).

1.2.2 Teste Funcional

Segundo Sommerville (2004), no teste funcional o testador se preocupa com a funcionalidade do sistema e não como o sistema foi implementado, pois se estuda o comportamento do software apresentando as entradas ao sistema e analisando as saídas relacionadas.

O testador deve saber quais são as funções específicas do *software*, a partir disso o teste é executado através de entradas inseridas e saídas produzidas comparando como as saídas esperadas, visando demonstrar que o sistema possui funções que são operacionais, não sendo necessário possuir conhecimento da estrutura lógica do sistema (PRESSMAN, 1995).

No teste de caixa preta, não é possível sem a estrutura lógica saber como o sistema se comporta, então para se detectar erros no software é necessário que haja diversas entradas no sistema, possibilitando criar várias maneiras de casos de teste, ou seja, deve-se fazer uma atividade denominada pelos profissionais como teste de exaustão (MYERS, 2004).

Segundo Pressman (1995), o teste de caixa preta tem uma abordagem e critérios, que procura descobrir erros distintos ao do método de teste de caixa branca. E são eles:

- Funções incorretas ou ausentes;
- Erros de Interface;
- Erros nas estruturas de dados ou no acesso a Bancos de Dados externos;
- Erros de desempenho;
- Erros de inicialização e término.

O teste de caixa preta (teste funcional) utiliza vários critérios, dois deles são o Particionamento em classe de equivalência e Análise do valor limite.

Particionamento em classes de equivalência: Segundo Pressman (2006), o particionamento em classes de equivalências define classes de equivalências para entrada de dados. As classes de equivalências são como uma tabela que existe condições de entradas que identifica o que será testado, e para essas condições de entradas existem algumas definições:

- Se a entrada utiliza intervalos define-se uma classe válida e duas inválidas, a classe válida é um valor que atende ao valor limite do intervalo e as duas classes inválidas seria um valor abaixo ao valor do limite inferior e um valor acima ao valor do limite superior;
- Se a entrada utiliza um valor determinado sem estar em intervalos, então se define uma classe válida com o determinado valor e duas classes inválidas, em que coloque um valor menor que o determinado e um valor maior que o determinado, que ambos serão inválidos;
- Se a entrada utiliza um elemento de conjunto defini uma classe válida que pertence ao conjunto e uma inválida que não pertence ao conjunto;
- Se utilizar booleano define-se uma classe válida que receba valor V ou F e uma inválida.

Um exemplo, considerando um cadastro de uma universidade em que o sistema possui um código para cada estudante que é o Registro Acadêmico (R.A), esse código possui até seis dígitos e esses dígitos são formados somente por números. Os horários de aula dos alunos é somente das 08:00 às 12:00 horas e das 14:00 às 17:00 horas, pois a aula é em período integral e a série em que o aluno está é até a 6ª série e só pode ser um dígito entre 1 e 6. Temos as classes de equivalência ilustrada na Tabela 1.

Condições de entrada	**Classes válidas**	**Classes inválidas**
Tamanho (t) do identificador	1<= t <=6 (1)	t<1 e t >6 (2) (3)
t contêm somente números	Sim (4)	Não (5)
Horário(h) da aula	08:00<=h<=12:00 (6)	12:00<=h<=14:00 (7)
Horário(h) da aula	14:00<=h<=17:00 (8)	17:00<=h<=08:00 (9)

h contém somente 5 dígitos	Sim (10)	Não (11)
Série (s) do identificador	1<= s <=6 (12)	s<1 e s >6 (13)
S contém somente 1 dígito	Sim (14)	Não (15)

Tabela 1- Exemplos de Classes de Equivalência

Tendo definido as classes de equivalência, podem-se criar os casos de testes:

T_0={(123, Válido), (,Inválido), (1582867, Inválido), (1a2, Inválido), (11:00,Válido),
(1, 4) (2,5) (3) (4) (6,10)
(13:00, Inválido), (15:00, Válido), (05:00, Inválido), (9:00,Inválido), (4, Válido)
(7) (8,10) (9) (11) (12,14)
(7, Inválido), (10, Inválido)}
(13) (15)

Temos que o primeiro caso de teste (123) é Valido, pois atinge duas classes válidas das classes de equivalência (1,4), o terceiro caso de teste (1582867) é inválido, pois atinge uma classe inválida (2).

Análise do valor limite: a análise do valor limite deve ser complementar ao da classe de equivalência, em vez de selecionar qualquer elemento da classe de equivalência, os casos de testes são feitos nas extremidades concentrando-se os dados na entrada e na saída, em que possivelmente pode ter grandes chances de encontrar erros, não encontrados com o particionamento em classes de equivalência (PRESSMAN, 1995).

De acordo com Pressman (1995) as diretrizes para as condições de entrada na análise do valor limite são semelhantes às do critério de Particionamento em classes de equivalência:

- Se uma condição de entrada especificar um limite de intervalo pelos valores a e b, os casos de testes devem ser desenvolvidos com valores a e b, logo acima e logo abaixo de a e b.
- Se uma condição de entrada especificar vários valores, são desenvolvidos casos de testes que explorem os números mínimos e máximos. Testam-se também os valores logo abaixo e logo acima do mínimo e do máximo.
- Aplique as diretrizes acima às condições de saída.

- Se as estruturas internas de dados do programa tiverem limites pré-definidos, assim como uma estrutura de dados, é necessário desenvolver casos de testes que atendam aos limites pré-determinados.

Segundo (GONÇALVES, 2003 apud PAULA FILHO, 2001), como análise de Valor Limite explora as extremidades, então permite detectar prováveis erros como: Valores abaixo do valor mínimo; Valores acima do valor máximo; Arquivo vazio; Cálculos que se efetuados ocasionam ‘*overflow’* ou ‘*underflow*; Erros no primeiro, no último registro.

As técnicas de teste objetivam avaliar a qualidade do teste e a diferença entre essas técnicas é na origem da informação utilizada e na construção dos casos de teste (MALDONADO, 1991).

O teste de caixa preta ou teste funcional é a técnica utilizada nesse trabalho onde são realizados testes no banco de dados com o programa em execução, através dos requisitos funcionais em que condições de entradas serão inseridas para obtenções de resultados.

CAPÍTULO 2 – Banco de Dados

Atualmente as empresas trabalham com software para administrar seus negócios. Os dados referentes a elas precisam ser armazenados de alguma maneira quando precisar obtê-los é necessário que seja de uma maneira fácil e precisa. Por exemplo, uma empresa que trabalha com clientes tem que possuir os dados principais de seu cliente como nome, telefone, CPF ou CNPJ entre outros, sendo necessário que esses dados sejam acessados de uma maneira prática.

Antigamente os dados eram armazenados em arquivos, o que dificultava muito a questão do custo e da eficiência. Ao longo dos anos foi pesquisado algo que obtivesse um resultado mais eficaz, criou-se então um repositório de dados chamado de banco de dados. Os bancos de dados atuais armazenam até mesmo imagens.

Elmasri e Navathe (2005), explicam que Banco de Dados é uma coleção de dados com relacionados, ele é projetado, construído e são inseridos dados nele. Os dados são fatos que podem ser gravados e possui um significado implícito. O Banco de Dados pode ser de qualquer tamanho e sua complexidade pode ser variável.

Um sistema gerenciador de banco de dados (SGBD) é uma coleção de programas que permite ao usuário criar, acessar e modificar esses programas. Ele facilita a definição, construção, manipulação e compartilhamento do BD entre aplicações e usuários diversos. Algumas funções importantes de um SGBD é a proteção do sistema com relação à falhas, funcionamento e tipos de acessos não autorizados e também a manutenção do BD (ELMASRI; NAVATHE, 2005).

Segundo Korth e Silberschatz (1999), um SGBD é uma coleção de dados inter-relacionados e um conjunto de programas para acessar os dados. Seu principal objetivo é ter um ambiente que seja adequado e eficiente para recuperar e armazenar as informações do BD.

Em 1970 Edgar Frank Codd, da IBM, criou o modelo relacional e publicou um artigo sobre o modelo relacional baseado em álgebra relacional e cálculos matemáticos, mas somente na década de 80 esse modelo foi disponibilizado, era o Oracle e o SQL/DS (Structured Query Language/Data System) da IBM (International Business Machines). Hoje em dia existem outros tipos de modelo relacional da Oracle, IBM e Microsoft. O SQL/DS atualmente é o DB2 da IBM, o Oracle, o Access da Microsoft, entre outros (ELMASRI; NAVATHE, 2005).

O modelo relacional representa o Banco de Dados Relacional (BDR) como um conjunto de esquemas de relações, baseado na teoria dos conjuntos, os dados são armazenados em tabelas (Tabela 2), elas podem ser relacionadas e manipuladas. Cada linha dessa tabela formada por uma ordem de colunas é denominada tupla, a tabela é denominada relação, em que relação é semelhante a uma tabela de valores, as colunas são os atributos e domínio são os valores possíveis de cada atributo (Figura 3) (MARQUES, 2007).

No modelo relacional existem quatro termos que são utilizados, domínio, tupla, atributo e relação. (ELMASRI; NAVATHE, 2005).

Domínio é um conjunto de valores atômicos, indivisível, um tipo de dado, um nome e um formato é especificado para cada domínio (ELMASRI; NAVATHE, 2005).

Em uma relação R (A1, A2, A3, .., An), o elemento A_i refere-se aos atributos de uma relação, que é o nome do papel desempenhado por um domínio D (ELMASRI; NAVATHE, 2005).

Uma relação R (A1, A2, A3, .., An), indicada por r(R) é um conjunto de várias tuplas r{t1,t2,...tn}(ELMASRI; NAVATHE, 2005).

