

**UNIVERSIDADE POSITIVO
ESPECIALIZAÇÃO EM BUSINESS INTELLIGENCE**

PATRICIO DOS SANTOS DA SILVA

**MODELAGEM DE DADOS PARA APLICAÇÃO DE BUSINESS ANALYTICS
UM CASO DE USO EM UMA IMOBILIARIA**

**Curitiba
2016**

PATRICIO DOS SANTOS DA SILVA

**MODELAGEM DE DADOS PARA APLICAÇÃO DE BUSINESS ANALYTICS
UM CASO DE USO EM UMA IMOBILIARIA**

Trabalho de conclusão de curso de especialização em Business Intelligence da Universidade Positivo de Curitiba.

Orientador: Marcelo Rosano Dallagassa

Curitiba

2016

RESUMO

Toda empresa ou departamento precisa encontrar respostas para suas perguntas, é uma necessidade do negócio, no entanto, a empresa ou departamento têm de reconhecer que precisa de algo para consolidar, gerir, apresentar e permitir a análise das informações que possui. Essa necessidade terá de ser um objetivo definido e encarado como estratégico. BI é uma abreviação para Business Intelligence, e é todo processo que envolve a habilidade de estruturar, acessar e explorar informações armazenadas em um banco de dados, data warehouse ou data mart, com o objetivo de desenvolver percepções, entendimentos e conhecimentos para produção de um melhor processo de análise e tomada de decisão. O presente trabalho tem como objetivo mostrar que o pré-requisito para um grande esforço de desenvolvimento é geralmente entender como o negócio funciona antes que se possa compreender como as ferramentas que auxiliarão o negócio deverão funcionar. A razão para fazermos a modelagem de dados é confirmar e documentar essa compreensão do negócio de diferentes perspectivas. Para alcançar nosso objetivo, construímos um sistema de banco de dados, onde foi preciso entender e documentar a ordem dos processos envolvidos, as regras do negócio, os dados e as relações entre eles, representadas em uma série de modelos de dados, cada um para sua finalidade. Verificaram-se ainda as possibilidades de trabalho com uma ferramenta analítica e um modelo multidimensional, evidenciando que um modelo relacional bem construído é um sólido alicerce para o desenvolvimento de um modelo dimensional bem-sucedido.

Palavras-Chave: Business Intelligence, data warehouse, modelagem de dados, modelo multidimensional.

ABSTRACT

Every company or department needs to find answers to your questions, it is a business necessity. However, the company or department must recognize that something is needed to consolidate, manage, display and allow analysis of the information available. This need must be an ultimate goal, seen as strategic. BI is an abbreviation for Business Intelligence, and it means every process that involves the ability to organize, access and explore information stored in a database, data warehouse or data mart, with the aim of developing perceptions, understandings and knowledge to produce a better analysis process and decision making. The current study aims to show that the prerequisite for a large development effort is usually understanding how the business works before understanding how the tools that will help the business should work. The reason to do the data modeling is to confirm and document this comprehension of the business from different perspectives. To achieve our goal, we build a database system where it was necessary to understand and document the sequence of processes involved, the business rules, data and relationships between them, represented in a series of data models, each one for its purpose. There was also possibilities of working with an analytic tool and a multidimensional model, showing that a well-built relational model is a solid foundation for the development of a successful dimensional model.

Keywords: Business Intelligence, data warehousing, data modeling, multidimensional model.

INDICE DE FIGURAS

Figura 1. Sistemas Isolados	21
Figura 2. Sistemas integrados com dados compartilhados	21
Figura 3. Diagrama simplificado de um sistema de banco de dados	24
Figura 4. A evolução dos modelos de banco de dados.....	25
Figura 5. Arquitetura de três esquemas	29
Figura 6. Estrutura de um Sistema de Banco de dados	31
Figura 7. Modelagem de dados.....	34
Figura 8. Exemplo de um modelo conceitual.....	37
Figura 9. Resumo da notação para diagramas ER proposta por Peter Chen.	38
Figura 10. Dois tipos de entidade e algumas entidades membro de cada uma	40
Figura 11. Duas entidades e seus atributos	41
Figura 12. Uma hierarquia de atributos compostos.....	43
Figura 13. Uma entidade com dois atributos-chave	46
Figura 14. Representação de um relacionamento entre duas entidades	48
Figura 15. Um tipo de relacionamento entre duas Entidades.....	50
Figura 16. Conjunto de relacionamento ternário	51
Figura 17. Cardinalidade máxima entre duas Entidades	52
Figura 18. Cardinalidade de mapeamento (a) 1:1, (b) 1:N, (c) N:1 e (d) N:N	54
Figura 19. Participação total em um conjunto de relacionamento	55
Figura 20. Participação parcial em um conjunto de relacionamento	56
Figura 21. Cardinalidade mínima entre duas entidades	57
Figura 22. Relacionamento entre duas entidades na notação (Min, Max)	58
Figura 23. Notação em diagrama de classes UML.....	60
Figura 24. Símbolos usados na notação do diagrama de classe UML.....	61
Figura 25. (a) Atributos e tuplas de uma relação, (b) O domínio de seus atributos...	62
Figura 26. História do modelo de banco de dados relacional.....	63
Figura 27. Etapas no processamento da consulta	64
Figura 28. Sistemas Gerenciadores de Bancos de dados Relacionais	81
Figura 29. Etapas de um processo de normalização	84
Figura 30. Características de um Data warehouse	98
Figura 31. Níveis de Granularidade.....	99
Figura 32. Interação entre um armazém de dados e Data Marts	101

Figura 33. Níveis de Arquitetura de um BI System.....	103
Figura 34. Tipos de consultas para diferentes níveis de dados.....	104
Figura 35. Um modelo de matriz bidimensional	107
Figura 36. Um modelo de Cubo de dados Tridimensional	108
Figura 37. Hierarquias de uma Dimensão Tempo e seus membros	110
Figura 38. Um modelo Dimensional em Star Schema.....	112
Figura 39. Arquitetura do ambiente BI proposto.....	115
Figura 40. Tela inicial do BrModelo	120
Figura 41. Criando uma entidade no BrModelo.....	121
Figura 42. Configurando a identificação de um objeto	121
Figura 43. Adicionando um atributo identificador	122
Figura 44. Configurando o atributo identificador.....	122
Figura 45. Nova entidade PROPOSTA criada.....	123
Figura 46. Adicionando relacionamento entre entidades no BrModelo	123
Figura 47. Criado um relacionamento entre duas entidades.....	124
Figura 48. Representação conceitual de duas entidades.....	124
Figura 49. DER das entidades Cliente e Proposta	126
Figura 50. DER das entidades Proposta e Vendedor.....	127
Figura 51. DER das entidades Proposta e Contrato	128
Figura 52. DER das entidades Contrato e Parcela.....	129
Figura 53. DER Inicial do Projeto da Imobiliária	130
Figura 54. Característica do minimundo modelado	131
Figura 55. DER atual do Projeto da Imobiliária	132
Figura 56. Diretório da aplicação DBDesignerFork	134
Figura 57. Tela inicial do DBDesigner Fork	135
Figura 58. Configuração do tipo de notação.....	135
Figura 59. Ícone para criação de nova tabela	136
Figura 60. Tabela criada na área de trabalho.....	136
Figura 61. Configurando a tabela selecionada no Table Editor.....	137
Figura 62. Configurando as colunas da tabela Cliente.....	138
Figura 63. Configuração da coluna que será a chave primária da tabela	138
Figura 64. Símbolo à esquerda do Atributo, indicando sua função	139
Figura 65.Tabela Cliente e seus Atributos	139
Figura 66. Alterando aparência das Tabelas para exibir domínio dos atributos	140

Figura 67. Salvando modelo.....	141
Figura 68. Tabela Cliente e nova tabela Dependente Criada.....	141
Figura 69. Relacionamentos que podem ser utilizados	142
Figura 70. Utilizando o relacionamento New 1:N Relation	143
Figura 71. Configurando as propriedades do relacionamento.....	144
Figura 72. Foreign Key criada automaticamente pelo DBDesigner Fork.....	145
Figura 73. Configuração das novas tabelas Proposta e Vendedor	146
Figura 74. Relacionamento entre Cliente, Proposta e Vendedor	147
Figura 75. Tabela Contrato e sua chave primária Num_contrato	147
Figura 76. Ícone de relacionamento "New 1:1 Non-Identifying-Relation"	148
Figura 77. Relacionamento entre Proposta e Contrato	149
Figura 78. Tabela Parcela e seus atributos	149
Figura 79. Relacionamento entre Contrato e Parcela	150
Figura 80. Tabelas Item Contrato, Lote e Loteamento	150
Figura 81. Relacionamento 1:N entre tabelas Contrato e Item Contrato	151
Figura 82. Relacionamento 1:N entre tabelas loteamento e lote.....	151
Figura 83. Relacionamento entre Item Contrato e Lote.....	152
Figura 84. Modelo Lógico Inicial.....	153
Figura 85. Modelo Lógico atual normalizado na 3FN	154
Figura 86. Página da Oracle para baixar o SGBD Express Edition	156
Figura 87. Conta Gratuita Oracle para baixar aplicações.....	156
Figura 88. Área de trabalho do Oracle SQL Developer.....	157
Figura 89. Exportando script para criação dos objetos	158
Figura 90. Configurando opções para exportar script de criação dos objetos.....	159
Figura 91. Copiando Script para o SQL Developer	160
Figura 92. Executando Script para criar objetos.....	161
Figura 93. Tabelas criadas pelo Oracle SQL Developer	162
Figura 94. Carga de dados na Tabela Loteamento	164
Figura 95. Carga de dados na Tabela Profissao	165
Figura 96. Carga de dados na Tabela Vendedor	165
Figura 97. Carga de dados na Tabela Lote	166
Figura 98. Carga de dados na Tabela Cidade	166
Figura 99. Carga de dados na Tabela Cliente.....	167
Figura 100. Carga de dados na Tabela Dependente	167

Figura 101. Carga de dados na Tabela Proposta.....	168
Figura 102. Carga de dados na Tabela Contrato	168
Figura 103. Carga de dados na Tabela Item Contrato	169
Figura 104. Carga de dados na Tabela Parcela.....	169
Figura 105. Job para executar todas as transformações de carga.....	170
Figura 106. Padronização opcional de nomenclatura	173
Figura 107. Dimensão Produto desnortinalizada	174
Figura 108. Esquema relacional da tabela Fato	178
Figura 109. Cardinalidade do modelo dimensional	180
Figura 110. Star Schema ODSLOTE	181
Figura 111. Indicadores no maior nível granular da dimensão Tempo.....	183
Figura 112. Aplicação de Drill-down I.....	183
Figura 113. Aplicação de Drill-down II.....	184
Figura 114. Aplicação de Drill-down III.....	184
Figura 115. Aplicação de Drill-down IV	185
Figura 116. Comparando o resultado mensal do 1º semestre de cada ano.....	185
Figura 117. Aplicando Roll-up no comparativo semestral de cada ano.....	186
Figura 118. Trocando de Dimensão localidade pela Dimensão Produto.....	186
Figura 119. Hierarquia configurada da dimensão tempo.....	187
Figura 120. Aproveitamento de Contratos em relação a propostas negociadas....	188
Figura 121. Drill-down da dimensão tempo para detalhar a análise.....	188
Figura 122. Análise com a adição do campo Nome_vendedor	189
Figura 123. Análise após aplicação de roll-up em período.....	190
Figura 124. Alterando parâmetros da análise	191
Figura 125. Aplicando drill-down em período	191
Figura 126. Análise sobre dimensão Produto.....	192
Figura 127. Adicionando detalhe sobre análise da dimensão Produto.....	193
Figura 128. Painel com Informações sobre os dados.	194

INDICE DE TABELAS

Tabela 1. Metadados de uma tabela do Banco de dados	22
Tabela 2. Classificação dos SGBDs.....	27
Tabela 3. Uma relação CLIENTE	65
Tabela 4. Nova relação criada a partir da relação CLIENTE	65
Tabela 5. Duas Relações, CLIENTE e FUNCIONARIO	68
Tabela 6. União sobre as relações CLIENTE e FUNCIONARIO	69
Tabela 7. Intersecção sobre as relações CLIENTE e FUNCIONARIO.....	69
Tabela 8. Subtração sobre as relações CLIENTE e FUNCIONARIO.....	70
Tabela 9. Subtração sobre as relações CLIENTE e FUNCIONARIO.....	70
Tabela 10. Duas Relações FUNCIONARIO e DEPENDENTE	71
Tabela 11. Produto cartesiano de FUNCIONARIO com DEPENDENTE	72
Tabela 12. Produto cartesiano com atributos chave combinados	73
Tabela 13. JUNÇÃO entre FUNCIONARIO e DEPENDENTE	75
Tabela 14. JUNÇÃO NATURAL entre FUNCIONARIO e DEPENDENTE	76
Tabela 15. LEFT JOIN entre FUNCIONARIO e DEPENDENTE	78
Tabela 16. RIGHT JOIN entre FUNCIONARIO e DEPENDENTE	78
Tabela 17. FULL JOIN entre FUNCIONARIO e DEPENDENTE	79
Tabela 18. Dependência funcional entre atributos de uma relação.....	82
Tabela 19. 1FN e uma tabela não normalizada - Layout 1.....	86
Tabela 20. 1FN e uma tabela não normalizada - Layout 2.....	86
Tabela 21. 1FN e uma tabela não normalizada - Layout 3.....	87
Tabela 22. 1FN nova relação criada	88
Tabela 23. Normalização na 1FN aplicada sobre a tabela CLIENTE	88
Tabela 24. Tabela na 1FN para aplicação da 2FN	89
Tabela 25. Valores exclusivos para atributos não chave em uma relação na 1FN ...	90
Tabela 26. Área do lote 2 conhecendo apenas o Código_lote	90
Tabela 27. Tabela com atributo caracterizando uma dependência parcial	90
Tabela 28. Uma tabela normalizada na 2FN	91
Tabela 29. Tabela com uma dependência funcional transitiva.....	92
Tabela 30. Uma tabela normalizada na 3FN	93
Tabela 31. Comparação entre OLTP e Data Warehouse.....	96
Tabela 32. Diferenças entre Data Mart (OLAP) e Data Warehouse	106

Tabela 33. Arquivos da Aplicação OLTP	116
Tabela 34. Metadados do arquivo de Clientes	117
Tabela 35. Metadados do arquivo de Proposta	117
Tabela 36. Metadados do arquivo de Contrato	118
Tabela 37. Metadados do arquivo de Parcelas	118
Tabela 38. Script de criação da Dimensão Tempo.....	172
Tabela 39. Exemplo de dados constantes na Dimensão Tempo	172
Tabela 40. Script de Criação da Dimensão Produto	174
Tabela 41. Script de Criação da Dimensão Vendedor	175
Tabela 42. Script de Criação da Dimensão Cliente.....	176
Tabela 43. Script de Criação da Dimensão Localidade.....	177
Tabela 44. Questões preliminares que o modelo deve responder	178
Tabela 45. Script de Criação da tabela Fato	179
Tabela 46. Script para atualizar modelo Dimensional ODSL0TE	181

LISTA DE SIGLAS

BD: Banco de dados

SGBD: Sistema Gerenciador de Banco de Dados

SQL: Structured Query Language - Linguagem de Consulta Estruturada

DDL: Data Definition Language – Linguagem de definição de dados

DML: Data Manipulation Language – Linguagem de manipulação de dados

DER: Diagrama Entidade Relacionamento

UML: Unified Modeling Language – Linguagem de modelagem unificada

BI: Business Intelligence – Inteligência de Negócios

OLAP: On-line Analytical Processing - Processamento analítico em tempo real

OLTP: Online Transaction Processing - Processamento de Transações em tempo real

DM: Data Mart – Repositório de dados Departamental

DW: Data Warehouse – Infraestrutura de dados corporativos

DSS: Decision Support System - Sistema de suporte à decisão

ETL: Extract Transform Load - Extração Transformação Carga

SUMÁRIO

1 INTRODUÇÃO.....	16
1.1 Tema proposto.....	16
1.2 Objetivos	17
1.2.1 Metodologia	17
2 FUNDAMENTAÇÃO TEÓRICA.....	18
2.1 Dados, informação e Conhecimento.....	18
2.1.1 Dados	18
2.1.2 Informação.....	19
2.1.3 Conhecimento.....	19
2.2 Banco de dados	20
2.2.1 Metadados	21
2.3 Sistema Gerenciador de Banco de Dados.....	22
2.3.1 Classificação dos SGBDs	24
2.3.2 Visão dos Dados.....	27
2.3.3 Componentes de um sistema de banco de dados	29
2.3.4 Data Definition Language (DDL)	30
2.3.5 Data Manipulation Language (DML)	30
2.4 Modelagem de dados	32
2.4.1 Modelo de dados	32
2.4.2 Levantamento e análise de requisitos.....	34
2.5 Modelo conceitual	35
2.6 O modelo Entidade-relacionamento.....	36
2.7 Diagramas de Entidade-relacionamento.....	37
2.8 Notação para Diagramas ER	38
2.9 Entidades.....	39
2.10 Atributos.....	40
2.11 Tipos de atributos	41
2.11.1 Atributos compostos versus Atributos simples (atômicos).....	42
2.11.2 Atributos de valor único versus multivalorados.....	43
2.11.3 Atributos armazenados versus derivados.....	44
2.11.4 Atributos de Valores Null	44
2.11.5 Atributos-chave.....	45

2.12 Entidades Fortes e Entidades Fracas	47
2.13 Relacionamentos e conjuntos de relacionamentos.....	48
2.13.1 Grau de um tipo de relacionamento	50
2.13.2 Restrições sobre tipos de relacionamentos binários	51
2.13.3 Razão de cardinalidade (Cardinalidade máxima)	52
2.13.4 Restrição de participação (Cardinalidade mínima)	54
2.14 Notações alternativas para Diagramas ER	58
2.14.1 Unified Modeling Language (UML)	58
2.15 Modelo de Banco de Dados Relacional.....	61
2.16 A Algebra relacional.....	63
2.17 Operações relacionais Unárias	66
2.18 Operações de álgebra relacional com base na teoria dos conjuntos	67
2.18.1 As operações UNIÃO, INTERSECÇÃO e SUBTRAÇÃO.	67
2.18.2 União (\cup)	68
2.18.3 Intersecção (\cap)	69
2.18.4 Diferença de Conjunto (ou Subtração) ($-$)	70
2.18.5 A operação PRODUTO CARTESIANO	71
2.19 Operações relacionais binárias.....	74
2.19.1 Junção(\bowtie).....	74
2.19.2 Variações de Junção: EQUIJUNÇÃO e JUNÇÃO NATURAL	75
2.19.3 Junção interna (INNER JOIN) e Junção externa (OUTER JOIN)	77
2.20 Sistemas Gerenciadores de Banco de Dados Relacionais (SGBDR).....	80
2.20.1 Objetivos de Bancos de dados Relacionais	81
2.20.2 Dependência funcional	82
2.21 Normalização de Relações	83
2.21.1 Uso das formas normais na prática	84
2.21.2 Primeira forma Normal (1FN)	85
2.21.3 Segunda forma Normal (2FN)	88
2.21.4 Terceira forma normal (3FN)	91
2.22 O ponto de partida	93
2.23 Data warehouses	96
2.24 Granularidade de Dados	99
2.25 Data Mart	100
2.26 DSS - Decision Support System	104

2.27	Tecnologia Multidimensional OLAP	105
2.28	Modelos Multidimensionais	107
2.28.1	Dimensão	108
2.28.2	Fatos	109
2.28.3	Hierarquia de Detalhes.....	109
2.29	Star Schema (Modelo Estrela).....	111
2.30	BI	112
3	ESTUDO DE CASO.....	113
3.1	Características da Empresa.....	113
3.2	Proposta de uma solução de BI.....	114
3.3	Análise de Requisitos	115
3.3.1	Metadados (Dicionário de dados) I	117
3.4	Modelagem conceitual	118
3.4.1	BrModelo	119
3.4.2	Diagrama Entidade Relacionamento	125
3.5	Modelo Lógico.....	132
3.5.1	DBDesigner 4 e DBDesigner Fork	133
3.5.2	Utilizando o DBDesigner Fork.....	134
3.6	Modelo físico.....	155
3.6.1	SGBD Oracle Express Edition	155
3.6.2	Oracle SQL Developer	157
3.6.3	DBDesigner e Scripts do modelo físico.....	158
3.6.4	ETL com Pentaho Data Integration.....	162
3.6.5	Carregando dados dos arquivos para as Tabelas	163
3.6.6	Conclusão da criação do banco de dados operacional	170
3.7	Modelagem Multidimensional	171
3.7.1	Dimensão Tempo (Quando)	171
3.7.2	Dimensão Produto (O quê)	173
3.7.3	Dimensão Vendedor (como)	175
3.7.4	Dimensão Cliente (Quem)	175
3.7.5	Dimensão Localidade (Onde)	176
3.7.6	Fato Indicadores (Métricas)	177
3.8	Utilizando o modelo multidimensional	182
3.8.1	O poder da do modelo multidimensional.....	182

4 CONCLUSÃO	195
REFERENCIAS.....	197

1 INTRODUÇÃO

Conforme relata Stair (2009), pessoas e organizações usam informações todos os dias. Muitas cadeias de varejo, por exemplo, coletam dados de suas lojas para ajudá-los a estocar o que os clientes desejam e desta forma, reduzir custos. Os componentes que são usados são frequentemente chamados de sistemas de informação.

Ainda conforme Stair (2009, p. 4), “um sistema de informação (SI) é um conjunto de componentes inter-relacionados que coletam, manipulam, armazenam e divulgam dados e informações e fornecem um mecanismo de feedback para atender a um objetivo”.

O centro de um sistema de informação é o dado, conforme relata Abreu e Machado (2004), devido ao crescimento natural dos negócios, das necessidades de informação e do aprimoramento dos sistemas de gerenciamento de banco de dados, a modelagem de dados passou a ser um fator fundamental no desenvolvimento de sistemas de informação.

Dados são gerados dentro e fora da organização a todo instante, a partir do momento em que a empresa nasce ela começa a gerar dados, estes dados dão origem à informação e é a eficácia no processo de transformação de dados em conhecimento que define o valor de uma solução de Business Intelligence. Quanto mais dados uma organização gera, mais ela depende de um sistema de informações.

Para assertividade nas decisões corporativas a organização precisa de dados confiáveis, quando os dados disponíveis são inconsistentes, duplicados ou ainda confusos, têm-se uma base duvidosa que compromete a precisão dos relatórios produzidos e consequentemente cria uma visão imprecisa e até mesmo incoerente sobre suas análises.

1.1 Tema proposto

A modelagem de dados é um processo utilizado para analisar e definir os requisitos de dados a fim de que se possa utilizá-los como recurso no gerenciamento do negócio. No âmbito dos sistemas de informação, modelos de

dados criam uma estrutura para os dados a serem utilizados, fornecendo definição e formato específico. Se um modelo de dados é elaborado de forma consistente entre os sistemas, então a compatibilidade e integridade da informação podem ser alcançadas, consequentemente todo o processo analítico e de tomada de decisão tornam-se favoráveis.

1.2 Objetivos

Modelagem de dados é repleta de terminologias e conceitos que soam como algo de outro mundo para alguns. O presente trabalho tem como objetivo oferecer entendimento de quais conceitos são fundamentais para que o leitor possa ter uma perspectiva de implementação de um banco de dados adequado às suas necessidades de negócio, bem como quais requisitos são importantes ou essenciais para o resultado desejado.

A pesquisa empreendida na parte teórica consiste em abordar de forma simples e clara todos os principais assuntos relacionados à implementação de um ambiente para análise de negócios, incluindo características, peculiaridades e diferenças entre bancos de dados OLTP, Relacionais e OLAP.

1.2.1 Metodologia

Para a fundamentação teórica necessária para aplicação dos procedimentos, será realizada a pesquisa bibliográfica em livros, artigos científicos e atualmente com material disponibilizado na Internet. A metodologia proposta, de ambientação até a elaboração do estudo de caso, é voltada para iniciantes no tema, com a introdução a vários conceitos essenciais para o entendimento do que é um processo de implementação de business analytics desde a concepção de um banco de dados relacional até o desenvolvimento de um modelo dimensional.

O objetivo é demonstrar que independente de qual tecnologia possa ser utilizada, o coração de um repositório de dados para aplicação de análises de negócios, é o dado em si, como os dados se relacionam, são armazenados e organizados pode ter impacto significativo sobre a eficiência do resultado esperado.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Dados, informação e Conhecimento

2.1.1 Dados

Como relata Stair (2009 p. 5), “dados consistem de fatos crus”, é a menor unidade de algo que existe, é um símbolo, uma imagem, uma palavra, um índice, um registro, ou seja, uma manifestação objetiva, passível de análise, exigindo interpretação da pessoa para sua manipulação. Dados isolados não possuem significado, têm pouco valor, mas quando classificados, armazenados e relacionados entre si, eles permitem a obtenção de informações.

Conforme Elmasri e Navathe (2011, p. 4) “dados são fatos conhecidos que podem ser registrados e possuem significado implícito”.

Segundo Inmon (2007), existem dois tipos básicos de dados, estruturados e não estruturados. Os dados estruturados são dados que chegam repetidamente com o mesmo formato e layout, exemplos típicos de dados estruturados incluem dados gerados por operações bancárias, companhias aéreas, operações de seguro, operações de varejo, etc. Os dados estruturados são convenientemente armazenados através de registros em uma base de dados, onde existem atributos, chaves, índices, tabelas, e assim por diante. O outro mundo é o de dados não estruturados, onde dados não estruturados existem em duas formas básicas, textuais e não textuais. Dados textuais não estruturados ocorrem em muitos lugares, e-mails, comentários em redes sociais, conversas telefônicas, apresentações em PowerPoint e muito mais. Dados não estruturados não textuais ocorrem como gráficos e imagens, incluindo, mas não se limitando a fotografias, diagramas e ilustrações.

Exemplo de Dado: 14,25%

2.1.2 Informação

Segundo Stair (2009), a informação é o dado inserido em um contexto, apresenta significado e intencionalidade, é o resultado do tratamento dos dados existentes acerca de alguém ou de alguma coisa.

Pode-se definir também a informação, conforme cita Stair (2009, p. 5), como “um conjunto de fatos organizados de tal forma que adquirem valor adicional além do valor do fato em si”. A informação não é necessariamente uma coisa boa, se uma pessoa não tem o poder de controlar a informação, com a capacidade de interpretar, compreender e usá-la, não há consequentemente o desenvolvimento de conhecimento, em outras palavras, informação sem o conhecimento não é muito útil.

Exemplo de informação:

14,25% é a taxa de juros do Brasil no ano de 2015.

2.1.3 Conhecimento

Conforme Stair (2009 p. 6), “conhecimento é a consciência e a compreensão de um conjunto de informações e a maneira com a qual elas podem ser úteis para apoiar uma tarefa específica ou chegar a uma decisão”.

O conhecimento é a interpretação e a absorção da informação. Basicamente é dividido em dois, o tácito e o explícito, o tácito é aquele que está na mente, aquele que o indivíduo adquire ao longo da vida, pela prática, experiência, geralmente é difícil de ser formalizado ou transmitido, este conhecimento não pode ser armazenado nos bancos de dados sem que seja transformado em explícito, o conhecimento explícito é aquele que pode ser formalizado, através de textos, diagramas, desenhos, etc. Pode-se dizer que conhecimento explícito se aproxima muito da informação, contudo o conhecimento explícito em seu conceito obtém-se ao interpretarmos a informação.

No exemplo abaixo, note que houve uma interpretação da informação, resgatou-se um conhecimento explícito armazenado em um banco de dados de nove anos para fazer este comparativo e transformá-la em conhecimento:

14,25% é a taxa de juros do Brasil no ano de 2015, o maior patamar em nove anos.

Exemplo de conhecimento tácito:

O executivo de marketing percebe que se o objetivo é vender para mães, comerciais de rádio funcionam muito melhor do que os de televisão, ele sabe que para mães ocupadas o rádio é uma companhia mais constante que a televisão.

2.2 Banco de dados

De acordo com Elmasri e Navathe (2011) um banco de dados é uma coleção de dados relacionados que representa algum aspecto do mundo real, às vezes chamado de minimundo ou universo de discurso. É uma coleção de dados logicamente coerente e com algum significado implícito. Uma variedade aleatória de dados não pode ser corretamente chamada de banco de dados. Um banco de dados é projetado, construído e populado com dados para uma finalidade específica e geralmente contém apenas informações relevantes para a empresa. Ele possui um grupo definido de usuários e algumas aplicações previamente concebidas nas quais estes usuários estão interessados. Em outras palavras, um banco de dados tem alguma fonte da qual o dado é derivado, algum grau de interação com eventos do mundo real e um público que está ativamente interessado em seu conteúdo.

“Banco de dados é um conjunto de dados integrados que tem por objetivo atender a uma comunidade de usuários”, relata Heuser (2008 p.6).

Elmasri e Navathe (2011) cita que um banco de dados pode ser gerado e mantido manualmente, como por exemplo, um livro de visitas de uma biblioteca, embora muito provavelmente hoje em dia este tipo de recurso já exista em um sistema de computador, neste caso, um banco de dados computadorizado é criado e mantido por um sistema gerenciador de banco de dados, um SGBD.

Conforme menciona Sudarshan e Silberschatz (2006), na abordagem de banco de dados uma das principais características é o compartilhamento de dados, um único repositório mantém dados que são configurados uma única vez, os tipos ou rótulos de dados definidos, que não mudam após a definição do metadados, serão acessados por vários usuários através de consultas, transações e aplicações.

Observe na figura 1, o cenário de uma empresa que não possui um banco de dados, repare que para cada área é criado um arquivo separado de produtos, além de não haver integração, há também o problema da redundância de dados, onde as informações referentes a um produto aparecem nos arquivos de produtos de cada um dos três sistemas.

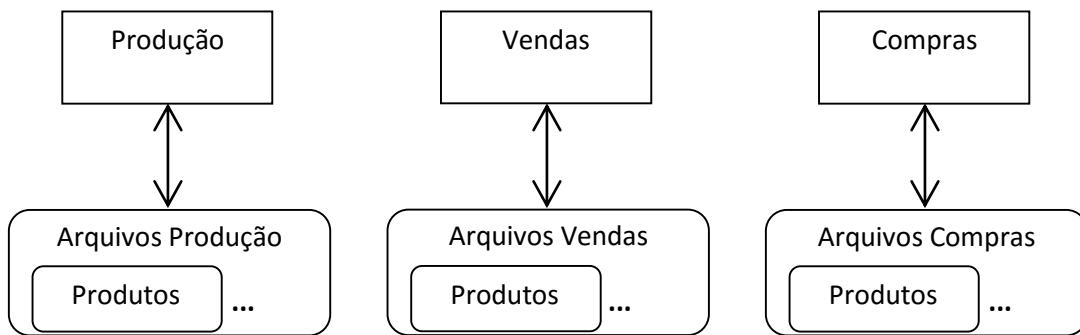


Figura 1. Sistemas Isolados
Fonte: Heuser (2008, p. 5).

Na Figura 2, cada informação é armazenada uma única vez, sendo acessada pelos vários sistemas que dela necessitam. Ao conjunto de arquivos de dados integrados e organizados dá-se o nome de banco de dados (BD).

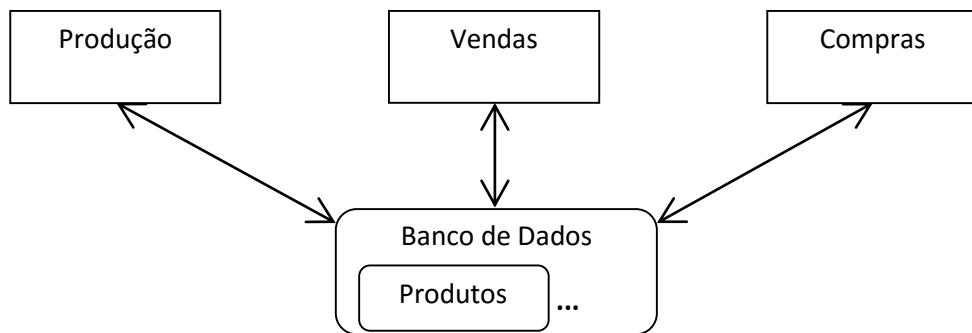


Figura 2. Sistemas integrados com dados compartilhados
Fonte: Heuser (2008, p. 4).

2.2.1 Metadados

Por definição, de acordo com Powell (2006), um banco de dados é um objeto estruturado, este objeto estruturado consiste de dados e metadados. Metadados, também chamado de dicionário de dados, é a descrição da estrutura e organização

dos dados. A informação é o dado real armazenado, como por exemplo, todos os nomes e endereços de seus clientes. O metadados descreve a estrutura aplicada pela base de dados para os dados do cliente. Em outras palavras, o metadados é a definição da tabela cliente. A definição da tabela do cliente contém os campos para os nomes e endereços, os comprimentos e tipos de dados de cada um desses campos, ao definir o tipo de dado, restringem-se os valores nos campos, como por exemplo, permitir apenas datas, para campos de data, apenas números para campos numéricos ou textos para campos de texto, se não houver metadados torna-se mais difícil para informar aos usuários e analistas como e onde estes campos e seus respectivos tipos de dados estão armazenados no banco de dados.

Nome Coluna	Tipo de Dados e Tamanho	Nulo	Num Coluna
ID_CLIENTE	NUMBER(38,0)	No	1
NOME_CLIENTE	VARCHAR2(255 BYTE)	Yes	2
LOGR_CLIENTE	VARCHAR2(255 BYTE)	Yes	3
NUMERO_LOGR_CLIENTE	VARCHAR2(10 BYTE)	Yes	4
COMPL_LOGR_CLIENTE	VARCHAR2(20 BYTE)	Yes	5
CEP_CLIENTE	NUMBER(38,0)	Yes	6
UF_CLIENTE	VARCHAR2(2 BYTE)	Yes	7
DDD_CLIENTE	NUMBER(38,0)	Yes	8
FONE_CLIENTE	NUMBER(38,0)	Yes	9
DT_NASC_CLIENTE	DATE	Yes	10
GRAU_INSTR_CLIENTE	VARCHAR2(20 BYTE)	Yes	11
PROFISSAO_CLIENTE	VARCHAR2(50 BYTE)	Yes	12
DATA_INC	DATE	Yes	13

Tabela 1. Metadados de uma tabela do banco de dados.

Fonte: Adaptado de Powell 2006, p. 7

2.3 Sistema Gerenciador de Banco de Dados

Conforme Elmasri e Navathe (2011, p. 3):

“Um sistema gerenciador de banco de dados (SGBD), do inglês, Database Management System (DBMS), é uma coleção de programas que permite aos usuários criar e manter um banco de dados. O SGBD é um sistema de software de uso geral que facilita o processo de definição, construção, manipulação e compartilhamento de bancos de dados entre diversos usuários e aplicações”.

De Acordo com Sudarshan e Silberschatz (2006), um sistema de gerenciamento de banco de dados é uma coleção de dados inter-relacionados e um

conjunto de programas para acessar esses dados. O acervo de dados, geralmente chamado de banco de dados, contém informações relevantes para a empresa. O principal objetivo de um SGBD é fornecer uma maneira de armazenar e recuperar informações do banco de dados de forma conveniente e eficiente.

Conforme Elmasri e Navathe, (2011), definir um banco de dados envolve especificar os tipos, estruturas e restrições dos dados a serem armazenados. A definição ou informação descritiva do banco de dados também é armazenada pelo SGBD na forma de um catálogo ou dicionário de dados, também chamado de metadados.

A construção do banco de dados é o processo de armazenar os dados em algum meio controlado pelo SGBD. A manipulação de banco de dados inclui funções como consultas para recuperar dados específicos, atualização do banco de dados para refletir mudanças no minimundo e geração de relatórios com base nos dados. O compartilhamento de um banco de dados permite que diversos usuários e programas acessem-no simultaneamente.

De acordo com Sudarshan e Silberschatz (2006), um programa de aplicação acessa o banco de dados ao enviar solicitações de dados (consultas) e transações ao SGBD. Uma consulta normalmente resulta na recuperação de alguns dados, já uma transação pode fazer com que alguns dados sejam lidos e outros gravados no banco de dados, essas consultas e transações precisam ser especificadas corretamente na linguagem de consulta do SGBD antes de serem processadas.

Outras funções importantes fornecidas pelo SGBD incluem proteção do banco de dados e sua manutenção por um longo período. A proteção inclui proteção do sistema contra defeitos (ou falhas) de hardware ou software e proteção de segurança contra acesso não autorizado ou malicioso. Um banco de dados grande pode ter um ciclo de vida de muitos anos, de modo que o SGBD precisa ser capaz de manter o sistema, permitindo que ele evolua à medida que os requisitos mudam com o tempo. Neste contexto, podemos chamar a união do banco de dados com o software de SGBD de sistema de banco de dados (Elmasri e Navathe, 2011).

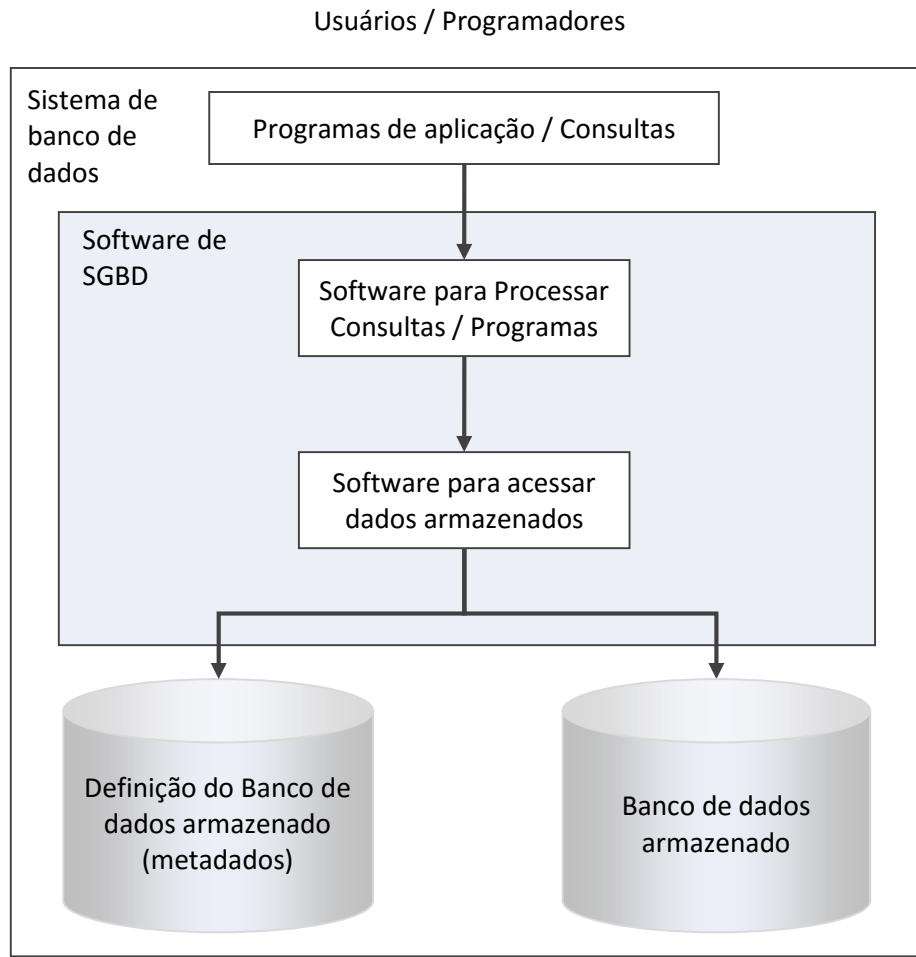


Figura 3. Diagrama simplificado de um sistema de banco de dados
Fonte: Adaptado de Elmasri e Navathe (2011, p. 4).

2.3.1 Classificação dos SGBDs

Conforme Elmasri e Navathe (2011), vários critérios são normalmente utilizados para classificar os sistemas gerenciadores de banco de dados. O primeiro e principal é o modelo de dados no qual o SGBD é baseado, isso porque o modelo de dados define como os dados serão armazenados no banco de dados. O modelo de dados mais utilizado atualmente em muitos SGBDs comerciais é o modelo de dados relacional. O modelo relacional é o sucessor dos modelos de dados hierárquicos e de rede, muitas aplicações ainda rodam em sistemas de banco de dados baseados nestes modelos, também chamados de sistemas de banco de dados legados.

O modelo de dados de objeto, segundo Elmasri e Navathe (2011), foi implementado em alguns sistemas comerciais, mas não tem seu uso generalizado,

este modelo define o banco de dados em termos de objetos, suas propriedades e operações. Os objetos com uma mesma estrutura e comportamento pertencem a uma classe, e as classes são organizadas em hierarquias. As operações de cada classe são especificadas com procedimentos predefinidos chamados métodos. Os autores citam ainda que os SGBDs relacionais têm estendido seus modelos para incorporar conceitos de banco de dados de objeto e outras funcionalidades.

De acordo com Elmasri e Navathe (2011), os SGBDs relacionais estão evoluindo continuamente e, em particular, têm incorporado muitos dos conceitos que foram desenvolvidos nos bancos de dados de objeto, isso tem levado a uma nova classe de SGBDs, chamada SGBDs objeto-relacionais. Os modelos de dados mais conhecidos são o relacional, objeto, objeto-relacional, hierárquico e de rede.

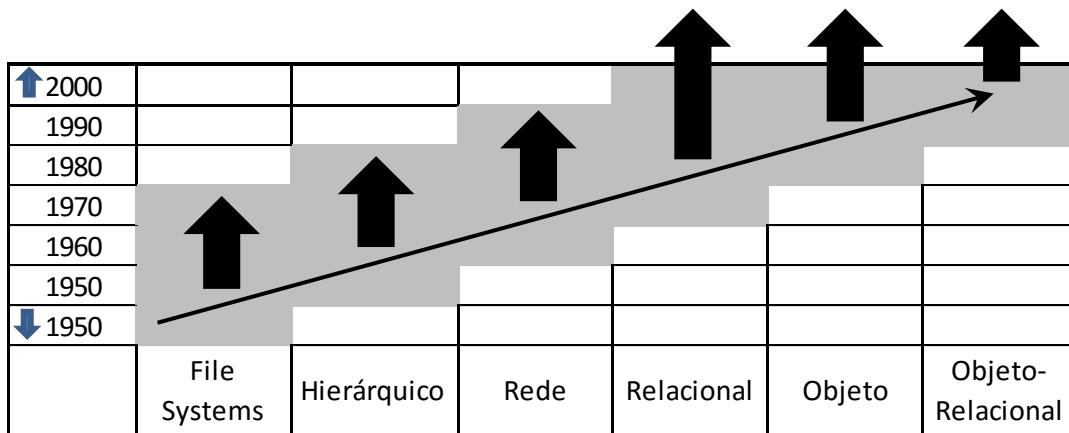


Figura 4. A evolução dos modelos de banco de dados
Fonte: Adaptado de Powell 2006, p. 7.

De acordo com Elmasri e Navathe (2011) o segundo critério usado para classificar SGBDs é o número de usuários suportados pelo sistema. Sistemas monousuários admitem apenas um usuário de cada vez e são usados principalmente com computadores convencionais, já os sistemas multiusuários, que incluem a maioria dos SGBDs, admitem múltiplos usuários simultaneamente e são usados em um ou mais servidores específicos.

O terceiro critério é o número de locais sobre os quais o banco de dados está distribuído. Um SGBD é classificado como centralizado se os dados estiverem armazenados em um único computador, observe que um SGBD centralizado pode atender a vários usuários, mas o SGBD e o banco de dados residem integralmente em um único computador. Um SGBD é classificado como distribuído, (SGBDD), se o

banco de dados e o software de SGBD estão distribuídos por vários locais, conectados por uma rede de computadores. Muitos SGBDDs utilizam arquitetura cliente/servidor. Os sistemas de banco de dados distribuídos podem ainda ser classificados em homogêneos, quando usam o mesmo software de SGBD em todos os locais, ou heterogêneos, quando usam um software de SGBD diferente em cada local (Elmasri e Navathe 2011).