Cada n-tuplas é uma lista ordenada de n valores, cada valor é um elemento do dom (A) ou um valor null (ELMASRI; NAVATHE, 2005).

R.A.	Nome	Telefone
233652	Anne Caroline	Null
235689	Gabriel Henrique	5482-5685
256694	Letícia Cristina	3265-6589
245586	Pedro Luis	Null

Tabela 2- Tabela Aluno.

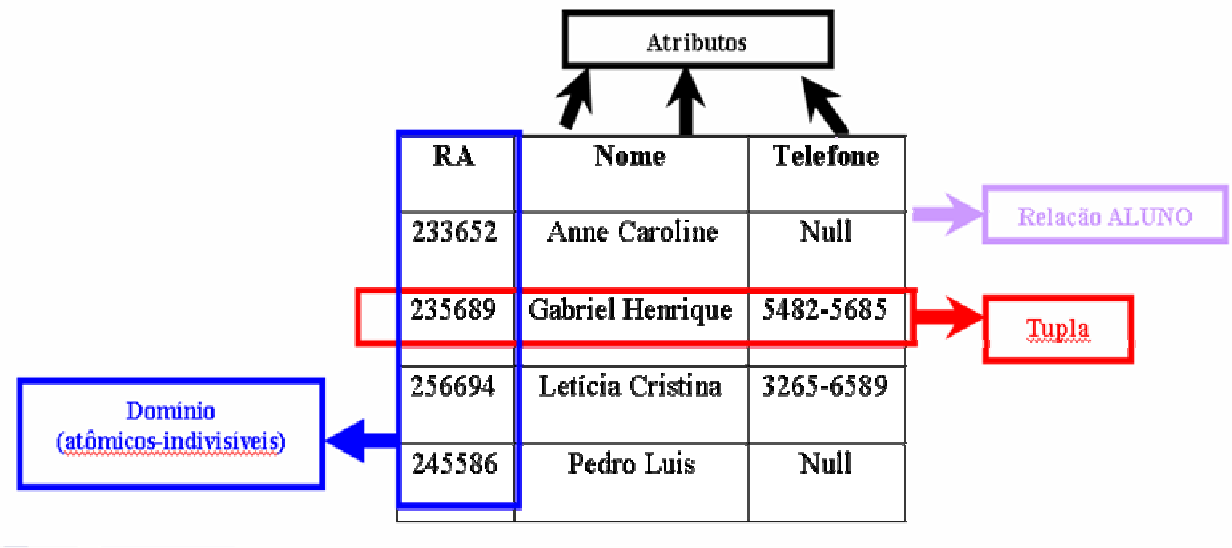


Figura 3- Relação ALUNO com indicações de atributos, tuplas e domínios.

2.1 Restrições de integridades

O BDR não possui somente uma relação, mas sim várias relações e para esse conjunto de relações, existem algumas restrições de integridades e suas características são:

Restrição de chaves: duas regras fundamentais para garantir a sua integridade.

A primeira se refere ao fato de duas ou mais tuplas diferentes não permitir ter o mesmo valor para cada atributo da chave primária, denominando-se *restrição de unicidade* (ELMASRI e NAVATHE, 2005). Para isso existem alguns tipos de restrições de chaves:

É chamado de SuperChave um conjunto de atributos que garantam não ter duplicidade nas tuplas de uma relação, ou seja, tem unicidade a relação e é possível através dela saber em a qual relação essa SuperChave pertence no modelo relacional (ELMASRI e NAVATHE, 2005). Alguns exemplos de SuperChave na tabela Aluno é (RA) e (R.A e Nome), pois dois alunos não podem ter o mesmo nome e o mesmo número de R.A. (Tabela 2).

Chaves candidatas é outro tipo de restrição usada, sua definição é não possuir SuperChaves dentro do mesmo conjunto (ELMASRI e NAVATHE, 2005). Por exemplo, se (RA e Nome) e (RA) são SuperChaves, então (RA e Nome) não pode ser chave candidata, pois RA já esta dentro de (RA e Nome), sendo assim RA é Super chave e também chave candidata.

A segunda se refere ao fato de uma chave primária não poder conter valores nulos (null), sendo o null uma não existência de um conteúdo e não quer dizer que seja um valor ‘0’

ou um caractere branco. Denomina-se essa restrição como *integridade de entidade* (ELMASRI e NAVATHE, 2005).

Restrição de Domínios: para cada coluna de uma relação existe um determinado domínio, em que está associado ao tipo do atributo. Por exemplo, se o tipo de dado de um atributo for inteiro, então o domínio daquela coluna tem que ser inteiro.

Segundo Elmasri e Navathe (2005), restrição de domínio é dentro de cada tupla, o valor de cada atributo A deve ser um valor atômico, ou seja, um valor indivisível do domínio Dom(A).

Restrição de valores nulos: Abreu e Machado (2000) define valor nulo um atributo que não é obrigatório ter um valor, então quando não é informado nenhum valor é dito ser um valor nulo. Na restrição de valores nulos não é permitido que em uma tupla inteira existam valores nulos, é obrigatório ter um atributo que seja não nulo (*Not Null*).

Restrição de integridade de entidade: ter um valor nulo em uma chave primária não é permitido, pois a chave primária é o que indica a tupla em uma relação e se precisar fazer uma referência a essa relação é muito difícil saber a qual relação determinada tupla pertence (ABREU e MACHADO, 2000).

Restrição de integridade referencial: os domínios da chave estrangeira de uma relação devem corresponder-se com uma chave primária, independente que seja da mesma relação ou não. A chave estrangeira faz referência à chave primária da tabela relacionada (Figura 4). Para se manter consistência nessa relação é preciso que uma tupla de uma tabela faça referência a uma tupla de outra tabela (ELMASRI e NAVATHE, 2005).

Existindo dois conjuntos de atributos X e Y que seus domínios sejam compatíveis, a chave estrangeira é um conjunto de atributos que não é chave primária em uma relação, mas é compatível com a chave primária de outra relação (MARQUES, 2007).

Segundo Abreu e Machado (2000), na integridade referencial, se uma tabela A possui uma chave estrangeira em que essa é considerada chave primária na tabela B, então essa chave deve o mesmo valor da chave primária da tabela B, pois não pode existir um valor que não exista na tabela da chave primária ou ser nula (null).

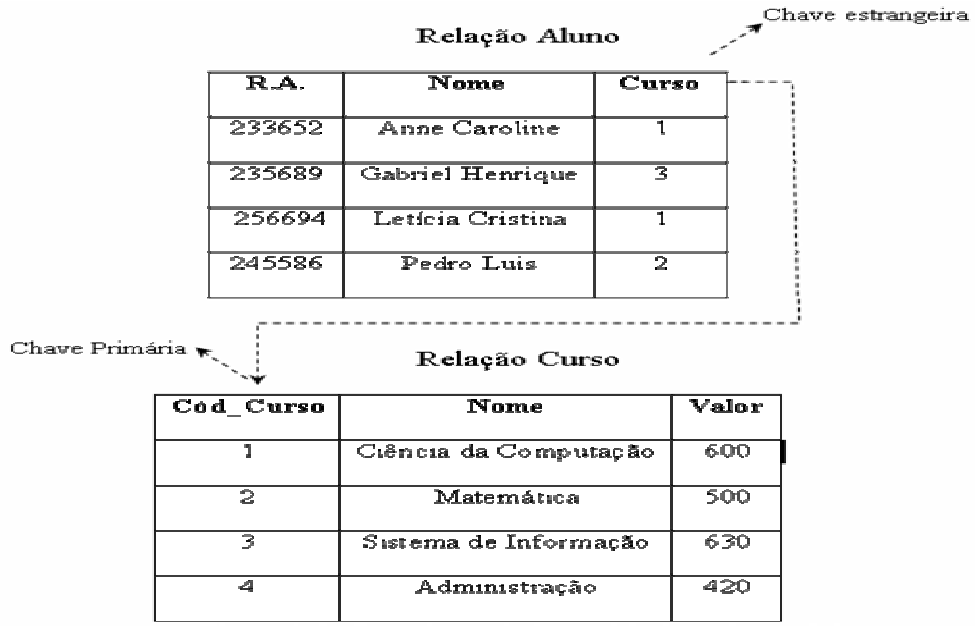
A restrição de integridade referencial pode ser violada se uma tupla for inserida ou excluída, ou os atributos forem alterados (ELMASRI e NAVATHE, 2005).

Restrição de integridade semântica: Elmasri e Navathe (2005) falam sobre integridade semântica dando o seguinte exemplo:

> "[..] os exemplos de restrições semânticas são 'O salário de um empregado não deveria exceder o do supervisor do empregado' e 'o número de horas que um empregado pode trabalhar por semana, é 56'. Essas restrições podem ser especificadas e impostas dentro dos programas de aplicação que atualizam o banco de dados ou pelo uso de uma linguagem de especificação de restrição de propósito geral. Os mecanismos conhecidos como gatilhos e asserções podem ser usados".

Conforme afirmam (SOUZA, 2008 apud MELLO, 2003), as restrições de integridade semântica abrange os valores permitidos e transições de valores válidos e não somente restringe os tipos de dados de um atributo, garantindo assim a integridade do banco de dados.

Restrição de dependência funcional: um atributo determinará sempre o valor do outro, sendo uma restrição entre dois conjuntos de atributos de uma relação, onde um atributo A é dependente funcional de um outro atributo B, se B aparecer nas linhas da entidade em que aparece um único valor de A (ABREU e MACHADO, 2000; ELMASRI e NAVATHE, 2005). Por exemplo, na Figura 4, o atributo Cód_Curso 1 , sempre vai determinar o nome do curso Ciência da Computação.



R.A.	Nome	Curso
233652	Anne Caroline	1
235689	Gabriel Henrique	3
256694	Letícia Cristina	1
245586	Pedro Luis	2

Cód_Curso	Nome	Valor
1	Ciência da Computação	600
2	Matemática	500
3	Sistema de Informação	630
4	Administração	420

Figura 4- Restrição de integridade referencial

2.2 Linguagem SQL

Na Linguagem SQL (*Structured Query Language*) a estrutura do BDR é definida através da linguagem de definição de dados (DDL), alguns dos comandos utilizados é a criação, exclusão e alteração na tabela, sendo permitido então que os dados sejam manipulados e para manipular esses dados existe uma linguagem de manipulação de dados (DML) (ABREU E MACHADO, 2000). Segundo Abreu e Machado (2000), o conceito de DML "permite ao usuário ou a um programa de aplicação, a inclusão, remoção, seleção ou atualização de dados previamente armazenados no banco". E suas operações são *Select, Insert, Update* e *Delete*. Para a geração de testes no banco é utilizada a DML, as operações de *Insert, Delete* e *Update* podem alterar o estado do banco. Esses comandos de DML podem ser aplicados em um banco de dados, como testes para verificação de restrições de integridades e para qualquer outro tipo de teste que se deseja inserir em um BDR (ABREU e MACHADO, 2000).

O *Insert* é um comando que quando executado inclui novas tuplas à tabela, dependendo do valor inserido ele pode quebrar algumas restrições impostas pelo modelo relacional. Por exemplo, se for inserido um valor de atributo nulo, sendo o atributo chave primária, isso violará a restrição de integridade de entidade, pois em uma chave primária não são permitidos valores nulos (ABREU e MACHADO, 2000). Um exemplo do comando *Insert*, em que serão inseridos os seguintes campos na tabela Aluno:

```
INSERT INTO Aluno
        VALUES ('256256', 'Maria Roberta', null);
```

O *Delete* remove apenas a tupla, mas pode violar apenas a restrição de integridade referencial. Uma tabela que é referenciada por outra não pode ter remoção de tuplas, pois a tabela que referenciou contém tuplas onde terá atributos que dependem da tabela referenciada (ABREU e MACHADO, 2000). Por exemplo, na Figura 4 a Tabela Aluno está referenciando a Tabela Curso, se for removido a primeira tupla da tabela Curso que é o curso de Ciência da Computação, viola-se a restrição, pois a tabela aluno contém tuplas que referenciam ao Curso de Ciência da Computação. Exemplo do comando *Delete*, em que estará removendo a tupla que corresponde ao RA informado:

```
DELETE FROM Aluno
        WHERE RA= '233652';
```

O *Update* atualiza alguns atributos de uma tupla, em que são selecionados por condições impostas no comando, a utilização do *update* também possui restrições quanto a não modificar atributos que são chaves primárias e estrangeiras (ABREU e MACHADO, 2000). Segundo Elmasri e Navathe (2005), “modificar um atributo de chave primária é similar a remover uma tupla e inserir outra em seu lugar, porque usamos a chave primária para identificar as tuplas”. Um exemplo de UPDATE, que multiplica por dois o valor da mensalidade somente do curso de Ciência da computação:

```
UPDATE Curso
    SET Valor=Valor*2
        Where Nome = ‘Ciência da Computação’;
```

Em alguns tipos de SGBD, por exemplo, em Banco de Dados transacionais para validar o que foi modificado no banco até o momento, pode ser utilizado o comando COMMIT, este comando efetiva uma transação em execução e a torna visível é um comando utilizado para controle de transações, tornando o dado armazenado persistente. Diferente do comando ROLLBACK que cancela as transações feitas, desfaz o trabalho (ELMASRI e NAVATHE, 2005).

CAPÍTULO 3 – Teste de Banco de Dados em ABDR.

Gonçalves (2003), diz que uma ABDR é formada por módulos de programas desenvolvidos em uma determinada linguagem de programação, essa linguagem é associada a um SGBD que utiliza o modelo relacional e esse SGBD usa o SQL para manipulação de seus dados.

Spoto (2000) descreve que as ABDR são formadas por um ou vários programas que utilizam diferentes linguagens de programação procedimentais, dependendo do SGBD adotado, e permite a utilização de comandos da linguagem SQL em seu código. A figura 5 exemplifica uma Aplicação de Banco de Dados Relacional.

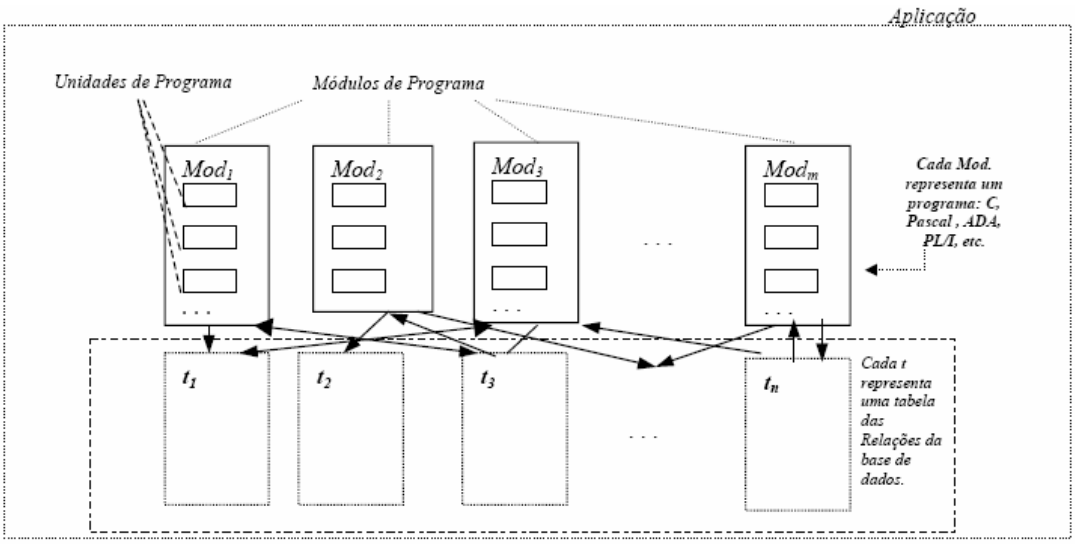


Figura 5 – Exemplo de uma Aplicação de Banco de Dados Relacional (SPOTO, 2000).

Spoto (2000) aborda que existem poucas publicações sobre teste em ABDR.

É utilizado critérios de teste para exercitar em uma ABDR.

Segundo Souza (2008), os critérios de testes tradicionais que são o particionamento por classes de equivalência, e a análise do valor limite, tratam unicamente das variáveis de domínio especificadas no sistema, sem levar em consideração as especificações do Banco de Dados.

Os critérios abordados nas seções 3.1 e 3.2 consideram as exigências do banco de dados, exercita o domínio de um ABDR e trata as dependências funcionais de uma ou mais tabelas utilizando a técnica de teste funcional *Intra-tabelas* (interações em uma mesma tabela) e a técnica de teste funcional *Inter-tabelas* (interações entre tabelas distintas).

3.1 Teste Funcional Intra-tabelas.

O Teste Funcional *Intra-tabelas* é realizado dentro da mesma tabela (função ou procedimento), ou seja, é quando uma tabela não tem vínculo com outra, como por exemplo, tabelas que são referenciadas (SOUZA, 2008). Como exemplo, os testes de domínio, integridade semântica de atributos são feitos em uma única tabela.

3.2 Teste Funcional Inter-tabelas.

O Teste Funcional *Inter-tabelas* é realizado entre duas ou mais tabelas distintas (funções ou procedimentos), ou seja, é preciso utilizar esse método, pois uma tabela referencia outra e assim um atributo pertencente a uma tabela como chave primaria ele é referenciado na outra tabela como chave estrangeira (SOUZA, 2008).

3.3 Critérios para Teste Funcional

Os Critérios de testes funcional apresentados na seção 1.2.2, abordam o critério por classe de equivalência e Análise do valor limite, esse critérios são utilizados na especificação de um sistema sem ter a necessidade de se testar um Banco de Dados. Os critérios propostos nas seções abaixo equivalem à necessidade de testar um Banco de Dados. São propostos critérios que tem por objetivo testar as dependências funcionais e restrições de integridades, de uma mesma tabela ou de tabelas racionadas, utilizando as técnicas de intra-tabelas e inter-tabelas.

3.3.1 Critérios para teste Intra-tabelas

Restrição de Chaves: A restrição de chaves pode ser dividida em dois tipos, Restrição de Unicidade e Restrição de Integridade de Entidade (ELMASRI e NAVATHE, 2005):

- Para satisfazer a Restrição de Unicidade, duas ou mais tuplas não podem conter o mesmo valor para cada atributo de chave primária.

- Para satisfazer a Restrição de Integridade de Entidade, as tuplas não podem conter valores nulos para cada atributo de chave primária, considerando que valor nulo não significa valor ‘0’ nem caractere branco.

Restrição de Domínio de atributo:

- Para satisfazer a restrição de domínio de atributo, todos os valores do domínio dos atributos têm que respeitar o seu tipo de dado, ou seja, cada coluna de uma tupla tem que possuir o mesmo tipo de dado. O valor de cada atributo deve ser atômico, indivisível (ELMASRI e NAVATHE, 2005).