É difícil propor uma classificação de SGBDs com base no custo, no entanto, este é o quarto critério de classificação. Atualmente temos SGBDs de código aberto, gratuitos, como MySQL e PostgreSQL, que têm suporte de fornecedores terceirizados com serviços adicionais. Também temos SGBDRs em versões gratuitas para testes, geralmente limitadas por tamanho ou por um período de 30 dias, é nesta última modalidade que estão disponíveis os principais sistemas gerenciadores de banco de dados relacionais. Os SGBDs maiores são vendidos na forma de licenças, que podem ser por local, ou por usuário. As licenças por local permitem uso ilimitado do SGBD com qualquer número de cópias rodando na instalação definida pelo comprador já as licenças por usuário limitam o número de usuários simultâneos ou o número total de usuários em determinado local. As versões de alguns sistemas para um único usuário isolado, como o Microsoft Access, são vendidas por cópia ou incluídas na configuração geral do computador. Observe que dependendo do tamanho e complexidade do projeto, a instalação e manutenção de um sistema de banco de dados podem exigir custos baixos ou o pagamento de milhões anualmente, (Elmasri e Navathe 2011).

“Também podemos classificar um Sistema gerenciador de banco de dados com base nas opções de tipos de caminho de acesso para armazenar arquivos. Uma estrutura bem conhecida de SGBDs é baseada em estruturas invertidas de arquivo”. (Elmasri e Navathe (2011, p. 33).

Por fim, para encerrarmos este tema, o último critério de classificação de um SGBD é quanto à sua finalidade que pode ser de uso geral ou uso especial. Segundo Elmasri e Navathe (2011), um SGBD é de uso especial quando ele é projetado e construído para uma aplicação específica, esse sistema não pode ser usado para outras aplicações sem mudanças relevantes. Por exemplo, um sistema de vendas, com cadastramento e alteração de dados de clientes é construído sobre um SGBD de uso especial, isso porque o desempenho é um dos aspectos mais

importantes. Estes SGBDs encontram-se na categoria de sistemas de processamento de transação on-line (OLTP – Online Transaction Processing), que precisam dar suporte a um grande número de transações simultâneas sem causar atrasos excessivos.

Inmon (2002), afirma que no ambiente OLTP, o tempo de resposta é quase sempre de missão crítica. O negócio começa a sofrer imediatamente quando o tempo de resposta se torna ruim.

Os SGBDs podem ser classificados em critérios como:

Modelo de dados	Relacionais		
	Orientados a Objetos		
	Objeto-Relacionais		
	Hierárquico		
	Rede		
	XML		
Nº de usuários	Monousuários		
	Multiusuários		
Nº de locais	Centralizados		
	Distribuídos	Homogêneos	
Custo	Baixo Custo	Gratuitos	
	Alto Custo	Licenças	Por local
			Por Usuário
Caminho de Acesso	Tipos de caminhos de acesso para armazenar arquivos		
Uso	Geral	Diversas Aplicações	
	Especial	Aplicação específica	

Tabela 2. Classificação dos SGBDs

Fonte: Autor

2.3.2 Visão dos Dados

Um sistema de banco de dados é uma coleção de dados inter-relacionados (BD) e um conjunto de programas que permitem aos usuários acessar e modificar esses dados (SGBD). Conforme Sudarshan e Silberschatz (2006), uma importante finalidade de um sistema de banco de dados é fornecer aos usuários uma visão abstrata dos dados, ou seja, o sistema oculta detalhes de como os dados são

armazenados e mantidos. Como muitos usuários de sistemas de banco de dados não são analistas de sistemas ou DBAs, os desenvolvedores ocultam dos usuários a complexidade da estruturação dos dados para simplificar as interações do usuário com o sistema. Essa abstração de dados é dividida em três níveis, físico, lógico e de view, também chamada de arquitetura de três esquemas onde Elmasri e Navathe (2011), também menciona que são respectivamente descritos como interno conceitual e externo.

De acordo com Elmasri e Navathe (2011, p. 19):

“Uma característica fundamental da abordagem de banco de dados é que ela oferece algum nível de abstração de dados. A abstração de dados, geralmente, se refere à supressão de detalhes da organização e armazenamento dos dados, destacando recursos essenciais para um melhor conhecimento desses dados. Uma das principais características da abordagem de banco de dados é possibilitar a abstração de dados, de modo que diferentes usuários possam percebê-los em seu nível de detalhe preferido”.

O nível físico, ou nível interno, descreve a estrutura do armazenamento físico do banco de dados, é o nível de abstração mais baixo, descreve como os dados são realmente armazenados assim como os caminhos de acesso para o banco de dados.

Segundo Elmasri e Navathe (2011), o nível lógico, ou nível conceitual, descreve que dados estão armazenados no banco de dados e que relações existem entre eles. Os administradores de banco de dados que precisam decidir que informações armazenar no banco de dados, usam o nível lógico de abstração, isso porque o nível lógico tem um esquema conceitual, que descreve a estrutura do banco de dados inteiro para uma comunidade de usuários. De forma mais sucinta Sudarshan e Silberschatz (2006), explica que o nível lógico, ou conceitual, oculta detalhes das estruturas de armazenamento físico e se concentra na descrição de entidades, tipos de dados, relacionamentos, operações do usuário e restrições. Normalmente, um modelo de dados representativo é usado para descrever o esquema conceitual quando um sistema de banco de dados é implementado.

Mesmo que o nível lógico use estruturas mais simples, a complexidade permanece devido à variedade de informações armazenadas em um grande banco de dados. Muitos usuários do sistema de banco de dados não precisam de toda essa informação, em vez disso eles precisam acessar apenas uma parte do banco.

Conforme Sudarshan e Silberschatz (2006), o nível de view é o nível de abstração mais alto, descreve apenas parte do banco de dados, existe para simplificar a interação do usuário com o sistema, inclui uma série de esquemas externos ou visões do usuário. Cada esquema externo descreve a parte do banco de dados em que um grupo de usuários em particular está interessado e oculta o restante do banco de dados do grupo de usuários.

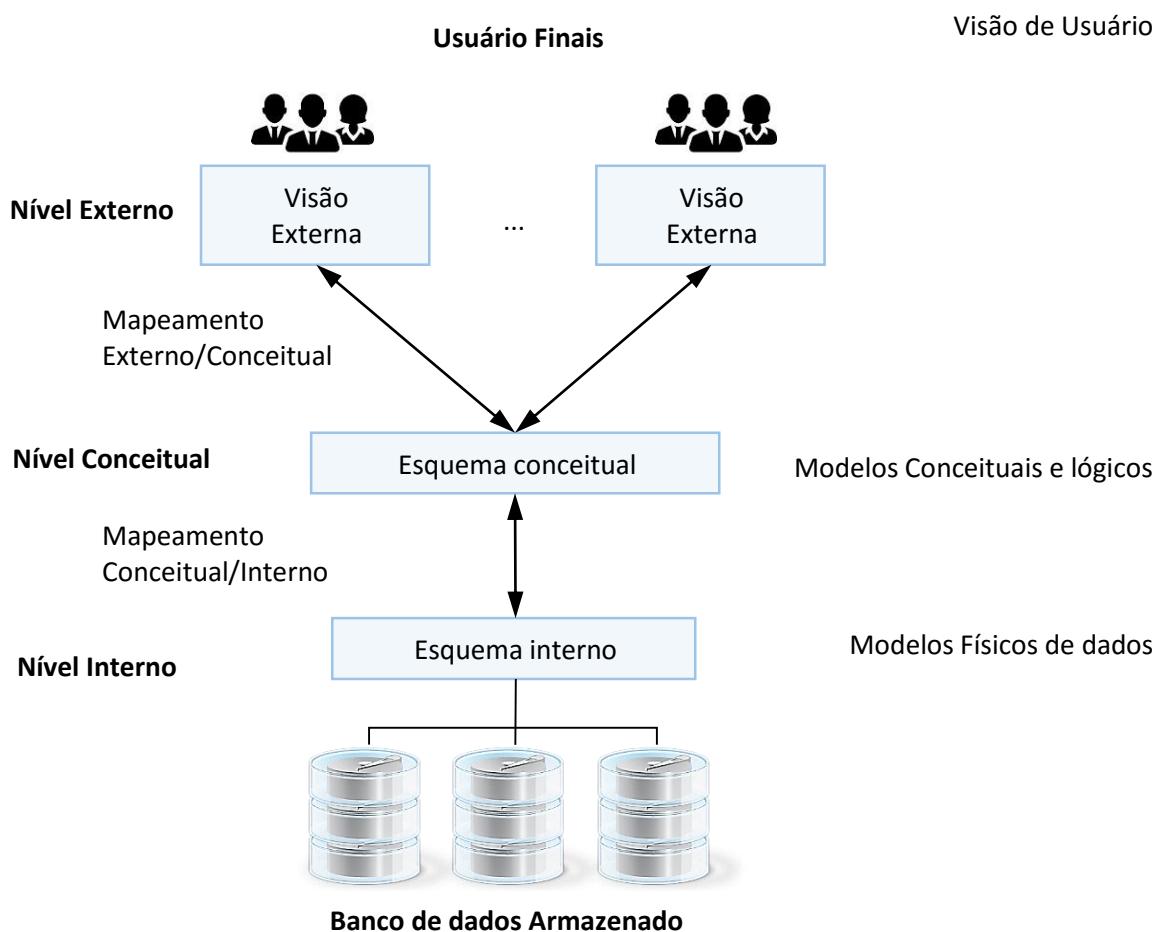


Figura 5. Arquitetura de três esquemas

Fonte: Adaptado de Elmasri e Navathe (2011, p. 22) e Sudarshan e Silberschatz, (2006 p. 4).

2.3.3 Componentes de um sistema de banco de dados

Um sistema de banco de dados possui vários subsistemas, ele é particionado em módulos que lidam com cada uma das responsabilidades do sistema geral. Os componentes funcionais podem ser divididos nos componentes gerenciador de armazenamento e processador de consulta.

De acordo com Sudarshan e Silberschatz (2006), o subsistema gerenciador de armazenamento fornece a interface entre os dados de baixo nível (físico) armazenados no banco de dados e os programas de aplicação e consulta submetidos ao sistema. O subsistema processador de consulta compila e executa instruções DDL e DML. O gerenciador de transação garante que o banco de dados permaneça em um estado consistente apesar de falhas no sistema e também assegura que as execuções de transação concorrentes sejam efetuadas sem conflitos.

2.3.4 Data Definition Language (DDL)

O termo DDL vem do inglês, Data Definition Language, significa linguagem de definição de dados, conforme Sudarshan e Silberschatz (2006, p. 4), “uma linguagem de definição de dados (DDL) é uma linguagem para especificar o esquema de banco de dados, bem como outras propriedades dos dados”. É usada para especificar a estrutura ou esquema do banco de dados permitindo alterar propriedades como estrutura de armazenamento e métodos de acesso aos dados. Permite ao usuário criar objetos no banco de dados (Create), alterar a estrutura da base de dados (Alter), remover todos os registros de uma tabela (Truncate), adicionar comentários ao dicionário de dados (Comment), eliminar um objeto (Drop), renomear um objeto (Rename), entre outros.

2.3.5 Data Manipulation Language (DML)

O termo DML vem do inglês, Data Manipulation Language, significa Linguagem de manipulação de dados, Sudarshan e Silberschatz (2006, p. 4), cita que “uma linguagem de manipulação de dados permite aos usuários acessar ou manipular dados”. Comandos DML permitem aos usuários acessar ou manipular dados conforme são organizados pelo modelo de dados implementado. Permite ao usuário recuperar dados do banco de dados (Select), inserir dados em uma tabela (Insert), atualizar os dados existentes em uma tabela (Update) e excluir registros de uma tabela (Delete).

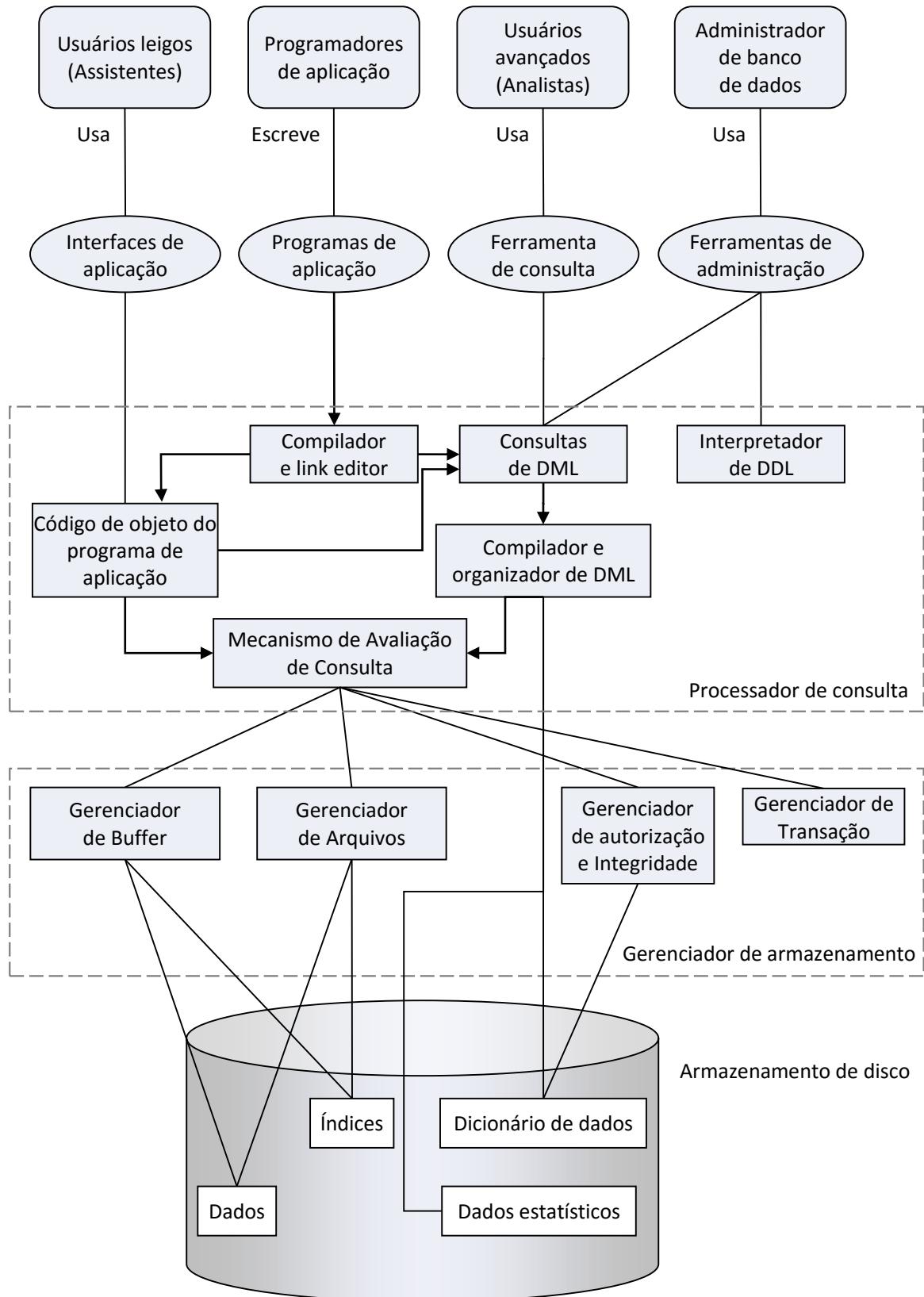


Figura 6. Estrutura de um Sistema de Banco de dados

Fonte: Adaptado de Elmasri e Navathe (2011, p. 22) e Sudarshan e Silberschatz, (2006 p. 4).

2.4 Modelagem de dados

Conforme relata Abreu e Machado (2004, p. 21), “Toda realidade é sempre, em princípio, bastante nebulosa e informal. Através da observação podemos extrair dela fatos que nos levam a conhecê-la de uma forma mais organizada”.

A modelagem de dados é um artifício utilizado para não entrarmos demais na complexidade do mundo real, quanto mais próximo do mundo real, mais abstrato será. É utilizada para definir e analisar os requisitos de dados necessários para apoiar os processos de negócios, no âmbito dos sistemas de informação correspondentes.

De acordo com Hoberman (2014), modelagem de dados é o processo de aprendizagem sobre os dados e o modelo de dados é o resultado final desse processo.

Por exemplo, sua empresa possui o departamento de vendas, recursos humanos, o setor financeiro, marketing, etc, dependendo do tamanho da organização, você dificilmente criará um modelo para o sistema completo de sua empresa, isso resultaria em um modelo muito extenso e complexo. O ideal é criar modelos em módulos, ou seja, criar um modelo para apenas uma parte do sistema, ou uma funcionalidade, lembre-se que o objetivo é, conforme cita Stair (2009, p. 6), “obter a consciência e a compreensão de um conjunto de informações e a maneira com a qual elas podem ser úteis para apoiar uma tarefa específica ou chegar a uma decisão”, desta forma, de nada adianta a modelagem de um sistema de grande porte desnecessário e difícil de interpretar.

2.4.1 Modelo de dados

Apoiando a estrutura de um banco de dados está o modelo de dados, conforme Sudarshan e Silberschatz (2006), é o conjunto de conceitos que podem ser usados para descrever a estrutura, tipos de dados, relacionamentos e restrições de um banco de dados.

Conforme Stair (2009), um modelo de dados é uma descrição precisa dos dados armazenados para um problema do mundo real, no entanto, o modelo de

dados não é uma descrição exata de uma situação real. Ele será sempre com base em definições e pressupostos, e tem um finito escopo.

De acordo com Sudarshan e Silberschatz (2006), um modelo é a representação abstrata e simplificada de uma determinada realidade, com a qual se deseja explicar ou testar o seu comportamento em sua totalidade ou em partes antes de sua existência real. É destinado principalmente para o processo de criação de bancos de dados e foi desenvolvido para facilitar o projeto, permitindo a especificação de um esquema empresarial. Esse esquema representa a estrutura lógica geral do banco de dados desde o nível conceitual até o nível físico.

Para chegarmos ao ponto das implementações, onde saímos do abstrato, fazemos uma coisa chamada modelagem conceitual, ou modelo de dados de alto nível, é uma descrição do banco de dados de forma independente de implementação em um SGBD, é mais voltada aos conceitos do negócio, pois oferecem conceitos que são próximos ao modo como muitos usuários percebem os dados.

Heuser (2008), explica que a técnica mais difundida de modelagem conceitual é a abordagem entidade-relacionamento (ER), nesta técnica, um modelo conceitual é usualmente representado através de um diagrama, chamado diagrama entidade-relacionamento (DER).

Depois do modelo conceitual há o modelo lógico. Um modelo lógico é uma descrição do banco de dados no nível de abstração visto pelo usuário do SGBD. No nível lógico precisamos definir em qual tipo de banco de dados trabalharemos, se é um BD relacional, se é um BD objeto relacional, se é um banco de dados orientado a objetos, todos estes tipos de bancos de dados precisam ser definidos no nível lógico que, dependendo das escolhas, levam a um caminho ou a outro, neste trabalho utilizaremos um modelo lógico referente a um SGBD relacional, onde os dados estão organizados na forma de tabelas.

Depois do modelo conceitual e do modelo lógico, há o modelo físico, também chamado de modelo de dados de baixo nível, o nível físico é o mapeamento do modelo lógico em um esquema físico de acordo com o SGBD específico, inversamente ao modelo conceitual, os conceitos oferecidos pelos modelos físicos são voltados para os administradores de banco de dados, não para usuários finais, é o nível dos códigos, das linguagens, da criação dos objetos.

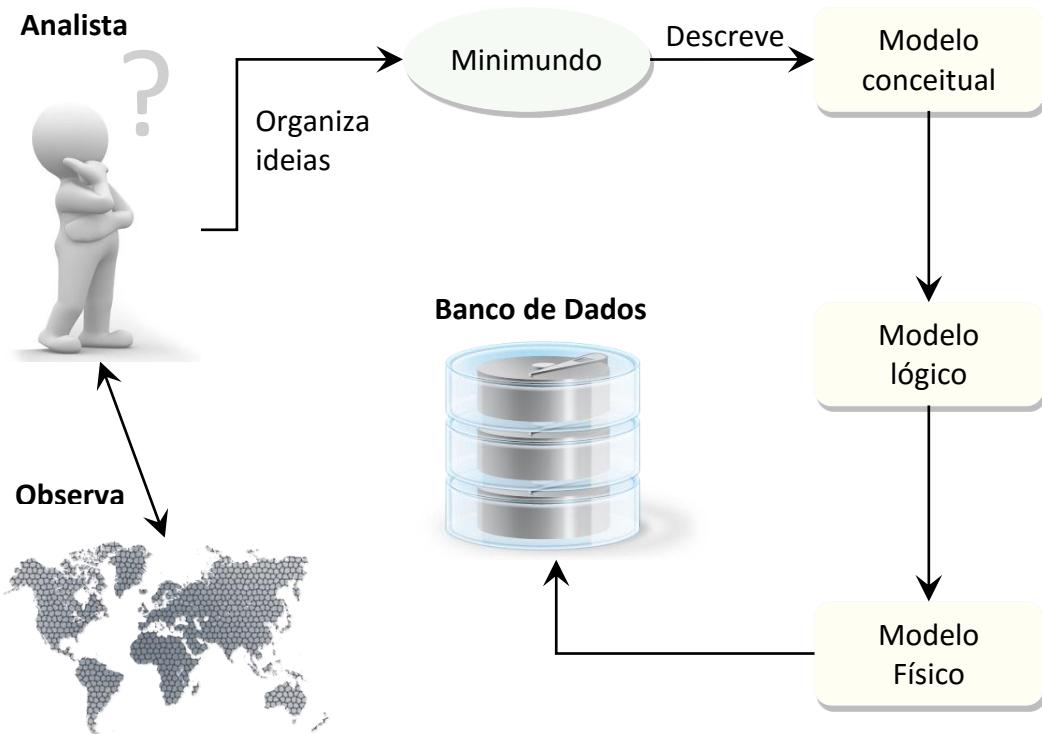


Figura 7. Modelagem de dados
Fonte: Autor

2.4.2 Levantamento e análise de requisitos

De acordo com Elmasri e Navathe (2011, p. 132):

A primeira etapa do projeto de um banco de dados é o levantamento e análise de requisitos. Durante esta etapa, os projetistas de banco de dados entrevistam os usuários solicitantes para entenderem e documentarem seus requisitos de dados. O resultado desta etapa é um conjunto de requisitos dos usuários, escrito de forma concisa. Esses requisitos devem ser especificados da forma mais detalhada e completa possível. Em paralelo com as especificações dos requisitos de dados, é útil determinar os conhecidos requisitos funcionais da aplicação. Estes consistem em operações ou transações definidas pelo usuário, que serão aplicadas ao banco de dados, incluindo recuperações e atualizações. Assim que os requisitos tiverem sido levantados e analisados, a próxima etapa é criar um esquema conceitual para o banco de dados, usando um modelo de dados conceitual.