3.3.2 Critério para teste Inter-tabelas

Restrição de Integridade Referencial:

- Para satisfazer a Restrição de Integridade Referencial, uma chave primaria (PK) de uma tupla na tabela B é chave estrangeira (FK) na tabela A, então essas chaves devem ter o mesmo valor e domínio (ELMASRI e NAVATHE, 2005; ABREU e MACHADO, 2000).

CAPÍTULO 4 – Estudo de Caso

O BD utilizado foi desenvolvido pela empresa MOVSOFT - Tecnologia Móvel que iniciou suas atividades na incubadora mantida pelo Centro Incubador de Empresas de Marília (CIEM) para uma loja de suplementos alimentares. O BD tem a finalidade de fazer o controle dos produtos venda e estoque sendo desenvolvido em SGBD MySQL 5.0.27 e utilizado uma ferramenta para suporte na criação do BD de uma maneira visual, esse BD possui seis tabelas que são: *descartados, lote, produto, produto_lote, venda e venda_produto*. A figura 6 mostra o Banco de Dados utilizado.

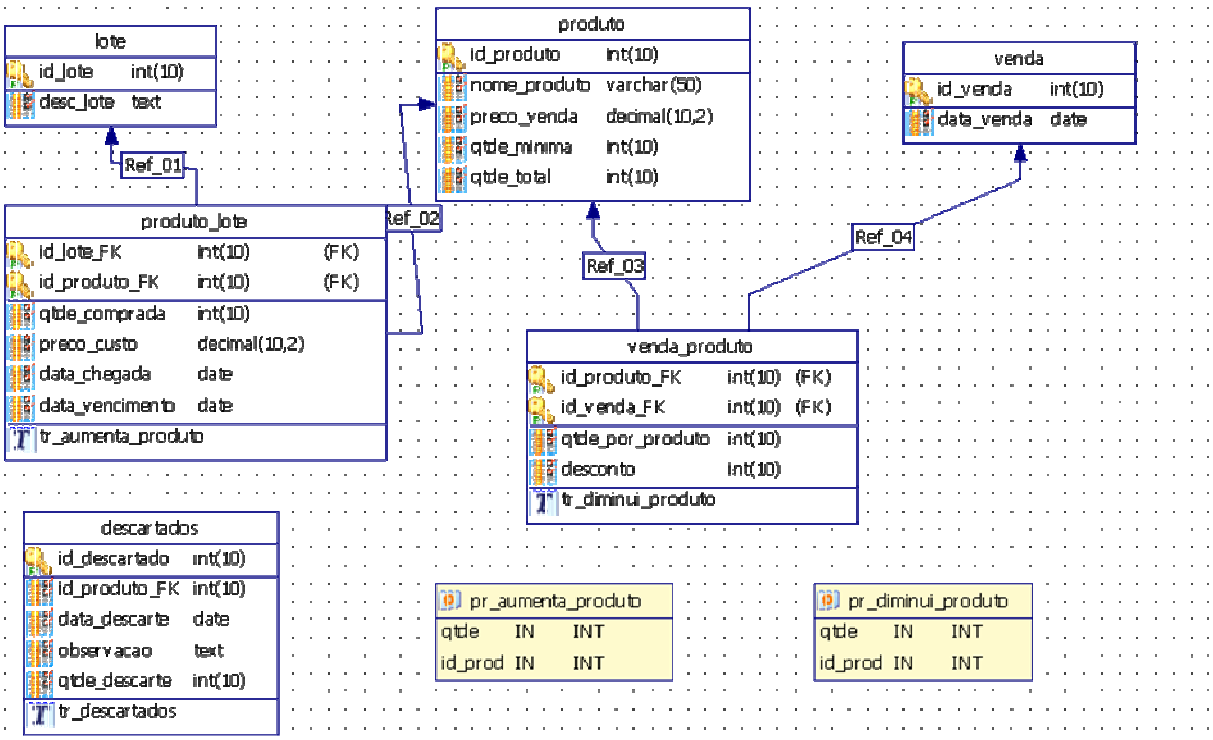


Figura 6 - Estrutura do Banco de Dados da empresa MOVSOFT - Tecnologia Móvel

A tabela lote tem um relacionamento muitos-para-muitos com a tabela produto, gerando com isso a tabela produto_lote.

A tabela venda tem um relacionamento muitos-para-muitos com a tabela produto, gerando com isso a tabela venda_produto.

A tabela descartados não tem relacionamento com outras tabelas.

Na tabela *descartados* um produto pode ser descartado e seus atributos descrevem o motivo e outros dados pelo qual um produto foi descartado. Nesta mesma tabela existe um

trigger (*tr_descartados*) que diminui automaticamente a quantidade de produtos na tabela *produto.*

A tabela *lote* é um cadastro do lote do produto, conforme é fabricado.

A tabela *produto* é a descrição das características do produto.

Na tabela *produto_lote*, o lote muda conforme a data que é produzido, mas nem sempre o mesmo produto tem o mesmo lote. Nesta tabela existe um trigger (*tr_aumenta_produto*), que aumenta automaticamente a quantidade de produto na tabela *produto*, essa tabela é como se fosse uma tabela compra_produto.

A tabela *venda* indica que uma venda está sendo efetuada e a referida data.

A tabela *venda_produto* indica que está sendo efetuada uma venda e indica qual produto está incluso nesta venda. Nesta tabela também existe um trigger (*tr_diminui_produto*) que diminui automaticamente a cada venda a quantidade de produto da tabela *produto.*

O BD ainda possui as *stores procedures (pr_diminui_produto e pr_aumenta_produto)* que são chamados pelos triggers para fazer as alterações nas tabelas.

Trigger (gatilho) é um comando executado pelo sistema utilizado automaticamente por uma modificação do banco de dados, ele mantém a consistência dos dados. Para criar o trigger é necessário especificar em quais condições um trigger deve ser acionado e qual ação tem que ser tomadas após a execução (KORTH E SILBERSCHATZ, 1999).

4.1 Elementos requeridos.

Segundo (Souza, 2008 apud Vilela, 2007), os Elementos Requeridos (ERs), são requisitos que satisfazem o teste, e eles são identificados através dos critérios de testes.

Os elementos requeridos mostrados nas seções a seguir são baseados nos critérios propostos nas seções 3.3.1 e 3.3.2 e utilizando as técnicas de teste funcional, por partição por classes de equivalência e análise do valor limite.

4.1.1 Intra-tabelas

Restrição de Chave Primária: Restrição de unicidade.

Tabelas produto, lote, venda e descartados.			
Atributo	**Classes válidas**	**Classe inválidas**	**Saída esperada**
id_lote	Duas tuplas não podem ter o mesmo valor de chave primária do atributo	Duas tuplas possuem o mesmo valor de chave	A chave primária da tabela *lote* é única, então o BD

	id_lote.	primária.	respeitou a restrição de unicidade.
id_produto	Duas tuplas não podem ter o mesmo valor de chave primária do atributo *id_produto*.	Duas tuplas possuem o mesmo valor de chave primária.	A chave primária da tabela *produto* é única, então o BD a respeitou a restrição de unicidade.
id_venda	Duas tuplas não podem ter o mesmo valor de chave primária do atributo *id_venda*.	Duas tuplas possuem o mesmo valor de chave primária.	A chave primária da tabela *venda* é única, então o BD respeitou a restrição de unicidade.
id_descartado	Duas tuplas não podem ter o mesmo valor de chave primária do atributo *id_descartado*.	Duas tuplas possuem o mesmo valor de chave primária.	A chave primária da tabela *descartado* é única, então o BD respeitou a restrição de unicidade.

Tabela 3 – Elementos requeridos para Restrição de Unicidade

Restrição de Chave Primária: Restrição de Entidade.

Tabela lote, produto, venda e descartados.			
Atributo	**Classes válidas**	**Classe inválidas**	**Saída esperada**
id_lote	Não pode conter valores nulos na chave primária do atributo *id_lote*	A chave primária *id_lote* contém valor nulo.	A chave primária da tabela *lote* não possui valor nulo, então o BD respeitou a restrição de entidade.
id_produto	Não pode conter valores nulos na chave primária do atributo *id_produto*	A chave primária *id_produto* contém valor nulo.	A chave primária da tabela *produto* não possui valor nulo, então o BD respeitou a restrição de entidade.
id_venda	Não pode conter valores nulos na chave primária do atributo *id_venda*	A chave primária *id_venda* contém valor nulo.	A chave primária da tabela *venda* não possui valor nulo, então o BD respeitou a restrição de entidade.
id_descartado	Não pode conter valores nulos na chave primária do atributo *id_descartado*	A chave primária *id_descartado* contém valor nulo.	A chave primária da tabela *descartado* não possui valor nulo, então o BD respeitou a restrição de entidade.

Tabela 4– Elementos requeridos para Restrição de Entidade

Restrição de Domínio de atributo

Tabela lote					
Atributo	**Classes válidas**	**Classe inválidas**	**Classe inválida Limite Superior**	**Classe inválida Limite Inferior**	**Saída esperada**
id_lote	1≤int ≤10, not null	int<1 ou int >10, null	int=11	int=0	Código do lote com domínio correto.
desc_lote	Domínio text Máximo caracteres = 65535	Caracteres acima do valor válido	-	-	Descrição do lote com domínio correto.
Tabela produto_lote					
Atributo	**Classes válidas**	**Classe inválidas**	**Classe inválida Limite Superior**	**Classe inválida Limite Inferior**	**Saída esperada**
id_lote_FK	1≤int ≤10, not null	int<1 ou int >10, null	int=11	int=0	Código do lote com domínio correto.

id_produto_FK	1≤int ≤10, not null	int<1 ou int >10, null	int=11	int=0	Código do produto com domínio correto.
qtde_comprada	1≤int ≤10, not null	int<1 ou int >10, null	int=11	int=0	Atributo com domínio correto.
preço_custo	Decimal formato ‘0.00’,not null	Formato diferente de ‘0.00’ , null	-	-	Preço de custo com domínio correto.
data_chegada	date no formato ’aaaa-mm-dd’	Fora do formato ‘aaaa-mm-dd’	-	-	Data de chegada com domínio correto.
data_vencimento	date no formato ’aaaa-mm-dd’	Fora do formato ‘aaaa-mm-dd’	-	-	Data do vencimento com domínio correto.
Tabela produto					
Atributo	**Classes válidas**	**Classe inválidas**	**Classe inválida Limite Superior**	**Classe inválida Limite Inferior**	**Saída esperada**
id_produto	1≤int ≤10, not null	int<1 ou int >10, null	int=11	int=0	Código do produto com domínio correto.
nome_produto	1≤varchar≤20, not null	varchar<1 ou varchar>20, null	varchar=21	varchar=0	Atributo com domínio correto.
preço_venda	Decimal formato ‘0.00’,not null	null	-	-	Preço de venda com domínio correto.
qtde_minima	1≤int ≤10, not null	int<1 ou int >10, null	int=11	int=0	Atributo com domínio correto.
qtde_total	1≤int ≤10	int<1 ou int >10	int=11	int=0	Atributo com domínio correto.
Tabela venda					
Atributo	**Classes válidas**	**Classe inválidas**	**Classe inválida Limite Superior**	**Classe inválida Limite Inferior**	**Saída esperada**
id_venda	1≤int ≤10, not null	int<1 ou int >10, null	int=11	int=0	Código da venda com domínio correto.
data_venda	date no formato ’aaaa-mm-dd’	Formato diferente de ‘aaaa-mm-dd’	-	-	Data da venda com domínio correto.
Tabela venda_produto					
Atributo	**Classes válidas**	**Classe inválidas**	**Classe inválida Limite Superior**	**Classe inválida Limite Inferior**	**Saída esperada**
id_produto_FK	1≤int ≤10, not null	int<1 ou int >10, null	int=11	int=0	Código do produto com domínio correto.
id_venda_FK	1≤int ≤10, not null	int<1 ou int >10, null	int=11	int=0	Código da venda com domínio correto.
qtde_por_produto	1≤int ≤10, not null	int<1 ou int >10, null	int=11	int=0	Atributo com domínio correto.
desconto	1≤int ≤10	int<1 ou int >10	int=11	int=0	Atributo com domínio correto.
Tabela descartados					
Atributo	**Classes válidas**	**Classe inválidas**	**Classe inválida Limite Superior**	**Classe inválida Limite Inferior**	**Saída esperada**
id_descartado	1≤int ≤10,	int<1 ou int	int=11	int=0	Código do descartado

	not null	>10, null			com domínio correto.
id_produto_FK	1≤int ≤10, not null	int<1 ou int >10, null	int=11	int=0	Código do produto com domínio correto.
observação	Domínio text Máximo caracteres = 65535	Caracteres acima do valor válido	-	-	Observação de descartados com domínio correto.
qtde_descarte	1≤int ≤10, not null	int<1 ou int >10, null	int=11	int=0	Atributo com domínio correto.

Tabela 5– Elementos requeridos para Restrição de Domínio de atributo

4.1.2 Inter-tabelas

Restrição de Integridade Referencial

Tabelas: relacionamento de venda e venda_produto			
Atributo	**Classes válidas**	**Classe inválidas**	**Saída esperada**
id_venda e id_venda_FK	A chave *id_venda(PK)* da tabela *venda* e a *id_venda_FK* da tabela *venda_produto* devem ter o mesmo valor e domínio	A chave *id_venda da tabela venda*, não tem o mesmo valor e domínio da chave *id_venda_FK* da tabela *venda_produto*	A Chave primaria *id_venda* e a chave estrangeira *id_venda_FK*, possuem o mesmo valor e domínio
id_venda	Excluir *id_venda* da tabela *venda*, e dar erro.	Excluir *id_venda* da tabela *venda* e o BD aceitar.	Excluindo *id_venda* da tabela *venda* o BD não aceita, pois existe uma *id_venda_fk* dependente na tabela *venda_produto.*
Tabelas: relacionamento de produto e venda_produto			
Atributo	**Classes válidas**	**Classe inválidas**	**Saída esperada**
id_produto e id_produto_FK	A chave *id_produto(PK)* da tabela *produto* e a *id_produto_FK* da tabela *venda_produto* devem ter o mesmo valor e domínio	A chave *id_produto* da tabela *produto*, não tem o mesmo valor e domínio da chave *id_produto_FK* da tabela *venda_produto*	A Chave primaria *id_produto* e a chave estrangeira *id_produto_FK*, possuem o mesmo valor e domínio
id_produto	Excluir *id_produto* da tabela *produto*, e dar erro.	Excluir *id_produto* da tabela *produto* e o BD aceitar.	Excluindo *id_produto* da tabela *produto* o BD não aceita, pois existe uma *id_produto_fk* dependente na tabela *venda_produto.*
Tabelas: relacionamento de lote e produto_lote			
Atributo	**Classes válidas**	**Classe inválidas**	**Saída esperada**
id_lote e id_lote_FK	A chave *id_lote(PK)* da tabela *lote* e a id_lote_FK da tabela *produto_lote* devem ter o mesmo valor e domínio	A chave *id_lote da tabela lote*, não tem o mesmo valor e domínio da chave *id_lote_FK* da tabela *produto_lote*	A Chave primaria *id_lote* e a chave estrangeira *id_lote_FK*, possuem o mesmo valor e domínio
id_lote	Excluir *id_lote* da tabela *lote*, e dar erro.	Excluir *id_lote* da tabela *lote* e o BD aceitar.	Excluindo *id_lote* da tabela *lote* o BD não aceita, pois existe uma *id_lote_fk* dependente na tabela *produto_lote.*
Tabelas: relacionamento de produto e produto_lote			
Atributo	**Classes válidas**	**Classe inválidas**	**Saída esperada**

id_produto e id_produto_FK	A chave *id_produto(PK)* da tabela *produto* e a id_produto_FK da tabela *produto_lote* devem ter o mesmo valor e domínio	A chave *id_produto da tabela produto*, não tem o mesmo valor e domínio da chave *id_produto_FK* da tabela *produto_lote*	A Chave primaria *id_produto* e a chave estrangeira *id_produto_FK*, possuem o mesmo valor e domínio
id_produto	Excluir *id_produto* da tabela *produto*, e dar erro.	Excluir *id_produto* da tabela *produto* e o BD aceitar.	Excluindo *id_produto* da tabela *produto* o BD não aceita, pois existe uma *id_produto_fk* dependente na tabela *produto_lote.*
Tabelas: relacionamento de produto e descartados			
Atributo	**Classes válidas**	**Classe inválidas**	**Saída esperada**
id_produto	Excluir *id_produto* da tabela *produto*, e dar erro.	Excluir *id_produto* da tabela *produto* e o BD aceitar.	Excluindo *id_produto* da tabela *produto* o BD não aceita, pois existe uma *id_produto_fk* dependente na tabela *descartados.*

Tabela 6 – Elementos requeridos para Restrição de Integridade Referencial

CAPÍTULO 5 - Casos de testes

Os Casos de testes são realizados no banco de dados através de comandos DML no SGBD MYSQL. São apresentadas as entradas de dados, o teste aplicado, a saída esperada, a mensagem que o banco de dados mostra e a descrição da saída realizada pelo banco de dados.

5.1 Casos de testes Intra - tabelas

Nesta seção são apresentados os casos de testes Intra - tabelas.

Restrição de Chave Primária

Tabelas: produto, lote, venda e descartados				
Entrada de Dados	**Teste aplicado**	**Saída esperada**	**Mensagem do BD**	**Saída do BD**
Inserir Chave Primária existente na tabela p*roduto*	INSERT INTO produto VALUES('001', 'Balas', 53.00, 16, 0);	Entrada duplicada para chave primária de valor ‘001’	ERROR 1062: Duplicate entry '1' for key 1	(OK) Restrição de Unicidade não violada.
Inserir Chave Primária existente na tabela *lote*	INSERT INTO lote VALUES(6, 'R5');	Entrada duplicada para chave primária de valor ‘6’	ERROR 1062: Duplicate entry '6' for key 1	(OK) Restrição de Unicidade não violada.
Inserir Chave Primária existente na tabela *venda*	INSERT INTO venda VALUES(4, '2008-01-09');	Entrada duplicada para chave primária de valor ‘4’	ERROR 1062: Duplicate entry '4' for key 1	(OK) Restrição de Unicidade não violada.
Inserir Chave Primária existente na tabela *descartados*	INSERT INTO Descartados VALUES(1,'006', curdate(), 'produto vencido', 2);	Entrada duplicada para chave primária de valor ‘1’	ERROR 1062: Duplicate entry '1' for key 1	(OK) Restrição de Unicidade não violada.