A melhor maneira de entender o que você precisa projetar e construir é pesquisar e praticamente interrogar os clientes. Embora a palavra “interrogar” soe como uma palavra hostil, para fazer o trabalho direito e evitar retrabalho, é preciso muito mais do que trocas de e-mails e reuniões. Aprender sobre os requisitos dos clientes pode ser um processo longo e cansativo. Pode levar dias ou até mesmo

semanas de pesquisa e mapeamento, estudando as práticas existentes, debruçado sobre inúmeras planilhas e outros arquivos corporativos e espionando os clientes enquanto eles fazem suas tarefas diárias. É importante saber que é fundamental a documentação de todo este processo, quando isso acabar, os clientes não odiarão você, mas eles podem gostar que você vá embora e deixe-os sozinhos por um tempo.

Uma boa sessão de perguntas e respostas deixa todo mundo esgotado, mas clientes que são realmente dedicados à empresa geralmente estão dispostos a colaborar até com a mais específica pergunta. A idéia é que quanto mais cedo você identificar potenciais problemas e quanto mais você conhecer e identificar as funcionalidades do projeto, mais fácil será para você desenvolver para eles e menos eles estragão o seu planejamento. Os seus encontros iniciais com o cliente darão a sua primeira oportunidade para abordar estas questões para que eles não o surpreendam mais tarde, neste caso, quando você começar um projeto, entenda os clientes, conheça-os e também o que eles fazem, mesmo se o problema que você está tentando resolver é apenas uma pequena parte do negócio.

2.5 Modelo conceitual

De acordo com Elmasri e Navathe (2011), o objetivo da modelagem conceitual é obter uma descrição abstrata dos dados que serão armazenados no banco de dados, independente de implementação em computador. O modelo conceitual é uma descrição concisa dos requisitos de dados dos usuários, como não incluem descrições detalhadas de implementação, este modelo é mais fácil de entender e pode ser usado para a comunicação com usuários não técnicos. O esquema conceitual também pode ser utilizado como uma referência para garantir que todos os requisitos de dados dos usuários sejam atendidos e que não estejam em conflito. Essa técnica permite que os projetistas de banco de dados se concentrem em especificar as propriedades dos dados, sem se preocuparem com detalhes de armazenamento e implementação.

Conforme Heuser (2008) a técnica mais difundida de modelagem conceitual é a abordagem entidade-relacionamento (ER). Nesta técnica, o modelo de dados é representado através de um modelo entidade-relacionamento (MER), que descreve

os objetos (entidades) envolvidos em um domínio de negócios, com suas características (atributos) e como elas se relacionam entre si (relacionamentos). Usualmente, um modelo ER é representado graficamente através de um diagrama entidade-relacionamento (DER). Enquanto o MER é um modelo conceitual, o diagrama entidade relacionamento (DER) é a sua representação gráfica e principal ferramenta.

2.6 O modelo Entidade-relacionamento

De acordo com Sudarshan e Silberschatz, (2006). O modelo de dados entidade-relacionamento (ER) foi desenvolvido para facilitar o projeto de banco de dados, permitindo especificação de um esquema de empresa que representa a estrutura lógica geral de um banco de dados. O modelo ER é muito útil no mapeamento dos significados e interações de empresas reais para um esquema conceitual. Devido a essa utilidade, muitas ferramentas de projeto de bancos de dados utilizam conceitos do modelo ER, isso porque os vários recursos do modelo ER oferecem ao projetista de banco de dados inúmeras escolhas de como representar melhor a empresa a ser modelada. Muitas vezes, o projetista precisa pesar as vantagens de um modelo simples e compacto com as de um modelo mais preciso, porém, mais complexo.

Conforme relata Heuser (2008), a abordagem ER foi criada em 1976 por Peter Chen no artigo “The Entity-Relationship Model – Toward a Unified View of Data”, base do modelo entidade-relacionamento (MER). Ela pode ser considerada como um padrão para modelagem conceitual. Mesmo as técnicas de modelagem orientada a objetos que têm surgido nos últimos anos baseiam-se nos conceitos da abordagem entidade-relacionamento.

A notação proposta por Peter Chen é composta de entidades representadas por retângulos, relacionamentos representados por losangos, atributos por elipses e linhas de conexão por linhas que indicam a cardinalidade de uma entidade em um relacionamento.

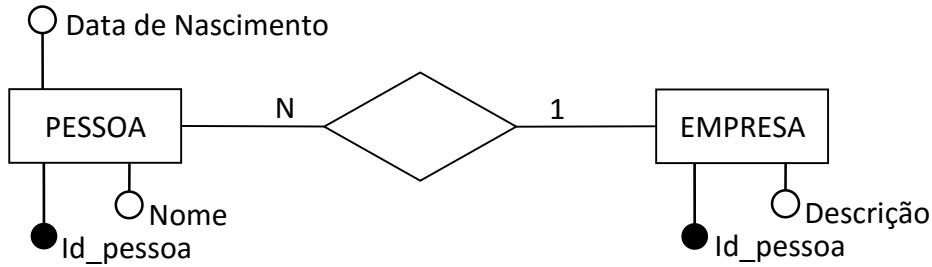


Figura 8. Exemplo de um modelo conceitual
Fonte: Adaptado de Heuser (2008, p. 6).

2.7 Diagramas de Entidade-relacionamento

Um diagrama ER pode expressar graficamente a estrutura lógica geral de um banco de dados.

De acordo com Heuser (2008), um diagrama ER é apresentado na forma de um grafo. A distribuição dos símbolos de DER no papel é totalmente arbitrária e não tem maior significado do ponto de vista formal. Entretanto, para tornar o diagrama mais legível é usual evitar-se cruzamentos de linhas. Para isso, a recomendação geral é a de posicionar os retângulos representativos de entidades que participam de muitos relacionamentos no centro do diagrama.

De acordo com Sudarshan e Silberschatz (2006), a escolha de nomes para tipos de entidade, atributos, tipos de relacionamento e particularmente funções nem sempre é simples. É preciso escolher nomes que transmitam, tanto quanto possível, os significados conectados às diferentes construções no esquema. Recomenda-se usar nomes no singular para os tipos de entidade, isso porque o nome se aplica a cada entidade individual pertencente a este tipo de entidade. Os nomes dos tipos de entidade e tipos de relacionamento são escritos em letras maiúsculas, preferencialmente o nome dos relacionamentos são compostos por verbos de ligação, os nomes de atributos têm apenas a letra inicial em maiúscula.

A figura 9 mostra os principais componentes de um diagrama ER, utilizando a notação proposta por Peter Chen.

2.8 Notação para Diagramas ER

Símbolo	Significado
	Entidade
	Entidade fraca
	Relacionamento
	Relacionamento de Identificação
	Atributo
	Atributo-chave
	Atributo Multivalorado
	Atributo composto
	Atributo Derivado
	Participação total de E2 em R
	Razão de cardinalidade de 1:N para E1: E2 em R
	Restrição estrutural (min, max) na participação de E em R

Figura 9. Resumo da notação para diagramas ER proposta por Peter Chen.
Fonte: Adaptado de Elmasri e Navathe (2011, p. 146).

2.9 Entidades

De acordo com Elmasri e Navathe (2011), o objeto básico que o modelo ER representa é uma entidade, uma entidade representa um objeto ou conceito do mundo real que possui uma existência independente, Sudarshan e Silberschatz (2006, p. 135) afirmam ainda que “uma entidade é uma ‘coisa’ ou ‘objeto’ do mundo real que é distingível de todos os outros objetos”. Uma entidade pode ser um objeto com uma existência física, concreta, como uma pessoa, um carro, um funcionário, ou, pode ser um objeto com uma existência conceitual, abstrata, como uma empresa, um cargo, um feriado, ou um curso universitário.

Heuser (2008, p. 21) afirma que uma entidade é “um conjunto de objetos da realidade modelada sobre os quais deseja-se manter informações no banco de dados”.

Por exemplo, cada pessoa em uma empresa é uma entidade, uma entidade possui um conjunto de propriedades e os valores para algum conjunto de propriedades podem identificar de maneira única uma entidade. Assim uma pessoa pode ter uma propriedade `Id_pessoa`, cujo valor identifica unicamente uma determinada pessoa na empresa.

De acordo com Sudarshan e Silberschatz (2006, p. 135), “um conjunto de entidades é um conjunto de entidades do mesmo tipo que compartilham as mesmas propriedades ou atributos”. O conjunto de todas as pessoas que são clientes de uma determinada empresa, por exemplo, pode ser definido como o conjunto de entidades **CLIENTE**. De igual modo, o conjunto de entidades **FUNCIONÁRIO** pode representar o conjunto de todas as pessoas de uma empresa que fazem parte do quadro de colaboradores. As entidades individuais que constituem um conjunto são chamadas de extensão do conjunto de entidades. Assim todos os clientes individuais da empresa são a extensão do conjunto de entidades **CLIENTE**.

Segundo Elmasri e Navathe (2011), um banco de dados em geral contém grupos de entidades que são semelhantes. Por exemplo, uma empresa que emprega centenas de funcionários pode querer armazenar informações semelhantes com relação a cada um dos funcionários. Essas entidades de funcionário compartilham os mesmos atributos, mas cada uma tem os próprios valores para cada atributo. Normalmente refere-se ao conjunto de entidades para usar o mesmo nome do tipo de entidade. Por exemplo, **FUNCIONARIO** refere-se ao tipo de

entidade e também ao conjunto atual de todas as entidades de funcionário no banco de dados.

A figura 10 mostra dois tipos de entidade: FUNCIONARIO E EMPRESA, e uma lista de alguns dos atributos para cada uma. Algumas entidades individuais de cada tipo também são ilustradas, junto com os valores de seus atributos.

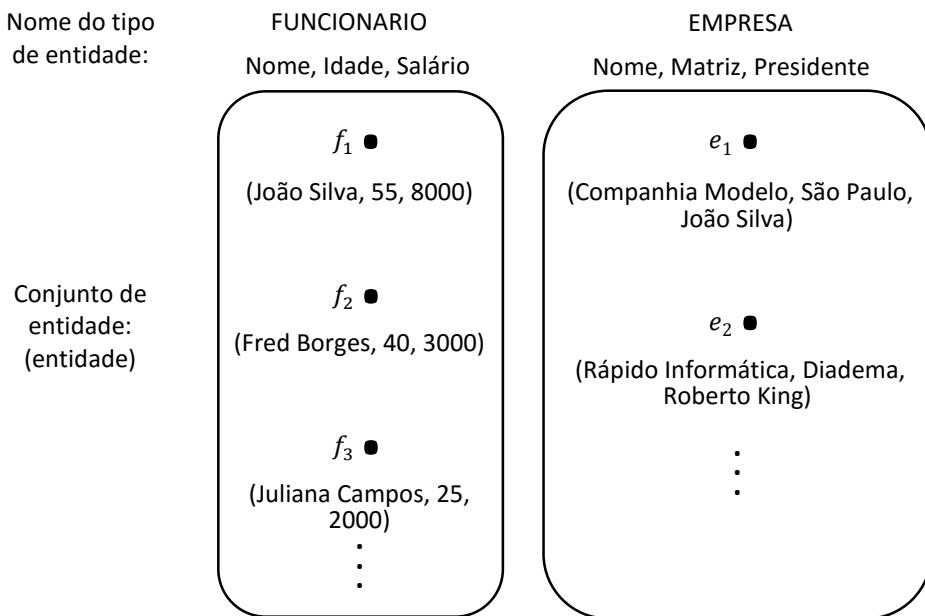


Figura 10. Dois tipos de entidade e algumas entidades membro de cada uma
Fonte: Elmasri e Navathe (2011, p. 137).

2.10 Atributos

Por definição, um conjunto de atributos representa uma Entidade. Conforme Sudarshan e Silberschatz (2006, p. 135), “os atributos são propriedades descritivas possuídas por membros de um conjunto de entidades”. Considere como exemplo, que os possíveis atributos de um conjunto de entidades CLIENTE são Id_cliente, Nome_cliente, Rua_cliente e Cidade_cliente.

Conforme Elmasri e Navathe (2011), os valores de atributo que descrevem cada conjunto de entidades tornam-se uma parte importante dos dados armazenados no banco de dados. Um conjunto de entidades FUNCIONÁRIO, por exemplo, pode ser descrita pelos atributos Nome, Idade, Endereço, Salário e Cargo do funcionário. Obviamente que são exemplos, na vida real, pode haver muito mais atributos.

Cada entidade possui um valor para cada um de seus atributos. Por exemplo, uma determinada entidade do conjunto de entidades CLIENTE pode ter o valor '97789' para Id_cliente, o valor 'Jose Siqueira' para Nome_cliente, o valor 'Rua das Tulipas, 800...' para Rua_cliente e o valor 'Rio de Janeiro' para Cidade_cliente.

Conforme cita Sudarshan e Silberschatz (2006), uma restrição importante das entidades de um tipo de entidade é a chave ou restrição de exclusividade sobre os atributos. Um tipo de entidade normalmente tem um ou mais atributos cujos valores são distintos para cada entidade individual no conjunto de entidades. Esse atributo é denominado atributo-chave, e seus valores podem ser usados para identificar cada entidade de maneira exclusiva. Em nosso exemplo da entidade CLIENTE, o atributo Id_cliente é usado para identificar clientes de maneira única, já que pode haver mais de um cliente com o mesmo nome, rua e cidade. Em geral, a empresa precisa criar e atribuir um identificador único para cada cliente, mas em alguns casos faz-se necessária a designação de atributos compostos como atributo chave.

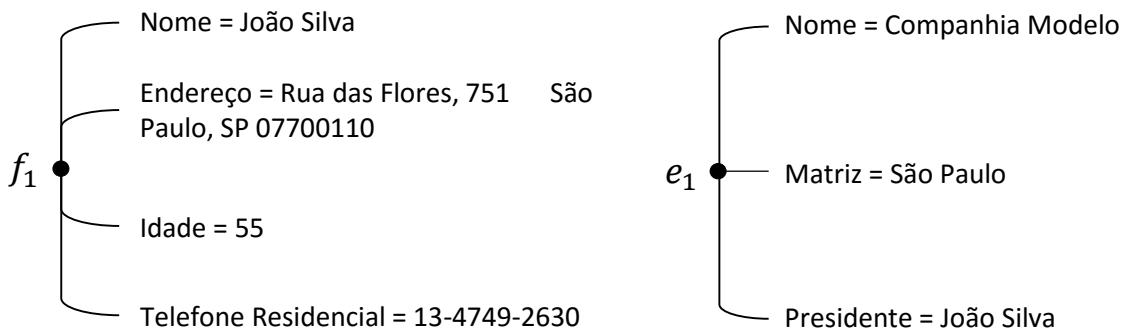


Figura 11. Duas entidades e seus atributos
Fonte: Elmasri e Navathe (2011, p. 135).

2.11 Tipos de atributos

Como visto anteriormente, uma entidade é um objeto que existe no mundo real e é distinguível dos outros objetos. Expressamos a distinção associando a cada entidade um conjunto de características, chamadas de atributos. Alguns atributos representam valores únicos que podem ser usados para identificar cada Entidade de maneira exclusiva, estes atributos no modelo conceitual são chamados de Atributos Chave. Os atributos referenciais geralmente estão ligados ao atributo chave da outra entidade.

Por exemplo, a entidade CLIENTE tem como atributo chave o Id_cliente, assim, a entidade VENDA também possui um campo “Id_cliente” que se relaciona com o campo Id_cliente da entidade CLIENTE. No modelo lógico estes dois tipos de atributos serão respectivamente chamados de chave primária, para o atributo Id_cliente da entidade CLIENTE e chave estrangeira, para o atributo Id_cliente da entidade VENDA.

No modelo ER ocorrem vários tipos de atributos, simples, compostos, atributos de Valor único, atributos multivvalorados, armazenados e derivados. Adiante veremos mais sobre cada um deles.

2.11.1 Atributos compostos versus Atributos simples (atômicos)

De acordo com Elmasri e Navathe (2011), atributos compostos podem ser divididos em subpartes menores, que representam atributos mais básicos, com significados independentes. Por exemplo, o atributo logradouro da entidade FUNCIONARIO, pode ser subdividido em logradouro, cidade, estado e CEP com os valores ‘Rua das Flores, 75’, ‘São Paulo’, ‘SP’ e o ‘CEP 89067555’. Os atributos não divisíveis são chamados atributos simples, também chamados de atributos atômicos, ou seja, não é possível dividi-los em vários campos. Os atributos compostos podem formar uma hierarquia, por exemplo, logradouro pode ser subdividido em três atributos simples: número, rua e número-apartamento, como mostra na figura 12. O valor de um atributo composto é a concatenação dos valores de seus componentes atributos simples.

Atributos compostos são úteis para modelar situações em que um usuário às vezes se refere ao atributo composto como uma unidade, mas outras vezes se refere especificamente a seus componentes. Se o atributo composto for referenciado apenas como um todo, não é necessário subdividi-lo em atributos componentes. Por exemplo, se não for preciso referenciar os componentes individuais de um endereço (CEP, rua, etc), então o endereço inteiro pode ser designado como um atributo simples Elmasri e Navathe (2011).

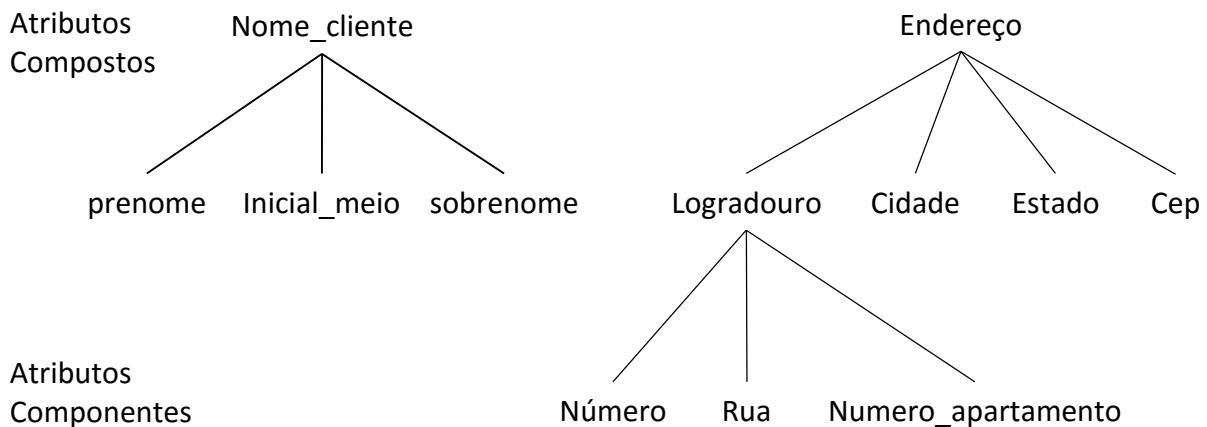


Figura 12. Uma hierarquia de atributos compostos

Fonte: Elmasri e Navathe (2011, p. 136).

2.11.2 Atributos de valor único versus multivvalorados

A maioria dos atributos possui um valor único para uma entidade em particular, tais atributos são chamados de valor único, por exemplo, idade é um atributo de valor único de uma pessoa, assim como seu número de registro no Cadastro de Pessoas Físicas.

Em alguns casos, um atributo pode ter um conjunto de valores para mesma entidade, por exemplo, um atributo cores para um carro ou um atributo formação acadêmica para uma pessoa. Os carros com uma cor têm um único valor, enquanto os carros com duas cores possuem dois valores de cor. De modo semelhante, uma pessoa pode não ter formação acadêmica, outra pessoa pode ter, e uma terceira pessoa pode ter duas ou mais formações, portanto diferentes pessoas podem ter distintos números de valores para o atributo formação acadêmica, esses atributos são chamados de multivvalorados. Note que um atributo multivvalorado pode ter um limite mínimo e um máximo para restringir o número de valores permitidos para cada entidade individual, contudo, pode também não ter um limite. Por exemplo, o atributo cores de um carro pode ser restrito a ter entre 1 e 3 valores se considerarmos que um carro pode ter no máximo três cores, já o atributo formação acadêmica não é restrito a um número máximo de valores possíveis, Elmasri e Navathe (2011).

2.11.3 Atributos armazenados versus derivados

Conforme relata Elmasri e Navathe (2011), em alguns casos, dois ou mais valores de atributos estão relacionados, por exemplo, os atributos Idade e Data_nascimento de uma pessoa.

Para uma entidade de pessoa em particular, o valor de idade pode ser determinado pela data atual (hoje) e o valor da data de nascimento dessa pessoa. O atributo idade, portanto, é chamado de atributo derivado e considerado derivável do atributo Data_nascimento, que é chamado, por sua vez, de atributo armazenado. Alguns valores de atributo podem ser derivados de entidades relacionadas. Por exemplo, um atributo numero_funcionarios de uma entidade DEPARTAMENTO pode ser derivado contando-se o número de funcionários relacionados a esse departamento.