Tabela 7-Caso de teste Restrição de Chave Primária - Restrição de Unicidade

Tabelas: produto, lote, venda e descartados				
Entrada de Dados	**Teste aplicado**	**Saída esperada**	**Mensagem do BD**	**Saída do BD**
Verificar Chave Primária existente na tabela *produto*	INSERT INTO produto VALUES(null, 'Sabão', 5.00, 18, 0);	Não aceitar valores nulos.	Query OK, 1 row affected.	(OK) A chave primária adicionou um código na seqüência, pois possui o auto-increment
Verificar Chave Primária existente na tabela *lote*	INSERT INTO lote VALUES (null, 'R2');	Não aceitar valores nulos.	Query OK, 1 row affected.	(OK) A chave primária adicionou um código na seqüência, pois possui o auto-increment
Verificar Chave Primária existente na tabela *venda*	INSERT INTO venda VALUES (null, '2009-01-25');	Não aceitar valores nulos.	Query OK, 1 row affected.	(OK) A chave primária adicionou um código na seqüência, pois possui o auto-

				increment
Verificar Chave Primária existente na tabela *descartados*	INSERT INTO descartados VALUES (null,'002', curdate(), 'produto vencido',1);	Não aceitar valores nulos.	Query OK, 1 row affected.	(OK) A chave primária adicionou um código na seqüência, pois possui o auto-increment

Tabela 8-Restrição de Chave Primária – Restrição de entidade

Restrição de Domínio de atributo.

Tabela lote				
Entrada de Dados	**Teste aplicado**	**Saída esperada**	**Mensagem do BD**	**Saída do BD**
Inserir valor int>10 no atributo id_lote.	INSERT INTO lote VALUES(12345678901, 'R2');	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'id_lote' at row 1	(OK) Fora do valor ajustado para id_lote
Inserir valor int<1 no atributo id_lote.	INSERT INTO lote VALUES(-1, 'R2');	Não é possível inserir int<0, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'id_lote' at row 1	(OK) Fora do valor ajustado para id_lote
Inserir um text para o atributo desc_lote.	INSERT INTO lote VALUES(13, 'Ghggg55');	BD aceita, pois está dentro do limites de caracteres.	Query OK, 1 row affected	(OK) Inserido valor em desc_lote com domínio correto.
Tabela produto_lote				
Inserir valor int>10 no atributo id_lote_fk.	INSERT INTO produto_lote VALUES (12345678901, '001',15,5.50,'2009-01-20',2 009-02-20);	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'id_lote_FK' at row 1	(OK) Fora do valor ajustado para id_lote_FK
Inserir valor int>10 no atributo id_produto_fk.	INSERT INTO produto_lote VALUES(11, '12345678901',15,5.50,'2009-01-20','2 009-02-20');	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'id_produto_FK' at ro w 1	(OK) Fora do valor ajustado para id_produto_FK
Inserir valor int>10 no atributo qtde_comprada.	INSERT INTO produto_lote VALUES(11, '005',12354689758,5.50,'2009-01-20','2 009-02-20');	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'qtde_comprada' at ro w 1	(OK) Fora do valor ajustado para qtde_comprada
Inserir um valor em preço_custo de formato text.	INSERT INTO produto_lote VALUES(12, '004',15,'sem preco','2009-01-10','2 09-01-20');	Não é possível inserir um valor não decimal	ERROR 1366 (HY000): Incorrect decimal value: 'sem preco' for column 'preco_cust ' at row 1	(OK) Valor decimal incorreto

Inserir data no atributo data_chegada em formato diferente.	INSERT INTO produto_lote VALUES(11, '005',15,5.50,'200129-01-20','2009-02-20');	Não é possível inserir a data está em formato incorreto	ERROR 1292 (22007): Incorrect date value: '20092-01-20' for column 'data_chegada' at row 1	(OK) Valor da data está incorreto.
Inserir data no atributo data_vencimento em formato diferente.	INSERT INTO produto_lote VALUES(11, '005',15,5.50,'2009-01-20','215009-02-20');	Não é possível inserir a data está em formato incorreto	ERROR 1292 (22007): Incorrect date value: '20092-01-20' for column 'data_vencimento' at row 1	(OK) Valor da data está incorreto.
Tabela produto				
Inserir valor int>10 no atributo id_produto.	INSERT INTO produto VALUES('12345678901', 'café',5.50,10,0);	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'id_produto' at row 1	(OK) Fora do valor ajustado para id_produto
Inserir um valor nulo para nome_produto	INSERT INTO produto VALUES ('13',null,15.50,10,0);	Não aceita valor nulo para nome_produto	ERROR 1048 (23000): Column 'nome_produto' cannot be null	(OK) Valor não pode ser nulo
Inserir um valor nulo para preço_venda	INSERT INTO produto VALUES ('12','macarrao',null,10,0);	Não aceita valores nulos para preco_venda.	ERROR 1048 (23000): Column 'preco_venda' cannot be null	(OK) Valor não pode ser nulo
Inserir valor int>10 no atributo qtde_minima.	INSERT INTO produto VALUES('10', 'arroz',5.50,11234578963,0);	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'qtde_minima' at ro w 1	(OK) Fora do valor ajustado para qtde_minima
Inserir valor int>10 no atributo qtde_total	INSERT INTO produto VALUES('11', 'feijão',4.50,10,12345785478);	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'qtde_total' at ro w 1	(OK) Fora do valor ajustado para qtde_total
Tabela venda				
Inserir valor int>10 no atributo id_venda.	INSERT INTO venda VALUES(12345678901, '2009-01-20');	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'id_venda' at row 1	(OK) Fora do valor ajustado para id_venda

Inserir data no atributo data_venda em formato diferente.	INSERT INTO venda VALUES(13, '20092-01-20');	Não é possível inserir a data está em formato incorreto	ERROR 1292 (22007): Incorrect date value: '20092-01-20' for column 'data_venda' at row 1	(OK) Valor da data está incorreto.
Tabela venda_produto				
Inserir valor int>10 no atributo id_produto_fk.	INSERT INTO venda_produto VALUES(12345678901, '006',1,0);	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'id_produto_FK' at r w 1	(OK) Fora do valor ajustado para id_produto_fk
Inserir valor int>10 no atributo id_venda_fk	INSERT INTO venda_produto VALUES(6, '12534678901',1,0);	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'id_venda_FK' at r w 1	(OK) Fora do valor ajustado para id_venda_fk
Inserir valor int>10 no atributo qtde_por_produto	INSERT INTO venda_produto VALUES(6, '005',12345678901,0);	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'qtde_por_produto' at r w 1	(OK) Fora do valor ajustado para qtde_por_produto
Inserir valor int>10 no atributo desconto	INSERT INTO venda_produto VALUES(6, '003',1,12345678901);	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'desconto' at r w 1	(OK) Fora do valor ajustado para desconto
Tabela descartados				
Inserir valor int>10 no atributo id_descartado	INSERT INTO descartados VALUES(12345678901, '005',curdate(),'produto ven ido',1);	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column 'id_descartado' at r w 1	(OK) Fora do valor ajustado para id_descartado
Inserir valor int>10 no atributo id_produto_fk	INSERT INTO descartados VALUES(6, '12345678910',curdate(),'produto ven ido',2);	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column ' id_produto_fk ' at r w 1	(OK) Fora do valor ajustado para id_produto_fk
Inserir um text para o atributo observacao.	INSERT INTO descartados VALUES(6, '005',curdate(),'produto amassado',2);	BD aceita, pois está dentro do limites de caracteres.	Query OK, 1 row affected	(OK) Inserido valor no atributo observacao com domínio correto.
Inserir valor int>10 no atributo qtde_descarte	INSERT INTO descartados VALUES(6, '004',curdate(),'produto ven ido',12345587852);	Não é possível inserir int>10, pois é classe inválida	ERROR 1264 (22003): Out of range value adjusted for column ' qtde_descarte ' at r w 1	(OK) Fora do valor ajustado para qtde_descarte

Tabela 9 –Caso de teste Restrição de Domínio de atributo.

5.2 Casos de testes Inter – tabelas

Nesta seção são apresentados os casos de testes Inter- tabelas.

Restrição de Integridade Referencial

Entrada de Dados	Teste aplicado	Saída esperada	Mensagem do BD	Saída do BD
Excluir Chave Primária existente na tabela *venda.*	DELETE from venda WHERE id_venda='8';	Não é possível excluir, pois a *PK (id_venda)* está referenciada na tabela *venda_produto* como *FK id_venda_fk*	ERROR 1451: Cannot delete or update a parent row: a foreign key constrai nt fails (`tcc/venda_produto`, CONSTRAINT `Ref_04` FOREIGN KEY (`id_venda_FK`) R EFERENCES `venda` (`id_venda`) ON DELETE NO ACTION ON UPDATE NO ACTION)	(OK) Integridade Referencial não violada
Excluir Chave Primária existente na tabela *produto.*	DELETE from produto WHERE id_produto='003';	Não é possível excluir, pois a *PK (id_produto)* está referenciada na tabela *produto_lote, venda_produto e descratados* como *FK id_produto_fk*	ERROR 1451: Cannot delete or update a parent row: a foreign key constrai nt fails (`tcc/produto_lote`, CONSTRAINT `Ref_02` FOREIGN KEY (`id_produto_FK`) REFERENCES `produto` (`id_produto`) ON DELETE NO ACTION ON UPDATE NO ACTION)	(OK) Integridade Referencial não violada
Excluir Chave Primária existente na tabela *lote.*	DELETE from lote WHERE id_lote=’1’;	Não é possível excluir, pois a *PK (id_lote)* está referenciada na tabela *produto_lote* como *FK id_lote_fk*	ERROR 1451: Cannot delete or update a parent row: a foreign key constrai nt fails (`tcc/produto_lote`, CONSTRAINT `Ref_01` FOREIGN KEY (`id_lote_FK`) REF ERENCES `lote` (`id_lote`) ON DELETE NO ACTION ON UPDATE NO ACTION)	(OK) Integridade Referencial não violada

Tabela 10 - Caso de Teste Restrição de Integridade Referencial

5.3 Análise dos Resultados obtidos.

Nos testes realizados referente ao critério de restrição de chaves dos casos de teste intra-tabelas, na restrição de unicidade, a chave primária de cada tabela foi testada, inserido valores já existentes e não houve violação de restrição de unicidade, pois o Banco de Dados foi projetado corretamente, ou seja, satisfez o critério de restrição de unicidade.