2.11.4 Atributos de Valores Null

Em alguns casos, uma entidade em particular pode não ter um valor aplicado para um atributo. Por exemplo, o atributo número apartamento de um endereço só se aplica a endereços que estão em prédios de apartamento, e não a outros tipos de residências como casas. De modo semelhante, um atributo formação acadêmica só se aplica a pessoas com esse tipo de formação. Para tais situações, foi criado um valor especial, chamado NULL. O endereço de uma casa teria NULL para seu atributo Numero_apartamento e uma pessoa sem formação acadêmica teria NULL para formação acadêmica. NULL também pode ser usado quando não conhecemos o valor de um atributo para determinada entidade, por exemplo, se não soubermos o número do telefone residencial de ‘João Silva’ o significado do primeiro tipo de NULL é “não aplicável” enquanto o significado do segundo é “desconhecido”. A categoria “desconhecido” de NULL pode ser classificada ainda em mais dois casos. O primeiro caso acontece quando se sabe que o valor do atributo existe, mas está faltando, por exemplo, se o atributo altura de uma pessoa for listado como NULL. O segundo caso surge quando não se sabe se o valor do atributo existe, por exemplo, se o atributo telefone residencial de uma pessoa for NULL (Elmasri e Navathe 2011).

2.11.5 Atributos-chave

Elmasri e Navathe (2011, p. 137) afirma que “uma restrição importante das entidades de um tipo de entidade é a chave ou restrição de exclusividade sobre os atributos”.

Um tipo de Entidade normalmente tem um ou mais atributos cujos valores são distintos para cada entidade individual no conjunto de entidades. Esse atributo é denominado atributo-chave, ou atributo determinante, e seus valores podem ser usados para identificar cada Entidade de maneira exclusiva.

Por exemplo, o atributo nome é uma chave do tipo de entidade EMPRESA, pois duas empresas não podem ter o mesmo nome. Para o tipo de entidade PESSOA, um atributo chave típico é o CPF. Às vezes, vários atributos juntos formam uma chave, significado que a combinação dos valores de atributo deve ser distinta para cada entidade. Se um conjunto de atributos possui essa propriedade, o modo correto de representar isso no modelo ER que descrevemos aqui é definir um atributo composto e designa-lo como um atributo chave do tipo de entidade. Observe que essa chave composta precisa ser mínima, ou seja, todos os atributos componentes precisam estar incluídos no atributo composto para ter a propriedade de exclusividade. Atributos supérfluos não devem ser incluídos em uma chave.

Na notação diagramática ER, cada atributo chave tem seu nome sublinhado dentro da Oval, conforme mostrado na figura 13.

Conforme Elmasri e Navathe (2011), especificar que um atributo é uma chave de um tipo de entidade significa que a propriedade anterior da exclusividade precisa ser mantida para cada conjunto de entidades do tipo de entidade. Logo, essa é uma restrição que proíbe que duas entidades tenham o mesmo valor para o atributo-chave ao mesmo tempo. Essa não é a propriedade de um conjunto de entidades em particular, em vez disso, é uma restrição sobre qualquer conjunto de entidades do tipo de entidade em qualquer ponto no tempo. Essa restrição-chave é derivada das restrições do minimundo que o banco de dados representa.

Alguns tipos de entidades possuem mais de um atributo chave, por exemplo, cada um dos atributos cod_veiculo e Registro do tipo de entidade CARRO, (figura 13), é uma chave por si só. O atributo Registro é um exemplo de uma chave composta formada por dois atributos componentes simples, Estado e número, nenhum deles sendo uma chave por si só. Um tipo de entidade também pode não

ter chave, nesse caso ela é chamada de tipo de entidade fraca. Em nossa notação diagramática, se dois atributos forem sublinhados separadamente, então cada um é uma chave por si só Elmasri e Navathe (2011).

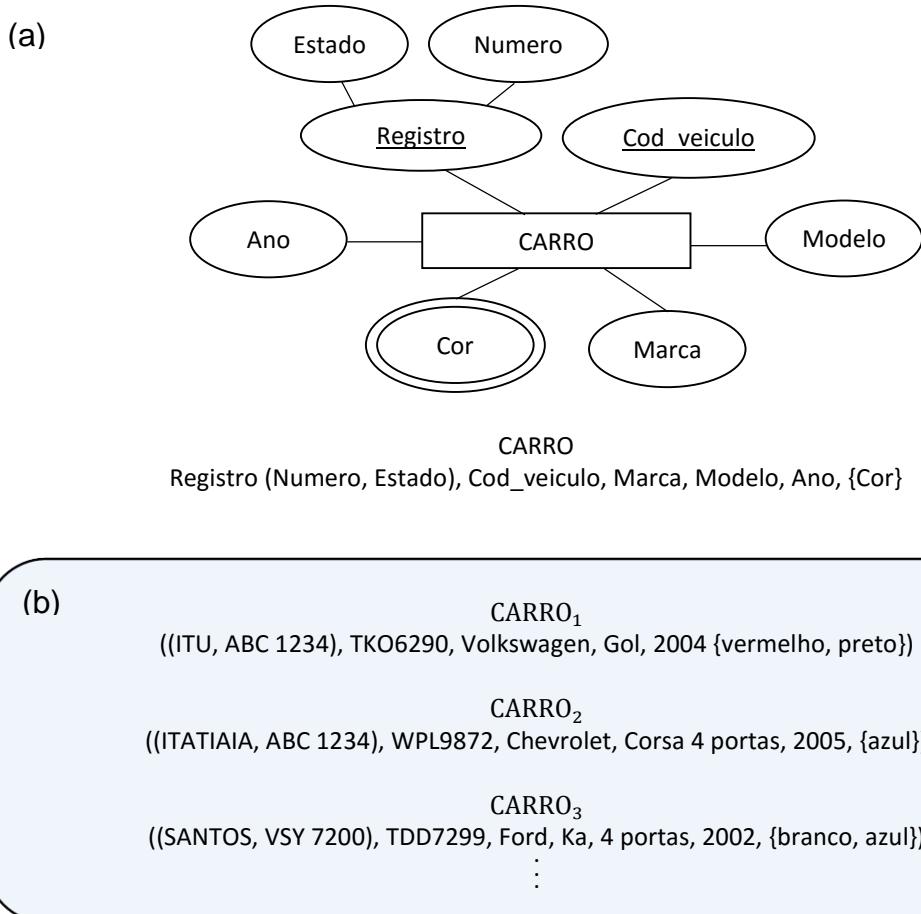


Figura 13. Uma entidade com dois atributos-chave
Fonte: Adaptado de Elmasri e Navathe (2011, p. 138).

A figura 13 mostra o tipo de entidade CARRO com dois atributos-chave, Registro e Cod_veiculo, onde (a) é a notação do diagrama ER e (b) o conjunto de entidade com três entidades.

Conforme Elmasri e Navathe (2011), um tipo de entidade é representado nos diagramas ER como uma caixa retangular delimitando seu nome. Os nomes de atributo são delimitados em ovais, sendo ligados a seu tipo de entidades por linhas retas, os atributos compostos são ligados a seus atributos componentes por linhas retas e os atributos multivvalorados aparecem em ovais duplas.

2.12 Entidades Fortes e Entidades Fracas

De acordo com Elmasri e Navathe (2011), os tipos de entidade que possuem um Atributo-chave são chamados tipos de entidade fortes, são entidades cuja existência independe de outras entidades, por si só elas já possuem total sentido de existir. Por exemplo, no sistema de uma empresa, a entidade FUNCIONARIO independe de qualquer outra entidade para existir. Tipos de entidade que não possuem atributos-chave próprios são chamados tipos de entidade fraca. As entidades pertencentes a um tipo de entidade fraca são identificadas por estarem relacionadas a entidades específicas de outro tipo em combinação com seus valores de atributo. Chamamos esse outro tipo de entidade de tipo de entidade de identificação ou proprietário, e chamamos o tipo de relacionamento que relaciona um tipo de entidade fraca a seu proprietário de relacionamento de Identificação do tipo de entidade fraca. Um tipo de entidade fraca sempre tem uma restrição de participação total (dependência de existência) com relação a seu relacionamento de identificação, porque a entidade fraca não pode ser identificada sem uma entidade proprietária.

Podemos dizer que entidades fracas são aquelas que dependem de outras entidades para existirem, pois individualmente elas não fazem sentido, por exemplo, a entidade Venda depende da entidade produto, pois uma venda sem itens vendidos não tem sentido.

Nem toda dependência de existência resulta em um tipo de entidade fraca. Por exemplo, uma entidade Carteira_motorista não pode existir a menos que esteja relacionada a uma entidade pessoa, embora tenha a própria chave (numero_habilitação) e, portanto, não seja uma entidade fraca. Por exemplo, considere o tipo de entidade DEPENDENTE, relacionado à FUNCIONARIO, que é usado para registrar os dependentes de cada funcionário por meio de um relacionamento um para muitos. Em nosso exemplo, os atributos de dependente são Nome, data Nascimento, sexo e parentesco (Com o funcionário). Dois dependentes de dois funcionários distintos podem, por coincidência, ter os mesmos valores para Nome, Data Nascimento, Sexo e Parentesco, mas ainda assim, eles são entidades distintas. Eles são identificados como entidades distintas apenas depois de determinar a entidade de funcionário em particular a qual cada dependente está

relacionado. Considera-se que cada entidade de funcionário possui as entidades dependentes que estão relacionadas a ele.

De acordo com Elmasri e Navathe (2011), um tipo de entidade fraca normalmente tem uma chave parcial, que é o atributo que pode identificar exclusivamente as entidades fracas que estão relacionadas à mesma entidade proprietária. Em nosso exemplo, se considerarmos que dois dependentes do mesmo funcionário não poderão ter o mesmo nome, o atributo “Nome” de DEPENDENTE é a chave parcial. No pior dos casos, um atributo composto de todos os atributos da entidade fraca será a chave parcial.

2.13 Relacionamentos e conjuntos de relacionamentos

Conforme relata Heuser (2008, p. 22):

Além de especificar os objetos sobre os quais se deseja manter informações, o Diagrama de entidade e relacionamento deve permitir a especificação das propriedades dos objetos que serão armazenadas no Banco de Dados. Uma das propriedades sobre as quais pode ser desejável manter informações é a associação entre os objetos. Exemplificando, pode ser desejável saber quais pessoas estão associadas a quais departamentos em uma organização.

Nos diagramas ER, os tipos e nomes de relacionamentos são exibidos como caixas em forma de losango, que são conectadas por linhas retas às caixas retangulares que representam os tipos de entidades participantes do relacionamento (Heuser 2008).

A Figura 14 apresenta um DER contendo duas entidades, FUNCIONARIO e DEPARTAMENTO, e um relacionamento, TRABALHA_PARA.



Figura 14. Representação de um relacionamento entre duas entidades
Fonte: Adaptado de Heuser (2008).

Um relacionamento é uma associação entre várias entidades. Um tipo de relacionamento entre tipos de entidade define um conjunto de associações ou um conjunto de relacionamento entre as entidades desses tipos de entidade. Um tipo de relacionamento e seu conjunto de relacionamento correspondente em geral são referenciados pelo mesmo nome, por exemplo, um funcionário trabalha para um departamento, logo o tipo de relacionamento é TRABALHA_PARA, o conjunto de relacionamentos entre todos os funcionários que trabalham para todos os departamentos também se chamará TRABALHA_PARA, Elmasri e Navathe (2011).

Conforme relata Elmasri e Navathe (2011) matematicamente, o conjunto de relacionamento R é um conjunto de instâncias de relacionamento r_i , onde cada instância de relacionamento r_i associa-se a n entidades individuais ($e_1, e_2, e_3, \dots, e_n$), e cada uma destas entidades individuais e_i em r_i é um membro do conjunto de entidades E_i , logo, um conjunto de relacionamento é uma relação matemática sobre um conjunto de entidades (E_1, E_2, \dots, E_n).

Cita ainda Elmasri e Navathe (2011), que, informalmente cada instância de relacionamento r_i em R é uma associação de entidades, onde a associação inclui exatamente uma entidade de cada tipo de entidade participante. Cada instância de relacionamento r_i desse tipo representa o fato de que as entidades participantes em r_i estão relacionadas de alguma maneira na situação do minimundo correspondente.

Por exemplo, considere um tipo de relacionamento TRABALHA_PARA entre os dois tipos de entidade FUNCIONARIO e DEPARTAMENTO, que associa cada funcionário ao departamento para o qual o funcionário trabalha. Cada instância de relacionamento no conjunto de relacionamentos TRABALHA_PARA associa uma entidade FUNCIONARIO a uma entidade DEPARTAMENTO.

A figura 15 ilustra esse exemplo, onde cada instância de relacionamento r_i aparece conectada às entidades FUNCIONARIO E DEPARTAMENTO que participam em r_i .

Observe que os funcionários f_1, f_3 e f_6 trabalham para o departamento d_1 , os funcionários f_2 e f_4 trabalham para o departamento d_2 e os funcionários f_5 e f_7 trabalham para o departamento d_3 .

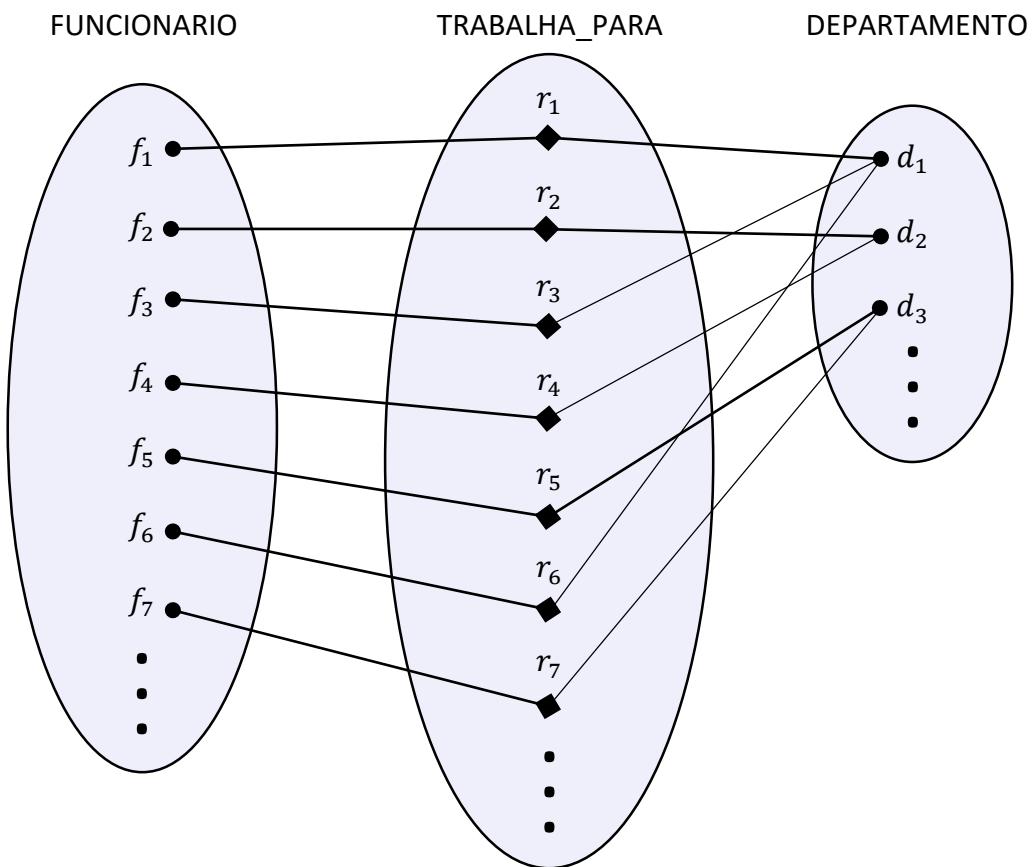


Figura 15. Um tipo de relacionamento entre duas Entidades
Fonte: Elmasri e Navathe (2011, p. 140).

2.13.1 Grau de um tipo de relacionamento

Segundo Elmasri e Navathe (2011), o grau de um tipo de relacionamento é o número dos tipos de entidades participantes. O relacionamento TRABALHA_PARA, mostrado no exemplo anterior, tem grau dois. Um tipo de relacionamento de grau dois é chamado de binário e um tipo de grau três é chamado de ternário. Os relacionamentos geralmente podem ser de qualquer grau, mas os mais comuns são os relacionamentos binários.

Relacionamentos de grau mais alto geralmente são mais complexos do que os binários. A figura 16 ilustra um exemplo de relacionamento ternário através do relacionamento FORNECE, em que cada instância de relacionamento r_i associa três entidades, um fornecedor f , uma peça p , e um projeto j , onde que f fornece a peça p ao projeto j .

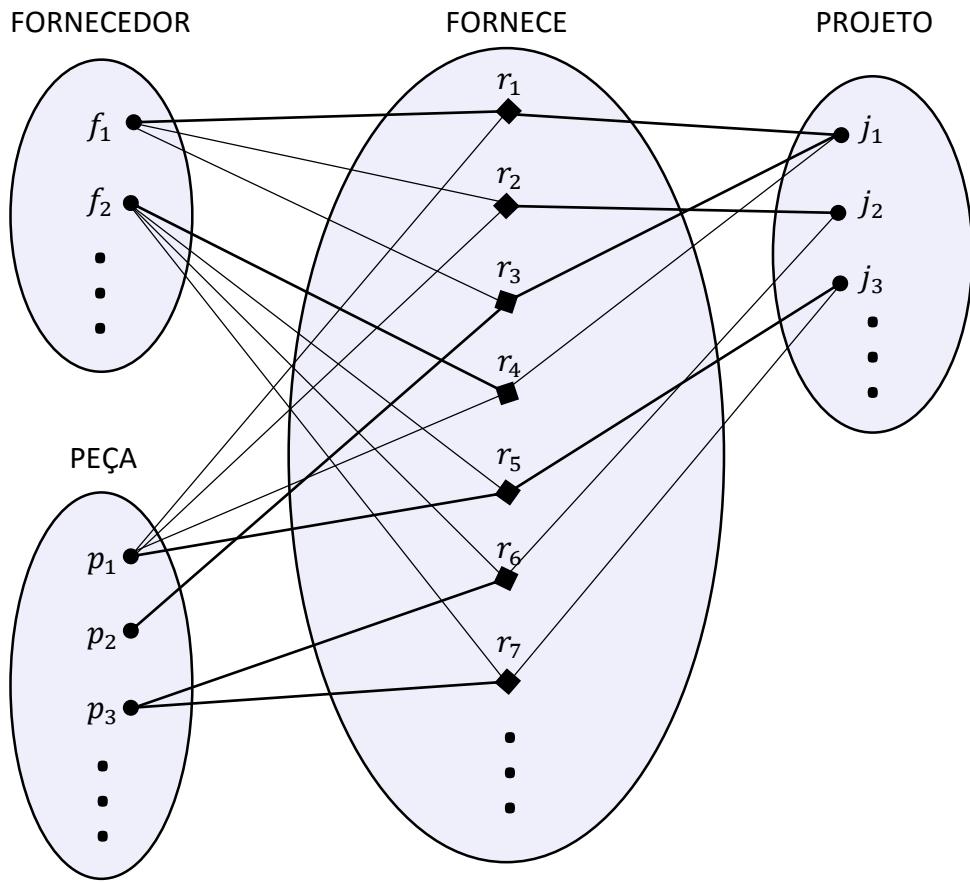


Figura 16. Conjunto de relacionamento ternário
Fonte: Elmasri e Navathe (2011, p. 141).

2.13.2 Restrições sobre tipos de relacionamentos binários

Segundo Elmasri e Navathe (2011), os tipos de relacionamento costumam ter certas restrições, que limitam as combinações de entidades que podem participar no conjunto de relacionamento correspondente. Essas restrições são determinadas com base na situação do minimundo que os relacionamentos representam.

Por exemplo, na figura 14, se a empresa tem uma regra a qual obriga que cada funcionário precisa trabalhar para exatamente um departamento, então é necessário descrever essa restrição no esquema.

Podemos distinguir dois tipos principais de restrições em relacionamentos binários: razão de cardinalidade, também chamada de cardinalidade máxima, e restrição de participação ou cardinalidade mínima.

2.13.3 Razão de cardinalidade (Cardinalidade máxima)

De acordo com Sudarshan e Silberschatz (2006) os fatores de cardinalidade expressam o número de entidades ao qual outra entidade pode estar associada por um conjunto de relacionamento. Embora possa contribuir para a descrição dos conjuntos de relacionamento que envolve mais de dois conjuntos de entidades, o mapeamento de cardinalidade é útil principalmente em descrever conjuntos de relacionamentos binários.

A razão de cardinalidade para um relacionamento binário especifica o número máximo de instâncias de relacionamento em que uma entidade pode participar, obviamente que a cardinalidade apropriada para um determinado conjunto de relacionamento depende da situação real que está sendo modelada, desta forma, a razão de cardinalidade, ou cardinalidade máxima, indica o número máximo de ocorrências de uma entidade associada à outra ocorrência de outra entidade por meio de um relacionamento. É representada por “1” (uma ocorrência) ou N (Várias ocorrências) Sudarshan e Silberschatz (2006).

Por exemplo, no tipo de relacionamento binário TRABALHA_PARA, FUNCIONARIO:DEPARTAMENTO tem razão de cardinalidade N:1. Isso significa que, para esse relacionamento TRABALHA_PARA em particular, um funcionário pode estar relacionado no máximo a um único departamento. Por sua vez, uma entidade de departamento em particular pode estar relacionada a qualquer número de funcionários (N indica que não existe um número máximo).