Na restrição de entidade, foi testado o banco inserindo valores nulos nas chaves primárias e como o BD foi projetado com a função auto-increment, que incrementa o próximo valor de chave primaria automaticamente assim que uma tupla for inserida e não existir dado para o atributo de chave primária. O teste foi satisfeito pelo critério, pois com a função auto-increment, o BD automaticamente incrementou uma chave primária, não deixando o atributo nulo e não dando erro no BD.

Os casos de testes realizados, baseando na restrição de domínio de atributo satisfizeram o critério proposto. Os domínios de cada atributo projetados no BD foram testados conforme os elementos requeridos e os testes realizados não apresentaram erros conforme a saída esperada determinada.

Baseando-se na integridade referencial os testes realizados satisfizeram ao critério proposto de inter-tabela. O BD foi projetado de maneira que não violasse a integridade referencial. Nas tabelas que possuem relacionamentos, os testes foram realizados tentando excluir os atributos referenciados, ou seja, tentou-se excluir um atributo da tabela A que é referenciada pela tabela B e nesta tabela esse atributo é chave estrangeira (FK), sendo assim o BD não deixou excluir o determinado atributo, respeitando a integridade referencial. Foram testados nas tabelas os atributos de chave primária que se relaciona com outra tabela, para saber se possuíam o mesmo valor e domínio.

Esse critério também foi satisfeito pelos atributos conterem o mesmo valor e domínio, mesmo estando em tabelas diferentes, mas que possuem referencias. A figura 7 mostra que o BD satisfez todos os critérios e não encontrou erros.

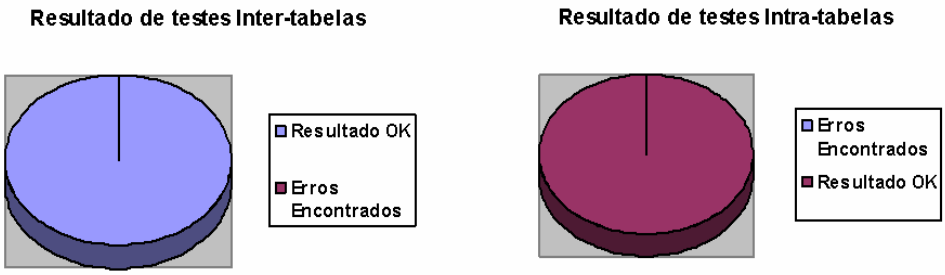


Figura 7 – Gráfico dos Resultados Obtidos com os testes realizados

CONCLUSÃO

Neste trabalho foram apresentados os conceitos relacionados a testes de software, as fases do teste, as técnicas de teste funcional e estrutural, dando ênfase à descrição da técnica de teste funcional e os critérios de classes de equivalência e análise do valor limite, que foi a técnica e os critérios utilizados neste trabalho. Apresentaram-se os conceitos de Banco de Dados, abordando algumas restrições de integridades e linguagem SQL com os comandos de DML. Foi descrito o teste de Banco de Dados em ABDR com as técnicas de intra-tabelas e inter-tabelas.

Os critérios propostos para realização de testes, foram classificados em critérios para intra-tabelas e critério para inter-tabelas.

Houve a apresentação e descrição do estudo de caso referente ao banco de dados proposto e a partir de um diagrama do Banco de Dados, foram apresentados os critérios de teste para a aplicação no Banco de Dados.

A realização dos elementos requeridos partiu do esquema do Banco de Dados com suas funcionalidades e relacionamentos, baseando nos critérios de teste proposto nas seções 3.3.1 e 3.3.2 e nas técnicas de teste funcional. Foram descritos os elementos requeridos de cada tabela e seus atributos.

Os casos de teste foram elaborados à partir dos elementos requeridos e foram realizados os testes no BD com os comandos SQL.

Foram determinadas entradas de dados para determinar os testes, através de comandos DML e inseridos os testes no BD, comparando a saída esperada com a saída que o BD mostrou. Todos os testes executados demonstraram eficiência e os resultados obtidos foram coerentes com a saída esperada pelo BD.

Os testes realizados obtiveram resultados positivos referente ao proposto nos ER’s, não houve ocorrências de erros, referente aos dados e a projeção do BD, pois em se tratando de um Banco de Dados de estrutura pequena, torna-se mais eficiente a projeção do BD com as relações e restrições de integridade que precisa ser desenvolvida.

Uma das maneiras de aperfeiçoamento desse projeto é dar continuidade nos testes no BD, abordando os critérios de restrição de valores nulos, restrição de integridade semântica e restrição de dependência funcional apresentados no Capítulo 2 sobre Banco de Dados.

REFERÊNCIAS BIBLIOGRÁFICAS

ABREU, Maurício Pereira de; MACHADO, Felipe Nery Rodrigues. **Projeto de banco de dados**. 6ª ed. São Paulo: Érica, 2000.

DELAMARO, M. E; MALDONADO, J. C; JINO, M. **Conceitos Básicos**. DELAMARO, M.E; MALDONADO, J, C; JINO, M. **Introdução ao Teste de Software**. Rio de Janeiro: Elsevier,2007.

DOMINGUES, A. L. S. ***Avaliação de critérios e Ferramentas de teste para Programas O.O.,*** Dissertação (Mestrado em Ciência da Computação e Matemática Computacional) Instituto de ciências e Matemática Computacional, ICMC-USP, São Carlos, SP, 2002.

ELMASRI, Ramez; NAVATHE, Shamkant. **Sistemas de banco de dados**. 4ª ed. São Paulo: Addison-Wesley, 2005.

GONÇALVES, Klausner V. "**Teste de Software em Aplicações de Banco de Dados Relacional".** Dissertação de Mestrado- UNICAMP, Campinas-SP- Brasil, Abril 2003.

KORTH, Henry F.; SILBERSCHATZ, Abraham; SUDARSHAN, S. **Sistema de banco de dados**. 3ª ed. São Paulo: Makron Books, 1999.

MALDONADO, J. C. ***Critérios potenciais usos: Uma contribuição ao teste estrutural de software***. Tese de Doutoramento, DCA/FEE/UNICAMP,Campinas,SP,1991.

MALDONADO, José Carlos et al. ***Introdução ao teste de software***. Notas didáticas do ICMC-USP, versão 2004-01. São Carlos, 2004.

MARQUES, Fátima L.S.N et al. ***Banco de Dados / Modelo Relacional*** . Notas didáticas do UNIVEM, Marília, 2007.

MODESTO, Lisandro Rogério. **"Teste Funcional Baseado em Diagramas da UML".** Dissertação de Mestrado em Ciência da Computação – Centro Universitário Eurípides de Marília (UNIVEM), Marília-SP-Brasil, 2006.

MYERS, Glenford J. ***The art of software testing***. 2. ed. Hoboken, New Jersey: John Wiley & Sons, Inc, 2004. 234p.

NARDI, Paulo Augusto. **“Utilização do critério Todos-T-Usos na Ferramenta JABUTI ”**. Dissertação de Mestrado em Ciência da Computação – Centro Universitário Eurípides de Marília (UNIVEM), Marília – SP – Brasil, 2006.

PEDRYCZ, Witold; PETERS, James F. **Engenharia de software: teoria e prática**. Rio de Janeiro: Campus, 2001.

PRESSMAN, Roger S. **Engenharia de software**. São Paulo: Makron Books, 1995.

SOMMERVILLE, Ian. **. Engenharia de software**. 6ª ed. São Paulo: Addison-Wesley, 2004.

SOMMERVILLE, Ian. **. Engenharia de software**. 8ª ed. São Paulo: Pearson, 2008.

SOUZA, Juliana Pereira de.**”Teste Funcional em Aplicação de Banco de Dados Relacional Baseado nos Diagramas da UML”** . Dissertação de Mestrado em Ciência da Computação – Centro Universitário Eurípides de Marília (UNIVEM), Marília- SP- Brasil, 2008.

SPOTO, Edmundo Sérgio**. “Teste estrutural de programas de aplicação de banco de dados relacional”** Tese de Doutorado em Engenharia Elétrica. Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas - FEEC/UNICAMP, Campinas, 2000.

SPOTO, E. S; DELAMARO, M. E. “**Uma Evolução do Teste Estrutural de Software”**: Teoria e Prática. In: Escola Regional de Informática São Paulo Oeste - ERI-SP-OESTE, 2, 2006, Marília, **Anais...** Marília, 2006, 9p. p.168-179.