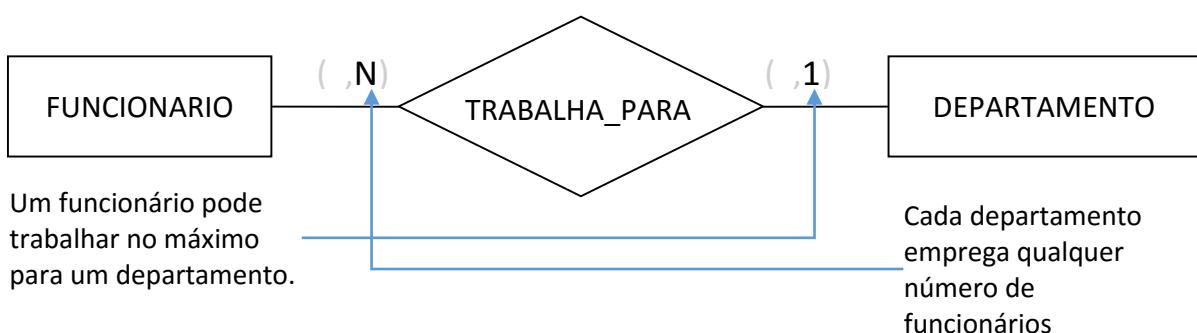
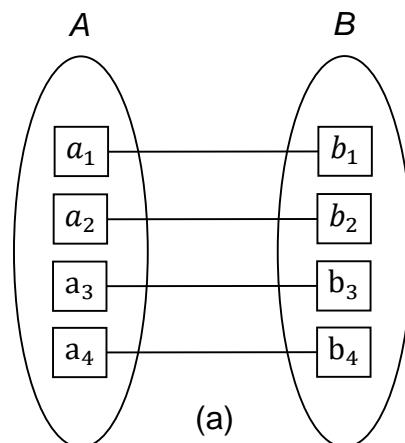


Figura 17. Cardinalidade máxima entre duas Entidades
Fonte: Adaptado Heuser (2008, p. 25)

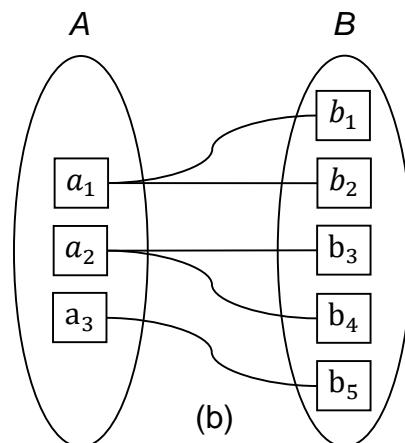
Conforme Elmasri e Navathe (2011), as razões de cardinalidade possíveis para relacionamentos binários são: 1:1, 1:N, N:1 e N:N.

Como visto anteriormente, um tipo de relacionamento de grau dois é aquele que envolve duas entidades, também chamado de binário, para exemplificar o entendimento sobre cardinalidade máxima, considere um conjunto de relacionamento binário entre conjuntos de entidades A e B onde as razões de cardinalidade, conforme descritas por Sudarshan e Silberschatz (2006, p. 140) são as seguintes:

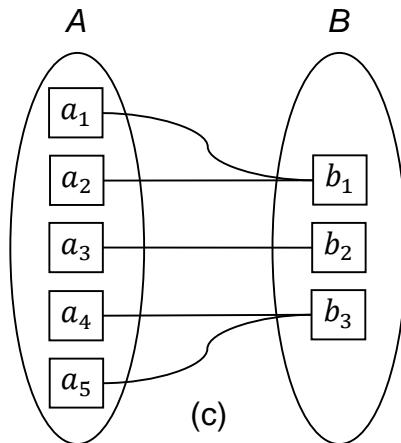
1:1 - Um para um. Uma entidade em A é associada a no máximo uma entidade em B, e uma entidade em B é associada a no máximo uma entidade em A.



1:N – Um para muitos. Uma entidade em A é associada a qualquer número de entidades (zero ou mais) em B. Entretanto, uma entidade em B pode ser associada a no máximo uma entidade em A.



N:1 – Muitos para um. Uma entidade em A é associada a no máximo uma entidade em B. Entretanto, uma entidade em B pode ser associada a qualquer número de entidades (zero ou mais) em A.



N:N – Muitos para Muitos. Uma entidade em A é associada a qualquer número de entidades (zero ou mais) em B, e uma entidade em B pode ser associada a qualquer número de entidades (zero ou mais) em A.

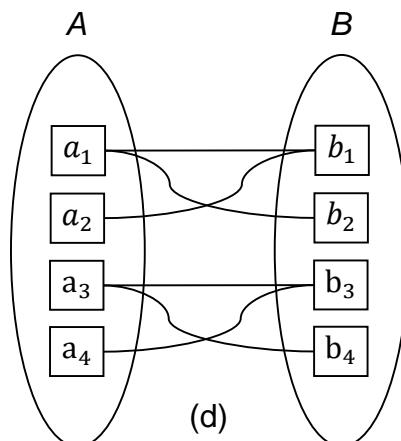


Figura 18. Cardinalidade de mapeamento (a) 1:1, (b) 1:N, (c) N:1 e (d) N:N
Fonte: Sudarshan e Silberschatz (2006, p. 140).

2.13.4 Restrição de participação (Cardinalidade mínima)

Conforme afirma Sudarshan e Silberschatz (2006, p. 143):

A restrição de participação especifica se a existência de uma entidade depende de ela estar relacionada a outra entidade por meio do tipo de relacionamento. Essa restrição especifica o número mínimo de instâncias de relacionamento em que cada entidade pode participar, e às vezes é chamada de restrição de cardinalidade mínima. Existem dois tipos de restrições de participação, total e parcial.

Considere um conjunto de entidades E_i , e um conjunto de relacionamentos R . A participação de um conjunto de entidades E_i em um conjunto de relacionamentos R é chamada de total se todas as entidades em E_i participam em pelo menos um relacionamento em R . Se apenas algumas entidades em E_i participam em

relacionamentos em R , a participação do conjunto de entidades E_i no relacionamento R é chamada de parcial (Sudarshan e Silberschatz 2006).

Conforme Elmasri e Navathe (2011), em diagramas ER, a participação total, ou dependência de existência, também pode ser exibida como uma linha dupla que conecta o tipo de entidade participante ao relacionamento, enquanto a participação parcial é representada por uma linha simples.

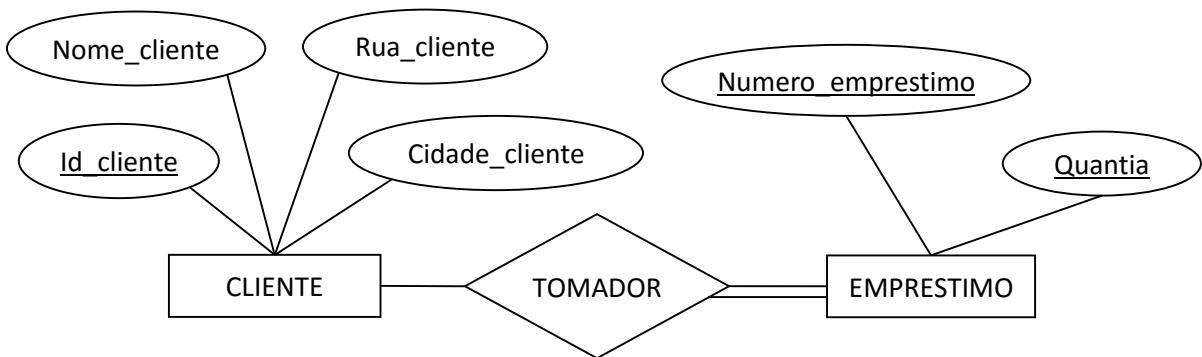


Figura 19. Participação total em um conjunto de relacionamento
Fonte: Sudarshan e Silberschatz (2006, p. 146).

Observe a figura 19, neste exemplo esperamos que toda entidade EMPRESTIMO esteja relacionada a um cliente por meio do relacionamento TOMADOR. Portanto, a participação de EMPRESTIMO no conjunto de relacionamento TOMADOR é total. Esperamos que todo empréstimo esteja relacionado a um cliente, caso contrário este banco teria um problema dos grandes.

Por outro lado, uma pessoa pode ser um cliente do banco quer ela tenha ou não um empréstimo com este banco. Consequentemente, é possível que apenas algumas das entidades cliente estejam relacionadas ao conjunto de entidades EMPRESTIMO por meio do relacionamento TOMADOR e a participação de cliente no conjunto de relacionamento TOMADOR portanto é parcial (Sudarshan e Silberschatz 2006).

Observe que, nessa notação, podemos especificar nenhum mínimo (0) (participação parcial) ou um mínimo de um (1), (Participação total), em outras palavras, a restrição de cardinalidade mínima indica o número mínimo de ocorrências de uma entidade associada à outra ocorrência de outra entidade relacionada (Sudarshan e Silberschatz 2006).

Conforme Elmasri e Navathe (2011) uma das maneiras de ser mostrada em um diagrama ER é através da notação (Min, Max), onde no lado esquerdo (Min),

pode ser representada por “0”, indicando a participação parcial, logo não há obrigatoriedade de correspondente na outra entidade, ou 1, quando a associação é obrigatória, indicando participação total, onde pelo menos um correspondente na outra entidade deve existir.

Em outro exemplo, conforme mostra a Figura 21, se a política de uma empresa afirma que todo funcionário precisa trabalhar para um departamento, então uma entidade de funcionário só pode existir se participar em, pelo menos, uma instância de relacionamento TRABALHA_PARA. Assim, a participação de FUNCIONARIO em TRABALHA_PARA é chamada de participação total, significando que cada entidade no conjunto total de entidades de funcionário deve estar relacionada no mínimo a uma entidade de departamento por meio de TRABALHA_PARA. A participação total também é conhecida como dependência de existência (Elmasri e Navathe, 2011).

Na figura 20 não esperamos que cada funcionário gerencie um departamento, de modo que a participação de FUNCIONARIO no tipo de relacionamento GERENCIA é parcial, significando que uma parte do conjunto de entidades de funcionário está relacionada a alguma entidade de departamento por meio de GERENCIA, mas não necessariamente todas, (Elmasri e Navathe, 2011).

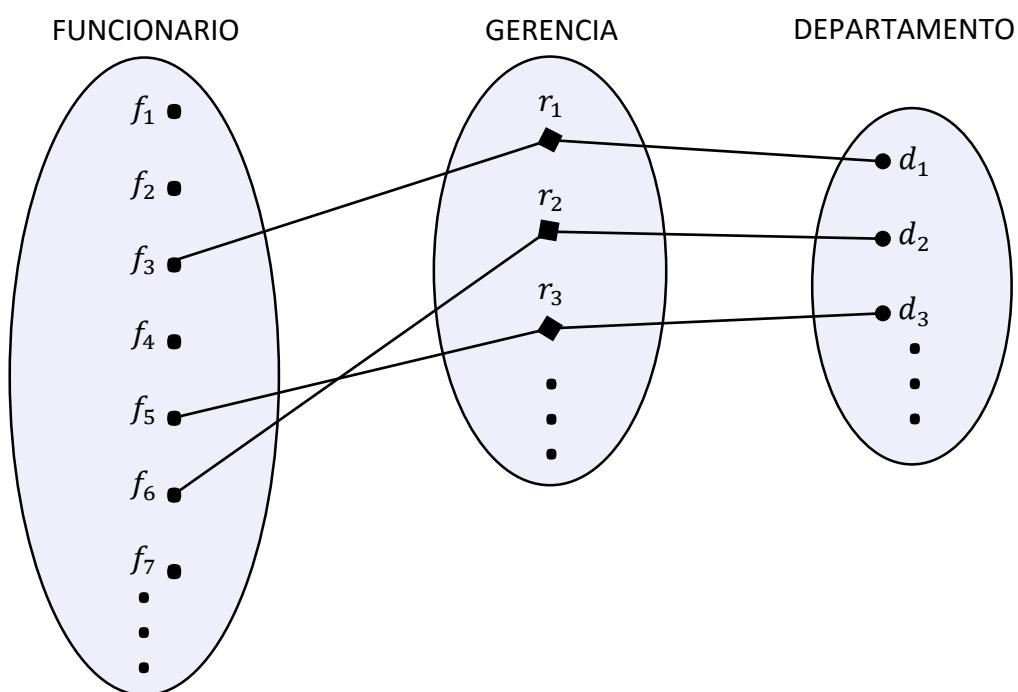


Figura 20. Participação parcial em um conjunto de relacionamento
Fonte: Elmasri e Navathe (2011, p. 143).

Na figura 21, no tipo de relacionamento binário TRABALHA_PARA, FUNCIONARIO:DEPARTAMENTO tem cardinalidade mínima 0:1. Isso significa que, para esse relacionamento TRABALHA_PARA em particular, um funcionário deve estar relacionado no mínimo a um departamento. Por sua vez, uma entidade de departamento em particular pode não estar relacionada a um funcionário.



Figura 21. Cardinalidade mínima entre duas entidades
Fonte: Adaptado Heuser (2008, p. 25).

Observe que apesar de sabermos que os departamentos em uma empresa existem para serem responsáveis por funcionários, especificamos a cardinalidade mínima de DEPARTAMENTO em FUNCIONARIO como sendo “0”. Com isso admitimos a possibilidade de existirem departamentos vazios. Esta cardinalidade foi especificada considerando o estado do banco de dados imediatamente após a criação de um novo departamento, bem como o estado imediatamente antes da eliminação de um departamento. Da forma como a restrição foi especificada, é possível incluir o departamento em uma transação, para, depois, em transações subsequentes, vinculá-lo aos funcionários sob sua responsabilidade. Se tivesse sido especificada a cardinalidade mínima de “1”, pelo menos 1 funcionário teria que ser vinculado ao departamento já na própria transação de inclusão do departamento.

Conforme cita Heuser (2008), para especificar a cardinalidade mínima é necessário possuir conhecimento sobre as transações de inclusão e exclusão das entidades.

A figura 22 mostra o esquema completo de relacionamento FUNCIONARIO:DEPARTAMENTO usando a notação (Min, Max). Observe que a leitura da notação é sempre utilizando a que vai anotada no outro lado do relacionamento, por exemplo, a cardinalidade mínima e máxima da entidade FUNCIONARIO é aquela que está anotada ao lado da entidade DEPARTAMENTO.

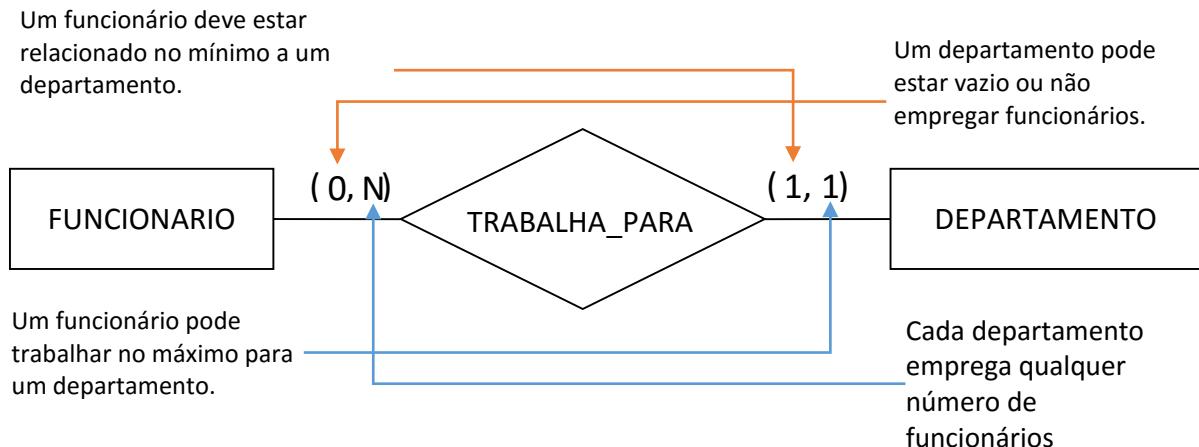


Figura 22. Relacionamento entre duas entidades na notação (Min, Max)
Fonte: Adaptado Heuser (2008, p. 25).

2.14 Notações alternativas para Diagramas ER

2.14.1 Unified Modeling Language (UML)

De acordo com Elmasri e Navathe (2011), existem muitas notações diagramáticas alternativas para exibir diagramas ER, uma delas é a notação Unified Modeling Language para diagramas de classe, que foi proposta como um padrão para a modelagem conceitual de objeto.

Conforme relata Sudarshan e Silberschatz (2006, p. 168) “a Unified Modeling Language (UML) é um padrão desenvolvido sob o patrocínio do Object Management Group (OMG) para criar especificações de vários componentes de um sistema de software, uma dessas especificações é o Diagrama de classe”.

Segundo Elmasri e Navathe (2011), um diagrama de classe é semelhante a um diagrama ER. Os tipos de entidade são modelados como classes. Uma entidade em ER corresponde a um objeto em UML. Nos diagramas de classe UML, uma classe é exibida como uma caixa que inclui três seções: a seção superior mostra o nome da classe (semelhante ao nome do tipo de entidade); a seção do meio inclui os atributos; e a última seção inclui as operações que podem ser aplicadas aos objetos individuais da classe, (semelhante às entidades individuais em um conjunto de entidades).

A UML mostra conjuntos de entidades como retângulos e, diferente do diagrama ER, mostra os atributos dentro do retângulo e não como elipses

separadas. Na verdade, a UML modela objetos, enquanto o ER modela entidades. Os objetos são como entidades e possuem atributos, mas, além disso, fornecem um conjunto de funções, também conhecidas por métodos, que podem ser chamadas para calcular valores na base dos atributos dos objetos, ou para atualizar o próprio objeto. Os diagramas de classe podem descrever métodos e atributos, Sudarshan e Silberschatz (2006).

Em terminologia UML, os tipos de relacionamento são chamados de associações, e as instâncias de relacionamento são chamadas de ligações. Uma associação binária (tipo de relacionamento binário) é representada como uma linha que conecta as classes participantes (tipos de entidade) e pode, de maneira opcional, ter um nome. Um atributo de relacionamento, chamado atributo de ligação, é colocado em uma caixa que está conectada à linha da associação por uma linha tracejada. A notação (Min, Max) é usada para especificar restrições de relacionamento, que em terminologia UML são chamadas multiplicidades. As multiplicidades são especificadas na forma min..max, e um asterisco (*) indica nenhum limite máximo na participação. Contudo, as multiplicidades são colocadas nos lados opostos do relacionamento quando comparadas à notação ER, (Elmasri e Navathe 2011, p. 149).

De acordo com Sudarshan e Silberschatz (2006), os diagramas de classe UML incluem várias outras notações que não correspondem às notações ER. Por exemplo, uma linha entre dois conjuntos de entidades com um losango em uma ponta, especifica que a entidade no lado do losango contém a outra entidade, essa representação de relacionamento entre um objeto inteiro e suas partes componentes é chamada de “agregação”. Por exemplo, uma entidade veículo pode conter uma entidade motor.

A figura 23 mostra o esquema conceitual de uma Empresa na notação do diagrama de classes UML.

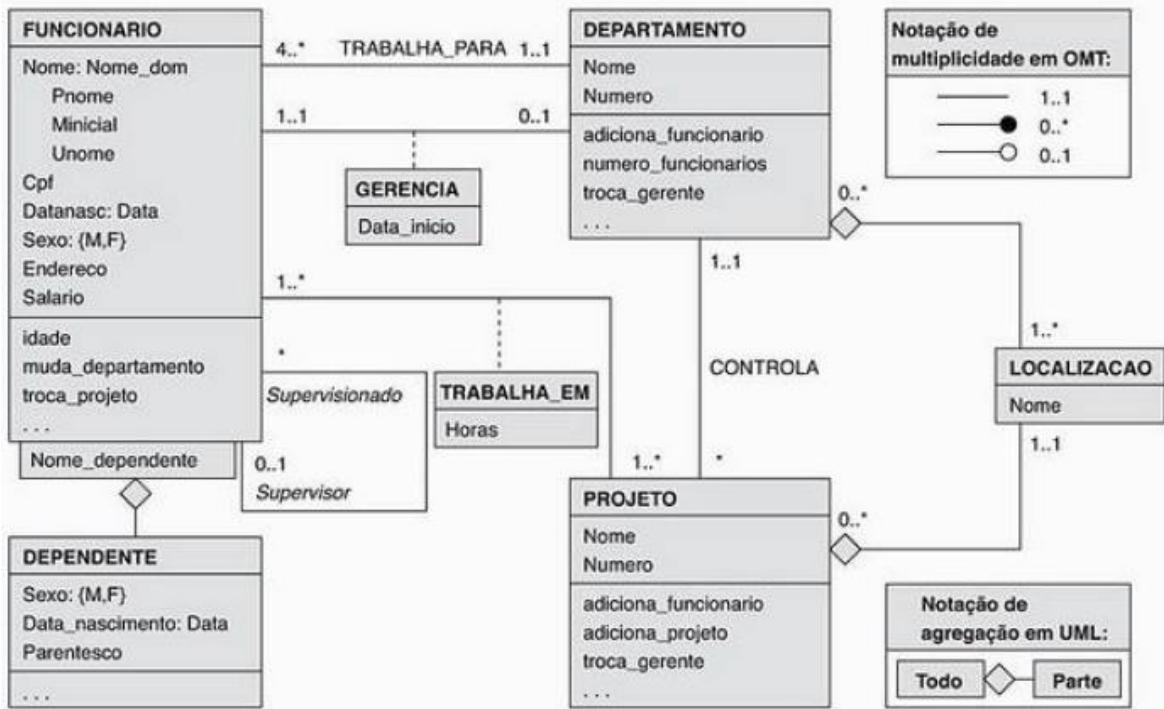


Figura 23. Notação em diagrama de classes UML

Fonte: Elmasri e Navathe (2011, p. 150).

Na figura 24 há uma comparativo entre os simbolos usados em um diagrama entidade relacionamento e um diagrama UML.

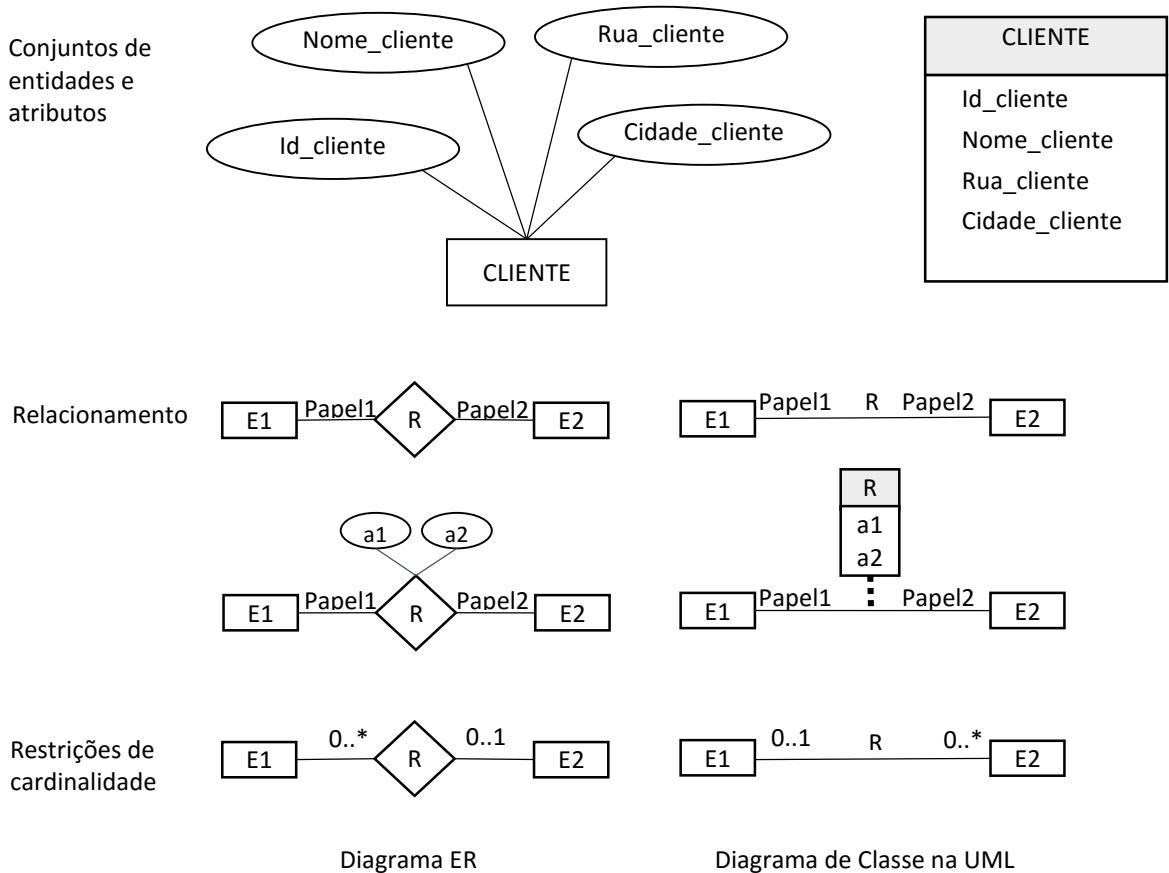


Figura 24. Símbolos usados na notação do diagrama de classe UML
Fonte: Adaptador de Sudarshan e Silberschatz (2006, p. 169).

2.15 Modelo de Banco de Dados Relacional

Segundo Elmasri e Navathe (2011), o modelo de banco de dados relacional foi introduzido inicialmente por um pesquisador da IBM chamado Dr. Edgar Frank Codd, que em 1970 publicou o artigo “A Relational Model of Data for Large Shared Data Banks”, (Modelo de dados relacional para grandes bancos de dados compartilhados) e atraiu atenção imediata devido a sua simplicidade e base matemática.

De acordo com Powell (2006), as primeiras implementações comerciais do modelo relacional se tornaram disponíveis no início da década de 1980, com o sistema SQL/DS no sistema operacional MVS, da IBM, e o SGBD, da Oracle. Desde então, o modelo foi implantado em uma grande quantidade de sistemas comerciais.

Conforme Sudarshan e Silberschatz (2006), o modelo relacional é hoje o principal modelo de dados para aplicações comerciais de processamento de dados,

é atualmente a melhor solução para armazenamento e recuperação de dados. Ele conquistou sua posição de destaque devido a sua simplicidade, que facilita o trabalho do programador, isso porque entre outros motivos, não tem caminhos predefinidos para se fazer acesso aos dados, comparado com os modelos de dados anteriores, os modelos hierárquicos e de rede.

De acordo com Elmasri e Navathe (2011), o modelo relacional representa o banco de dados como uma coleção de relações. Na terminologia formal, uma tabela é chamada de relação, uma linha é chamada de tupla e um cabeçalho da coluna é chamado de atributo. O tipo de dado que descreve os tipos de valores que podem aparecer em cada coluna é representado por um domínio de valores possíveis, os tipos de domínio podem ser textuais, numéricos ou de data.

O usuário do sistema de banco de dados pode consultar essas relações, inserir novas tuplas, excluir tuplas e atualizar (modificar) tuplas, estas operações são possíveis utilizando a Structured Query Language (SQL), linguagem utilizada para interagir com a maioria dos bancos de dados relacionais e que é baseada em uma derivação da teoria dos conjuntos, a álgebra relacional.

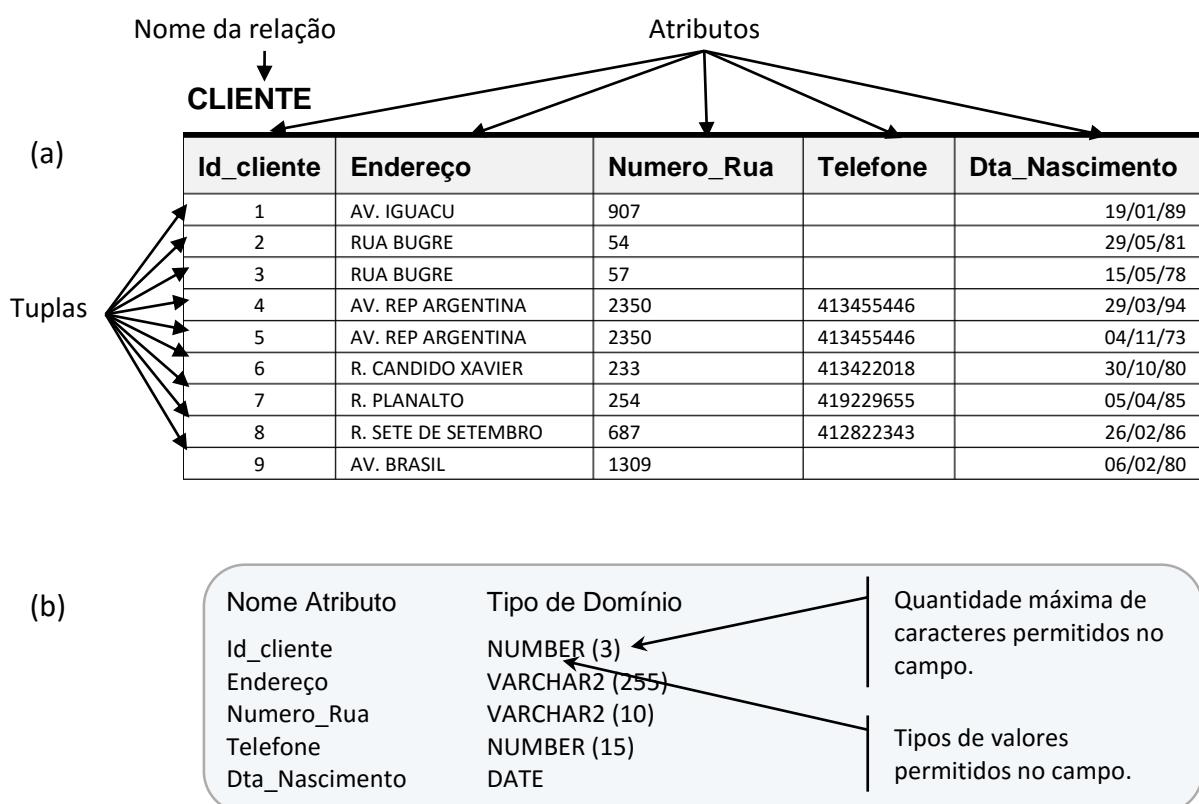


Figura 25. (a) Atributos e tuplas de uma relação, (b) O domínio de seus atributos
Fonte: Elmasri e Navathe (2011, p. 40).

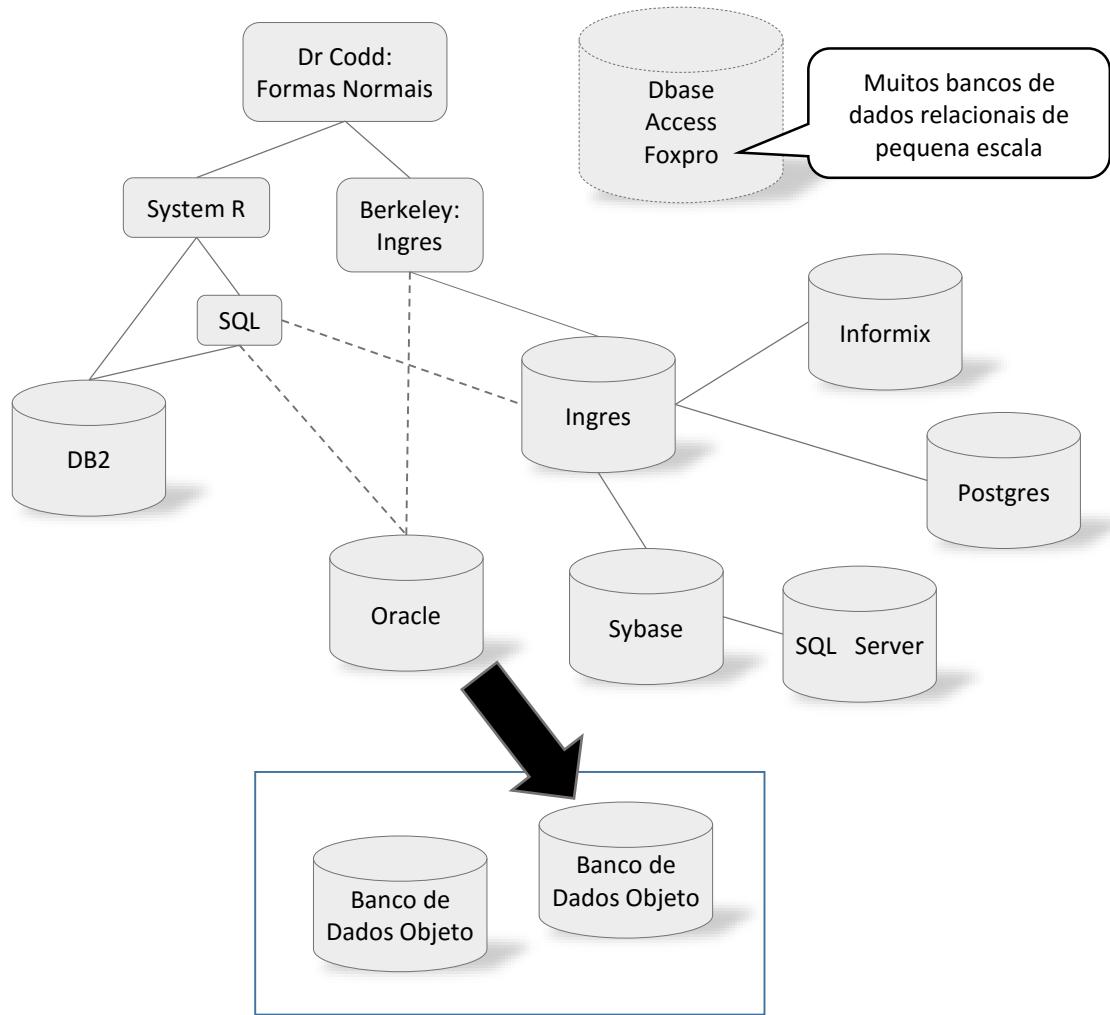


Figura 26. História do modelo de banco de dados relacional
Fonte: Adaptado de Powell (2006, p. 12).

2.16 A Álgebra relacional

De acordo com Elmasri e Navathe (2011), as duas linguagens formais para o modelo relacional são a álgebra e o cálculo relacional, historicamente, foram desenvolvidas antes da linguagem SQL. A álgebra relacional é usada nos detalhes internos de SGBDRs para processamento e otimização de consulta e o cálculo relacional é considerado a base para a linguagem SQL. O resultado de uma recuperação (consulta) é uma nova relação (Tabela), que pode ter sido formada de uma ou mais relações. As operações da álgebra, assim, produzem novas relações, que podem ser manipuladas ainda mais usando operações da mesma álgebra. Uma sequência de operações da álgebra relacional forma uma expressão da álgebra

relacional, cujo resultado também será uma relação que representa o resultado de uma consulta de banco de dados (ou consulta de recuperação).

A álgebra relacional é importante por diversos motivos, primeiro, ela oferece um alicerce formal para as operações do modelo relacional, segundo, e talvez o mais importante, as operações e funções essenciais nos módulos internos na maioria dos sistemas relacionais são baseadas nas operações da álgebra relacional, que é usada como base para a implementação e otimização de consultas nos módulos de otimização e processamento de consulta dos SGBDRs, e terceiro, alguns de seus conceitos são incorporados na linguagem de consulta SQL para os SGBDRs (Elmasri e Navathe 2011).

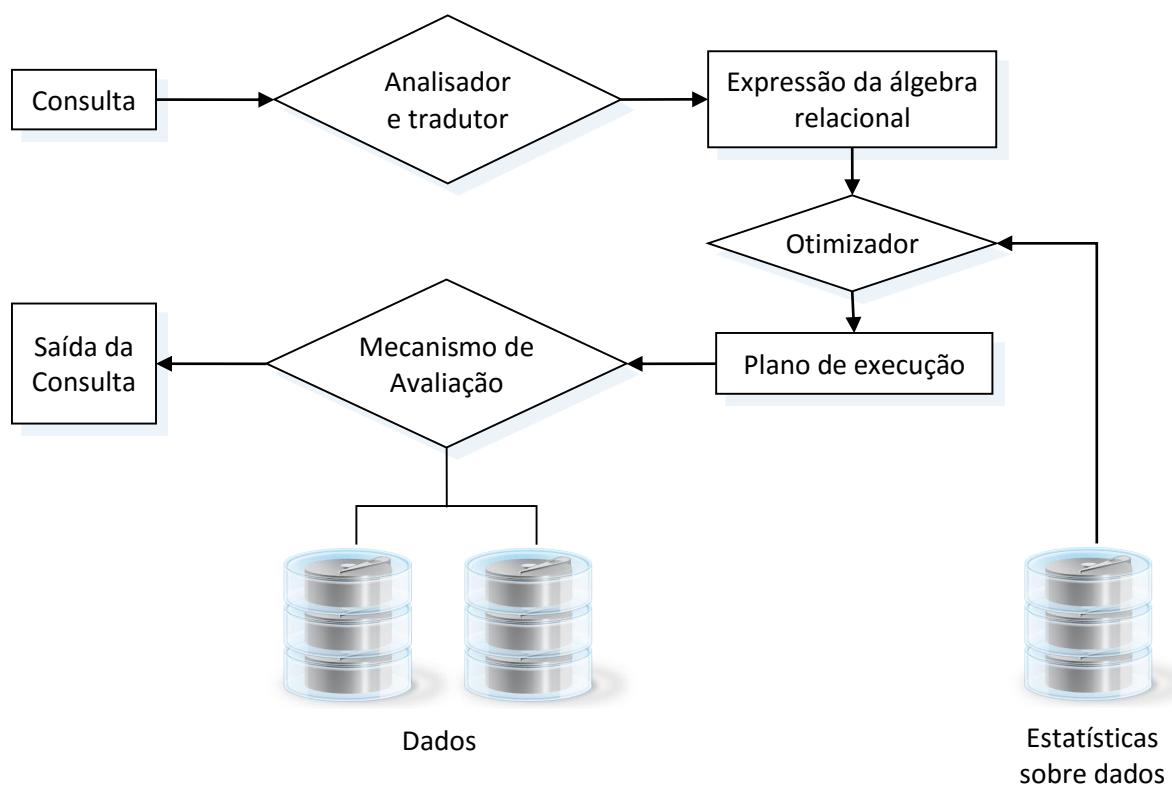


Figura 27. Etapas no processamento da consulta

Fonte: Adaptado de Sudarshan e Silberschatz (2006, p. 357).

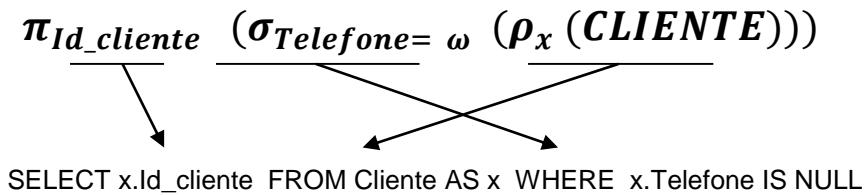
De acordo com Redmond (2012), SGBDRs são construídos com base em uma derivação da teoria dos conjuntos, a álgebra relacional, esta é uma combinação de projeções (SELECT...), de seleções (WHERE...), produtos cartesianos (JOIN...), entre outros.

Observe no exemplo a seguir a relação CLIENTE, note que nesta relação há tuplas com valores null para o atributo telefone. Veja de forma simplificada algumas operações da álgebra relacional sobre esta relação:

CLIENTE

Id_cliente	Telefone
1	9229655
2	
3	3455446
4	
5	3455446
6	3422018
7	

Tabela 3. Uma relação CLIENTE
Fonte: adaptado de Redmond (2012, p. 12).



O exemplo acima mostra uma *projeção* sobre a relação CLIENTE de todas as tuplas do atributo Id_cliente cuja *seleção* Telefone represente um subconjunto de indivíduos para qual a condição IS NULL seja avaliada como verdadeira. O resultado neste caso é uma nova relação:

Id_cliente
2
4
7

Tabela 4. Nova relação criada a partir da relação CLIENTE
Fonte: adaptado de Redmond (2012, p. 12).

2.17 Operações relacionais Unárias

Uma operação é chamada de unária, quando ela opera em uma única relação, as operações PROJEÇÃO, SELEÇÃO e RENOMEAÇÃO são portanto operações relacionais unárias.

De acordo com (Elmasri e Navathe 2011), a PROJEÇÃO, indicada pela letra grega pi (π), pode ser entendida como uma operação que filtra os atributos (*Colunas*) de uma relação (*tabela*), ou seja, a operação PROJEÇÃO seleciona certas Colunas da tabela e descarta as outras, por exemplo, se estivermos interessados apenas em certos atributos de uma relação, usamos a operação PROJEÇÃO para projetar a relação apenas por esses atributos. Em SQL a lista de atributos de PROJEÇÃO é especificada na cláusula SELECT de uma consulta.

Representada pela letra grega sigma (σ) a SELEÇÃO pode ser entendida como uma operação que filtra as tuplas (*linhas*) de uma relação (*tabela*). Em geral permitimos comparações usando $=$, \neq , $<$, $>$, \leq , \geq . O número de tuplas na relação resultante é sempre menor ou igual ao número de tuplas da Relação selecionada. Em uma consulta SQL, a condição de SELEÇÃO normalmente é especificada na cláusula WHERE (Elmasri e Navathe 2011).

A operação relacional unária RENOMEAR, indicada pela letra grega minúscula rho (ρ), é usada para mudar o nome do atributo de uma relação ou a relação em si. É útil para auto-relacionamentos, onde precisamos fazer a junção de uma tabela com ela mesma, e nesse caso cada versão da tabela precisa receber um nome diferente. Em SQL a renomeação é obtida utilizando apelidos precedidos da palavra AS (Elmasri e Navathe 2011).

As demais operações que operam em partes de relações, ou seja, aplicadas a dois conjuntos de tuplas são chamadas de operações binárias (Sudarshan e Silberschatz 2006).

2.18 Operações de álgebra relacional com base na teoria dos conjuntos

2.18.1 As operações UNIÃO, INTERSECÇÃO e SUBTRAÇÃO.

O próximo grupo de operações da álgebra relacional é composto pelas operações matemáticas padrão sobre conjuntos.

Várias operações de teoria de conjunto são usadas para mesclar os elementos de dois conjuntos de diversas maneiras, incluindo UNIÃO, INTERSECÇÃO e DIFERENÇA DE CONJUNTO, (também chamada de SUBTRAÇÃO ou EXCETO). Estas são operações binárias, ou seja, cada uma é aplicada a dois conjuntos (de tuplas). Quando essas operações são adaptadas aos bancos de dados relacionais, as duas relações sobre as quais qualquer uma dessas três operações são aplicadas precisam ter o mesmo tipo de tuplas, essa condição é chamada de compatibilidade de união ou compatibilidade de tipo (Elmasri e Navathe 2011, p. 102).

Isso significa que duas relações são compatíveis na união ou compatíveis no tipo se as duas relações tiverem o mesmo número de atributos e cada par correspondente de atributos tem o mesmo domínio, em outras palavras, você não será capaz de aplicar estas operações se a quantidade de colunas não for igual e se o tipo de dado entre os pares correspondentes for diferente, por exemplo, textual com numérico. De acordo com Elmasri e Navathe (2011), ao aplicar as operações de União, Intersecção e Subtração, a relação resultante tem os mesmos nomes de atributo da primeira relação e sempre será possível renomear os atributos no resultado usando o operador de renomeação.

Para facilitar o entendimento utilizaremos duas relações, sendo elas CLIENTE e FUNCIONARIO, a relação CLIENTE representaremos como R e a relação FUNCIONARIO representaremos como S .

CLIENTE (Representado como R)

CPF_Cli	Cli_Nome
98745632147	Susana Yao
99956215478	Ronaldo Lima
88956231471	José Gonçalves
78945563214	Barbara Pires
88745663218	Ana Tavares
88987785456	Jonas Wang
78899865478	Ernesto Gilberto

FUNCIONARIO (Representado como S)

CPF_Func	Func_Nome
78945563214	Barbara Pires
88987785456	Jonas Wang
95236874195	Alana Peres
78699863428	Felipe Lima
91738246654	Karine Yao

Tabela 5. Duas Relações, CLIENTE e FUNCIONARIO

Fonte: Adaptado de Elmasri e Navathe (2011, p. 103)

Observe que as duas relações são compatíveis pois contém o mesmo número de colunas, e tanto a primeira quanto a segunda coluna possuem o mesmo tipo de dados nas duas relações, ou seja, a primeira coluna da relação CLIENTE tem um domínio numérico, assim como a primeira coluna da relação FUNCIONARIO. O domínio da segunda coluna da relação CLIENTE é textual, assim como da segunda coluna da relação FUNCIONARIO. É importante entender o conceito de domínio. Domínio não é apenas o conjunto de valores permitidos para um dado, domínio também é o tipo de dado permitido para um conjunto de valores em um atributo. Por exemplo, um domínio textual pode aceitar valores de data e valores numéricos, no entanto, um domínio do tipo numérico não aceitará valores textuais, assim como um domínio do tipo data permitirá apenas datas nos valores de um determinado atributo.