VINCENZI, A. M. R. *Orientação a objetos: Definição e análise de recursos de teste e validação.*

APÊNDICE A - SCRIPT DO BANCO DE DADOS UTILIZADO

```
SET FOREIGN_KEY_CHECKS=0;
DROP TABLE IF EXISTS `produto`;
CREATE TABLE `produto` (
  `id_produto` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  `nome_produto` varchar(50) CHARACTER SET latin1 COLLATE latin1_swedish_ci NOT
NULL,
  `preco_venda` decimal(10,2) NOT NULL DEFAULT '0.00',
  `qtde_minima` int(10) NOT NULL DEFAULT '0',
  `qtde_total` int(10) DEFAULT '0',
  PRIMARY KEY(`id_produto`)
)
ENGINE=INNODB;

DROP TABLE IF EXISTS `venda`;
CREATE TABLE `venda` (
  `id_venda` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  `data_venda` date DEFAULT '0000-00-00',
  PRIMARY KEY(`id_venda`)
)
ENGINE=INNODB;



DROP TABLE IF EXISTS `descartados`;
CREATE TABLE `descartados` (
  `id_descartado` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  `id_produto_FK` int(10) NOT NULL,
  `data_descarte` date DEFAULT '0000-00-00',
  `observacao` text CHARACTER SET latin1 COLLATE latin1_swedish_ci,
  `qtde_descarte` int(10) UNSIGNED NOT NULL,
  PRIMARY KEY(`id_descartado`)
)
ENGINE=INNODB;
```

```
DELIMITER ||
CREATE TRIGGER `tr_descartados` AFTER INSERT
ON `descartados` FOR EACH ROW
BEGIN
  DECLARE id_prod INT;
    DECLARE qtde INT;
    SET qtde = NEW.qtde_descarte;
    SET id_prod = NEW.id_produto_FK;
    CALL `pr_diminui_produto`(qtde, id_prod);
END; ||
DELIMITER ;


DROP TABLE IF EXISTS `lote`;
CREATE TABLE `lote` (
  `id_lote` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  `desc_lote` text CHARACTER SET latin1 COLLATE latin1_swedish_ci,
  PRIMARY KEY(`id_lote`)
)
ENGINE=INNODB;


DROP TABLE IF EXISTS `venda_produto`;
CREATE TABLE `venda_produto` (
  `id_produto_FK` int(10) UNSIGNED NOT NULL DEFAULT '0',
  `id_venda_FK` int(10) UNSIGNED NOT NULL DEFAULT '0',
  `qtde_por_produto` int(10) UNSIGNED NOT NULL,
  `desconto` int(10),
  PRIMARY KEY(`id_produto_FK`, `id_venda_FK`),
  CONSTRAINT `Ref_03` FOREIGN KEY (`id_produto_FK`)
    REFERENCES `produto`(`id_produto`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION,
  CONSTRAINT `Ref_04` FOREIGN KEY (`id_venda_FK`)
    REFERENCES `venda`(`id_venda`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
)
ENGINE=INNODB;


DELIMITER ||
CREATE TRIGGER `tr_diminui_produto` AFTER INSERT
ON `venda_produto` FOR EACH ROW
BEGIN
  DECLARE id_prod INT;
    DECLARE qtde INT;

    SET qtde = NEW.qtde_por_produto;
    SET id_prod = NEW.id_produto_FK;
      CALL pr_diminui_produto(qtde, id_prod);
END; ||
DELIMITER ;



DROP TABLE IF EXISTS `produto_lote`;
CREATE TABLE `produto_lote` (
  `id_lote_FK` int(10) UNSIGNED NOT NULL DEFAULT '0',
  `id_produto_FK` int(10) UNSIGNED NOT NULL DEFAULT '0',
  `qtde_comprada` int(10) UNSIGNED NOT NULL,
  `preco_custo` decimal(10,2) DEFAULT '0.00',
  `data_chegada` date DEFAULT '0000-00-00',
  `data_vencimento` date DEFAULT '0000-00-00',
  PRIMARY KEY(`id_lote_FK`, `id_produto_FK`),
```

```
  CONSTRAINT `Ref_01` FOREIGN KEY (`id_lote_FK`)
    REFERENCES `lote`(`id_lote`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Ref_02` FOREIGN KEY (`id_produto_FK`)
    REFERENCES `produto`(`id_produto`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
)
ENGINE=INNODB;


DELIMITER ||
CREATE TRIGGER `tr_aumenta_produto` AFTER INSERT
ON `produto_lote` FOR EACH ROW
BEGIN
  DECLARE id_prod INT;
    DECLARE qtde INT;
     SET qtde = NEW.qtde_comprada;
     SET id_prod = NEW.id_produto_FK;
    CALL `pr_aumenta_produto`(qtde, id_prod);
END; ||
DELIMITER ;

SET FOREIGN_KEY_CHECKS=1;
DROP PROCEDURE IF EXISTS `pr_aumenta_produto`;

DELIMITER ||
CREATE PROCEDURE `pr_aumenta_produto` (IN `qtde` INT, IN `id_prod` INT)
LANGUAGE SQL
SQL SECURITY DEFINER
BEGIN
  UPDATE Produto SET `qtde_total` = `qtde_total` + `qtde`
      where Produto.id_produto = `id_prod`;
```

```
END; ||
DELIMITER ;


DROP PROCEDURE IF EXISTS `pr_diminui_produto`;
DELIMITER ||
CREATE PROCEDURE `pr_diminui_produto` (IN `qtde` INT, IN `id_prod` INT)
LANGUAGE SQL
SQL SECURITY DEFINER
BEGIN
  UPDATE Produto SET `qtde_total` = `qtde_total` - `qtde`
     WHERE id_produto = `id_prod`;
END; ||
DELIMITER ;
```

APÊNDICE B – INSERÇÕES INICIAIS NO BANCO DE DADOS

Tabela Produto
INSERT INTO produto VALUES('001', 'Chocolate', 10.25, 10, 0)
INSERT INTO produto VALUES('002', 'Biscoito', 20.00, 15, 0)
INSERT INTO produto VALUES('003', 'Leite Condensado', 30.00, 40, 0)
INSERT INTO produto VALUES('004', 'Sabonete', 4.50, 20, 0)
INSERT INTO produto VALUES('005', 'Iogurte', 15.00, 10, 0)
INSERT INTO produto VALUES('006', 'Mussarela', 21.00, 50, 0)
INSERT INTO produto VALUES('007', 'Sorvete', 41.00, 20, 0)
INSERT INTO produto VALUES('008', 'Leite', 53.00, 16, 0)
INSERT INTO produto VALUES('009', 'Sabão', 5.00, 18, 0)

Tabela Lote
INSERT INTO lote VALUES(1, 'A1');
INSERT INTO lote VALUES(2, 'B2');
INSERT INTO lote VALUES(3, 'C3');
INSERT INTO lote VALUES(4, 'D4');
INSERT INTO lote VALUES(5, 'E5');
INSERT INTO lote VALUES(6, 'C31');
INSERT INTO lote VALUES(7, 'H1');
INSERT INTO lote VALUES(8, 'G2');
INSERT INTO lote VALUES(9, 'F3');
INSERT INTO lote VALUES(10, 'T5');

Tabela Produto_Lote
INSERT INTO produto_lote VALUES(1, '001', 15, 6.50, '2009-01-05', '2009-07-05');
INSERT INTO produto_lote VALUES(2, '002', 25, 12.00, '2009-01-05', '2010-01-05');
INSERT INTO produto_lote VALUES(3, '003', 7, 18.00, '2009-01-05', '2009-05-05');
INSERT INTO produto_lote VALUES(4, '004', 40, 1.80, '2009-01-05', '2009-01-05');
INSERT INTO produto_lote VALUES(5, '005', 20, 3.20, '2009-01-05', '2010-01-05');
INSERT INTO produto_lote VALUES(6, '003', 5, 18.00, '2009-01-07', '2009-05-07');
INSERT INTO produto_lote VALUES(7, '001', 10, 6.75, '2009-01-07', '2009-11-05');
INSERT INTO produto_lote VALUES(8, '002', 8, 12.00, '2009-01-05', '2009-07-07');
INSERT INTO produto_lote VALUES(9, '003', 10, 18.00, '2009-01-06', '2009-08-07');
INSERT INTO produto_lote VALUES(10, '004', 5, 1.80, '2009-01-06', '2009-07-18');

Tabela Venda
INSERT INTO venda VALUES(2, '2009-01-09');
INSERT INTO venda VALUES(1, '2009-01-08');
INSERT INTO venda VALUES(3, '2009-01-10');
INSERT INTO venda VALUES(4, '2009-01-09');

```
INSERT INTO venda VALUES(5, '2009-01-08');
INSERT INTO venda VALUES(6, '2009-01-10');
INSERT INTO venda VALUES(7, '2009-01-09');
INSERT INTO venda VALUES(8, '2009-01-08');
INSERT INTO venda VALUES(9, '2009-01-10');
INSERT INTO venda VALUES(10, '2009-01-07');
```

Tabela Venda_Produto

```
INSERT INTO Venda_Produto VALUES('001', 1, 3, 0)
INSERT INTO venda_produto VALUES('002', 1, 1, 0)
INSERT INTO venda_produto VALUES('001', 2, 1, 0)
INSERT INTO venda_produto VALUES('004', 2, 1, 0)
INSERT INTO venda_produto VALUES('003', 3, 2, 0)
INSERT INTO venda_produto VALUES('005', 3, 2, 0)
INSERT INTO venda_produto VALUES('001', 4, 5, 0)
INSERT INTO venda_produto VALUES('001', 5, 3, 0)
INSERT INTO venda_produto VALUES('003', 5, 1, 0)
INSERT INTO venda_produto VALUES('005', 5,  1, 0)
INSERT INTO venda_produto VALUES('002', 6,  1, 0)
INSERT INTO venda_produto VALUES('003', 7,  1, 0)
INSERT INTO venda_produto VALUES('004', 7,  1, 0)
INSERT INTO venda_produto VALUES('005', 7,  1, 0)
INSERT INTO venda_produto VALUES('001', 8,  2, 0)
INSERT INTO venda_produto VALUES('002', 8,  1, 0)
```

Tabela Descartados

```
INSERT INTO Descartados VALUES(1,'006', curdate(), 'produto vencido', 2);
INSERT INTO Descartados VALUES(2,'003', curdate(), 'produto amassado', 2);
INSERT INTO Descartados VALUES(3,'008', curdate(), 'produto vencido', 5);
INSERT INTO Descartados VALUES(4,'005', curdate(), 'produto aberto', 7);
```