Veja a seguir as três operações UNIÃO, INTERSECÇÃO e SUBTRAÇÃO sobre as duas relações compatíveis na união, R e S como se segue.

2.18.2 União (U)

De acordo com Elmasri e Navathe (2011, p. 102), “o resultado dessa operação, indicada por $R \cup S$, é uma relação que inclui todas as tuplas que estão em R ou S , ou tanto em R quanto em S ”. Note que nesta operação as tuplas duplicadas são eliminadas e relação resultante tem os mesmos nomes de atributo da primeira relação.

CPF_Cli	cli_Nome
78699863428	Felipe Lima
78899865478	Ernesto Gilberto
78945563214	Barbara Pires
88745663218	Ana Tavares
88956231471	José Gonçalves
88987785456	Jonas Wang
91738246654	Karine Yao
95236874195	Alana Peres
98745632147	Susana Yao
99956215478	Ronaldo Lima

Em SQL:

```
SELECT CPF_CLI, CLI_NOME
FROM CLIENTE
UNION
SELECT CPF_FUNC, DEP_NOME
FROM FUNCIONARIO;
```

Tabela 6. União sobre as relações CLIENTE e FUNCIONARIO
Fonte: Adaptado de Elmasri e Navathe (2011, p. 103)

2.18.3 Intersecção (\cap)

Conforme Elmasri e Navathe (2011, p. 102), “O resultado dessa operação, indicada por $R \cap S$, é uma relação que inclui todas as tuplas que estão tanto em R quanto em S ”. No nosso exemplo, o resultado da operação INTERSECÇÃO inclui apenas aqueles que são tanto Clientes quanto Funcionários.

CPF_Cli	cli_Nome
78945563214	Barbara Pires
88987785456	Jonas Wang

Em SQL:

```
SELECT CPF_CLI, CLI_NOME
FROM CLIENTE
INTERSECT
SELECT CPF_FUNC, DEP_NOME
FROM FUNCIONARIO;
```

Tabela 7. Intersecção sobre as relações CLIENTE e FUNCIONARIO
Fonte: Adaptado de Elmasri e Navathe (2011, p. 103).

Observe em nosso exemplo, que apenas dois supervisores retornaram como funcionários, assumimos que os demais supervisores são terceiros e portanto não fazem parte do quadro de funcionários.

2.18.4 Diferença de Conjunto (ou Subtração) (-)

O resultado dessa operação, indicada por $R - S$, é uma relação que inclui todas as tuplas que estão em R mas não em S , ou seja, mostra apenas os clientes que não são funcionários (Elmasri e Navathe 2011).

CPF_Cli	Cli_Nome
78899865478	Ernesto Gilberto
88745663218	Ana Tavares
88956231471	José Gonçalves
98745632147	Susana Yao
99956215478	Ronaldo Lima

Em SQL:

```
SELECT CPF_CLI, CLI_NOME
FROM CLIENTE
MINUS
SELECT CPF_FUNC, DEP_NOME
FROM FUNCIONARIO;
```

Tabela 8. Subtração sobre as relações CLIENTE e FUNCIONARIO
Fonte: Adaptado de Elmasri e Navathe (2011, p. 103).

Observe que tanto UNIÃO quanto INTERSECÇÃO são operações comutativas, ou seja, a união de CLIENTE com FUNCIONARIO é igual à união de FUNCIONARIO com CLIENTE ($R \cup S = S \cup R$) e a intersecção de CLIENTE com FUNCIONARIO é igual à intersecção de FUNCIONARIO com CLIENTE ($R \cap S = S \cap R$), a ordem não interfere no resultado. Conforme (Elmasri e Navathe 2011), a operação de SUBTRAÇÃO não é comutativa, em geral, $R - S \neq S - R$, abaixo veja um exemplo de $S - R$, observe que o resultado agora é diferente pois mostra os funcionários que não são clientes.

CPF_Func	Func_Nome
78699863428	Felipe Lima
91738246654	Karine Yao
95236874195	Alana Peres

Em SQL:

```
SELECT CPF_FUNC, DEP_NOME
FROM FUNCIONARIO
MINUS
SELECT CPF_CLI, CLI_NOME
FROM CLIENTE;
```

Tabela 9. Subtração sobre as relações CLIENTE e FUNCIONARIO
Fonte: Adaptado de Elmasri e Navathe (2011, p. 103).

2.18.5 A operação PRODUTO CARTESIANO

Conforme Elmasri e Navathe (2011), a operação PRODUTO CARTESIANO, também conhecida como PRODUTO CRUZADO ou JUNÇÃO CRUZADA, é indicada por \times (sinal de vezes). Esta também é uma operação de conjunto binária, mas as relações sobre as quais ela é aplicada *não* precisam ser compatíveis na união. Em sua forma binária, esta operação de conjunto produz um novo elemento combinando cada membro (tupla) de uma relação (Conjunto) com cada membro (Tupla) de outra relação (Conjunto). O resultado dessa operação, indicada por $R \times S$, é a combinação dos atributos das duas relações.

Para demonstrar o exemplo de produto cartesiano, considere agora como R a relação FUNCIONARIO e representada como S a relação DEPENDENTE.

FUNCIONARIO (Representado como R)

CPF_Func	Func_Nome
78945563214	Barbara Pires
88987785456	Jonas Wang
95236874195	Alana Peres
78699863428	Felipe Lima
91738246654	Karine Yao

DEPENDENTE (Representado como S)

F_CPF	Nome_Dep	Sexo_Dep	Data_Nasc_Dep
55689735444	Joao Mendes	M	15/03/83
91738246653	Ricardo Filho	M	24/10/05
91738246654	Susana Yao	F	30/12/99
88987785456	Seiko Wang	F	01/06/97
78699863428	Celio Lima	M	24/02/00

Tabela 10. Duas Relações FUNCIONARIO e DEPENDENTE

Fonte: Adaptado de Elmasri e Navathe (2011, p. 103).

A Relação a seguir é resultado da aplicação da operação PRODUTO CARTESIANO a DEPENDENTE. Cada tupla de FUNCIONARIO é combinada com cada tupla de DEPENDENTE, dando um resultado que não é muito significativo.

Como regra o número de tuplas retornadas na nova relação RESULTADO_PROD é o número de tuplas da relação R multiplicado pelo o número de tuplas da relação S .

RESULTADO_PROD

CPF_Func	Func_Nome	F_CPF	Nome_Dep	Sexo_Dep	Data_Nasc_Dep
78945563214	Barbara Pires	55689735444	Joao Mendes	M	15/03/83
88987785456	Jonas Wang	55689735444	Joao Mendes	M	15/03/83
95236874195	Alana Peres	55689735444	Joao Mendes	M	15/03/83
78699863428	Felipe Lima	55689735444	Joao Mendes	M	15/03/83
91738246654	Karine Yao	55689735444	Joao Mendes	M	15/03/83
78945563214	Barbara Pires	91738246653	Ricardo Filho	M	24/10/05
88987785456	Jonas Wang	91738246653	Ricardo Filho	M	24/10/05
95236874195	Alana Peres	91738246653	Ricardo Filho	M	24/10/05
78699863428	Felipe Lima	91738246653	Ricardo Filho	M	24/10/05
91738246654	Karine Yao	91738246653	Ricardo Filho	M	24/10/05
78945563214	Barbara Pires	91738246654	Susana Yao	F	30/12/99
88987785456	Jonas Wang	91738246654	Susana Yao	F	30/12/99
95236874195	Alana Peres	91738246654	Susana Yao	F	30/12/99
78699863428	Felipe Lima	91738246654	Susana Yao	F	30/12/99
91738246654	Karine Yao	91738246654	Susana Yao	F	30/12/99
78945563214	Barbara Pires	88987785456	Seiko Wang	F	01/06/97
88987785456	Jonas Wang	88987785456	Seiko Wang	F	01/06/97
95236874195	Alana Peres	88987785456	Seiko Wang	F	01/06/97
78699863428	Felipe Lima	88987785456	Seiko Wang	F	01/06/97
91738246654	Karine Yao	88987785456	Seiko Wang	F	01/06/97
78945563214	Barbara Pires	78699863428	Celio Lima	M	24/02/00
88987785456	Jonas Wang	78699863428	Celio Lima	M	24/02/00
95236874195	Alana Peres	78699863428	Celio Lima	M	24/02/00
78699863428	Felipe Lima	78699863428	Celio Lima	M	24/02/00
91738246654	Karine Yao	78699863428	Celio Lima	M	24/02/00

Tabela 11. Produto cartesiano de FUNCIONARIO com DEPENDENTE

Fonte: Adaptado de Elmasri e Navathe (2011, p. 103).

Segundo Elmasri e Navathe 2011, a operação PRODUTO CARTESIANO aplicada isoladamente não tem significado. Ela é útil quando seguida por uma seleção que combina valores de atributos vindos das relações componentes. A tabela 12 mostra a nova Relação de um Produto cartesiano com atributos chave combinados.

$$\text{FUNC_DEPENDENTES} \leftarrow \sigma_{\text{CPF_Func}=\text{F_CPF}} (\text{RESULTADO_PROD})$$

FUNC_DEPENDENTES

CPF_Func	Func_Nome	F_CPF	Nome_Dep	Sexo_Dep	Data_Nasc_Dep
88987785456	Jonas Wang	88987785456	Seiko Wang	F	01/06/97
78699863428	Felipe Lima	78699863428	Celio Lima	M	24/02/00
91738246654	Karine Yao	91738246654	Susana Yao	F	30/12/99

Tabela 12. Produto cartesiano com atributos chave combinados
Fonte: Adaptado de Elmasri e Navathe (2011, p. 103).

Ainda conforme Elmasri e Navathe (2011), o PRODUTO CARTESIANO cria tuplas com os atributos combinados de duas relações. Podemos selecionar (SELEÇÃO) tuplas relacionadas somente das duas relações especificando uma condição de seleção apropriada após o produto cartesiano. Como essa seqüência de produto cartesiano seguida por seleção é muito utilizada para combinar tuplas relacionadas de duas relações, uma operação especial, chamada JUNÇÃO, foi criada para especificar essa seqüência como uma única operação. Em SQL, o PRODUTO CARTESIANO pode ser realizado usando a opção CROSS JOIN nas tabelas juntadas. Como alternativa, se houver duas tabelas na consulta e na cláusula WHERE não houver condição de junção correspondente, ou ainda se não houver cláusula WHERE, o resultado também será o PRODUTO CARTESIANO das duas tabelas.

O produto cartesiano RESULTADO_PROD em SQL:

```
select * from FUNCIONARIO, DEPENDENTE;
```

A relação FUNC_DEPENDENTES em SQL:

```
select * from FUNCIONARIO, DEPENDENTE
WHERE funcionario.cpf_func = dependente.f_cpf;
```

2.19 Operações relacionais binárias

2.19.1 Junção(\bowtie)

A operação de junção, indicada por \bowtie , é usada para combinar tuplas relacionadas de duas relações em uma única tupla maior. Essa operação é muito importante para qualquer banco relacional com mais de uma relação única, porque nos permite processar relacionamentos entre as relações. A operação JUNÇÃO forma um produto cartesiano dos seus dois argumentos e realiza uma seleção forçando igualdade nos atributos que aparecem nas duas relações, o resultado da JUNÇÃO é uma nova relação sempre que a combinação de tuplas satisfaça a condição de junção. Se nenhuma combinação de tuplas satisfizer a condição de junção, o resultado de uma JUNÇÃO é uma relação vazia, com zero tuplas. Importante ressaltar que em JUNÇÃO, apenas combinações de tuplas que satisfaçam a condição de junção aparecem no resultado, enquanto no PRODUTO CARTESIANO todas as combinações de tuplas são incluídas no resultado (Elmasri e Navathe 2011).

A forma geral de uma operação de junção sobre duas relações é:

$$R \bowtie_{\text{condição Junção}} S$$

Considere nosso exemplo com as relações FUNCIONARIO e DEPENDENTE, observe que o atributo F_CPF é uma chave estrangeira da relação DEPENDENTE que referencia CPF_FUNC, a chave primária da relação FUNCIONARIO. Essa restrição de integridade referencial desempenha um papel importante para que haja tuplas combinando na relação referenciada DEPENDENTE.

$$\text{FUNC_DEPENDENTES} \leftarrow \text{FUNCIONARIO} \bowtie_{\text{CPF_Func=F_CPF}} \text{DEPENDENTES}$$

Conforme Elmasri e Navathe 2011, a condição de junção é especificada sobre atributos das duas relações R e S e é avaliada para cada combinação de tuplas. Cada combinação de tupla para qual a condição de junção é avaliada como verdadeira é incluída na relação resultante como uma única tupla combinada.

FUNC_DEPENDENTES

CPF_Func	Func_Nome	F_CPF	Nome_Dep	Sexo_Dep	Data_Nasc_Dep
88987785456	Jonas Wang	88987785456	Seiko Wang	F	01/06/97
78699863428	Felipe Lima	78699863428	Celio Lima	M	24/02/00
91738246654	Karine Yao	91738246654	Susana Yao	F	30/12/99

Tabela 13. JUNÇÃO entre FUNCIONARIO e DEPENDENTE

Fonte: Adaptado de Elmasri e Navathe (2011, p. 104).

De acordo com Elmasri e Navathe (2011), as tuplas cujos atributos de junção são NULL ou dos quais a condição de junção é falsa, não aparecem no resultado. Nesse sentido a operação de JUNÇÃO não necessariamente preserva toda a informação das relações participantes, pois as tuplas que não são combinadas com as correspondentes na outra relação não aparecem no resultado.

2.19.2 Variações de Junção: EQUIJUNÇÃO e JUNÇÃO NATURAL

O uso mais comum de JUNÇÃO envolve condições de junção apenas em comparações de igualdade. Esse tipo de JUNÇÃO, em que o único operador de comparação usado é =, é chamado de EQUIJUNÇÃO. O exemplo anterior foi EQUIJUNÇÃO. Observe que, no resultado de uma EQUIJUNÇÃO, sempre temos um ou mais pares de atributos que possuem valores idênticos em cada tupla, por exemplo, na tabela 13, os valores dos atributos CPF_Func e F_CPF são idênticos na tupla de FUNC_DEPENDENTES (o resultado da EQUIJUNÇÃO), porque a condição de junção de igualdade especificada sobre esses dois atributos requer que os valores sejam idênticos em cada tupla no resultado. Como um de cada par de atributos com valores idênticos é desnecessário, uma nova operação, chamada JUNÇÃO NATURAL, indicada por * foi criada para eliminar o segundo atributo (desnecessário) na condição de EQUIJUNÇÃO. A definição padrão de JUNÇÃO NATURAL requer que os dois atributos de junção tenham o mesmo nome nas duas relações. Se isso não acontecer, uma operação de renomeação é aplicada primeiro (Elmasri e Navathe 2011).

Note na expressão abaixo a operação de renomeação sobre o atributo F_CPF para CPF_Func. Se os atributos sobre os quais a junção natural é

especificada já tiverem os mesmos nomes nas duas relações, a renomeação não é necessária.

FUNC_DEPENDENTES \leftarrow **FUNCIONARIO** *

$$\rho_{(\text{CPF_Func}, \text{Nome_Dep}, \text{Sexo_Dep}, \text{Data_Nasc_Dep})} (\text{DEPENDENTES})$$

O atributo **CPF_Func** é chamado de atributo de junção para a operação JUNÇÃO NATURAL, pois é o único atributo com o mesmo nome nas duas relações.

A Relação resultante é ilustrada a seguir:

CPF_Func	Func_Nome	Nome_Dep	Sexo_Dep	Data_Nasc_Dep
88987785456	Jonas Wang	Seiko Wang	F	01/06/97
78699863428	Felipe Lima	Celio Lima	M	24/02/00
91738246654	Karine Yao	Susana Yao	F	30/12/99

Tabela 14. JUNÇÃO NATURAL entre FUNCIONARIO e DEPENDENTE
Fonte: Adaptado de Elmasri e Navathe (2011, p. 106).

Uma definição mais geral, porém não padrão, para JUNÇÃO NATURAL é:

$Q \leftarrow R *_{(<\text{lista1}>), (<\text{lista2}>)} S$

Nesse caso, $<\text{lista1}>$ especifica uma lista de atributos de R , e $<\text{lista2}>$ especifica uma lista de atributos de S . As listas são usadas para formar condições de comparação de igualdade entre pares de atributos correspondentes, e as condições passam então por um AND. Somente a lista correspondente aos atributos da primeira relação R ($<\text{Lista1}>$), é mantida no resultado Q . Em SQL, a JUNÇÃO pode ser realizada de diversas maneiras, o mais comum é especificar as condições de junção na cláusula WHERE (Elmasri e Navathe 2011).

O resultado da junção natural, a relação FUNC_DEPENDENTES em SQL é obtida da seguinte maneira:

```
select FUNCIONARIO.*, NOME_DEP, SEXO_DEP, DATA_NASC_DEP
from FUNCIONARIO, DEPENDENTE
WHERE funcionario.cpf_func = dependente.f_cpf;
```

2.19.3 Junção interna (INNER JOIN) e Junção externa (OUTER JOIN)

As operações de JUNÇÃO descritas anteriormente combinam com tuplas que satisfazem a condição de junção. Por exemplo, para uma operação JUNÇÃO NATURAL $R * S$, somente tuplas de R que possuem tuplas combinando em S , e vice-versa, aparecem no resultado. Logo, as tuplas sem uma tupla correspondente, ou relacionada, são eliminadas do resultado. As tuplas com valores NULL nos atributos de junção também são eliminadas. Esse tipo de junção, em que as tuplas sem correspondência são eliminadas, é conhecida como junção interna ou INNER JOIN. As operações de junção descritas até aqui são todas internas. Isso equivale à perda de informações se o usuário quiser que o resultado da JUNÇÃO inclua todas as tuplas em uma ou mais relações componentes.

De acordo com (Elmasri e Navathe 2011) um conjunto de operações chamadas junções externas, ou OUTER JOINS, foi desenvolvido para o caso em que o usuário deseje manter todas as tuplas em R , ou todas em S , ou todas aquelas nas duas relações no resultado da JUNÇÃO, independente delas possuírem ou não tuplas correspondentes na outra relação. Isso satisfaz a necessidade de consultas em que as tuplas das duas tabelas devem ser combinadas por linhas correspondentes, mas sem perda de quaisquer tuplas por falta de valores correspondentes.

Por exemplo, considere nossas relações FUNCIONARIO e DEPENDENTE, suponha que queiramos uma lista de todos os funcionários, bem como a relação de dependentes relacionados. Caso não haja dependentes para este funcionário, queremos que o valor das tuplas não correspondentes seja indicado por um valor NULL. Podemos aplicar uma operação JUNÇÃO EXTERNA À ESQUERDA, indicada por \bowtie , para recuperar o resultado da seguinte forma:

$\text{TEMP} \leftarrow (\text{FUNCIONARIO} \bowtie_{\text{CPF_Func}=\text{F_CPF}} \text{DEPENDENTE}$

$\text{RESULTADO} \leftarrow \pi_{\text{CPF_Func}, \text{Func_nome}, \text{Nome_Dep}} (\text{TEMP})$

RESULTADO

CPF_Func	Func_Nome	Nome_Dep
91738246654	Karine Yao	Susana Yao
88987785456	Jonas Wang	Seiko Wang
78699863428	Felipe Lima	Celio Lima
95236874195	Alana Peres	
78945563214	Barbara Pires	

Tabela 15. LEFT JOIN entre FUNCIONARIO e DEPENDENTE

Fonte: Adaptado de Elmasri e Navathe (2011, p. 114).

Repare que a operação JUNÇÃO EXTERNA À ESQUERDA mantém cada tupla na primeira relação, R , ou da esquerda, em $R \bowtie S$. Se nenhuma tupla correspondente for encontrada em S , então os atributos de S no resultado da junção são preenchidos com valores NULL (Elmasri e Navathe 2011).

Em SQL utiliza-se LEFT OUTER JOIN, ou apenas LEFT JOIN, para aplicação da JUNÇÃO EXTERNA À ESQUERDA:

```
select CPF_FUNC, FUNC_NOME, NOME_DEP
from FUNCIONARIO
left join DEPENDENTE on (funcionario.cpf_func = dependente.f_cpf);
```

Uma operação parecida, a JUNÇÃO EXTERNA À DIREITA, indicada por $\bowtie\lhd$, mantém cada tupla na segunda relação, S , ou da direita, no resultado de $R \bowtie\lhd S$. A tabela a seguir mostra o resultado da operação RIGHT JOIN entre FUNCIONARIO e DEPENDENTE.

RESULTADO de $R \bowtie\lhd S$

CPF_Func	Func_Nome	Nome_Dep
88987785456	Jonas Wang	Seiko Wang
78699863428	Felipe Lima	Celio Lima
91738246654	Karine Yao	Susana Yao
		Ricardo Filho
		Joao Mendes

Tabela 16. RIGHT JOIN entre FUNCIONARIO e DEPENDENTE

Fonte: Adaptado de Elmasri e Navathe (2011, p. 114).

Em SQL utiliza-se RIGHT OUTER JOIN, ou apenas RIGHT JOIN, para aplicação da JUNÇÃO EXTERNA À DIREITA:

```
select CPF_FUNC, FUNC_NOME, NOME_DEP
from FUNCIONARIO
right join DEPENDENTE on (funcionario.cpf_func = dependente.f_cpf);
```

Uma terceira operação, JUNÇÃO EXTERNA COMPLETA, indicada por \bowtie , mantém todas as tuplas nas relações da esquerda e da direita quando nenhuma tupla correspondente for encontrada, preenchendo-as com valores NULL conforme a necessidade. Observe que agora é possível identificar dois dependentes que não estão associados a um funcionário, isso é uma inconsistência na relação do nosso exemplo, onde o dependente possui um ID_Dependente como chave mas não possui o número de documento ao qual ele pertence. Na vida real, casos como este geram divergências nos números apresentados em relatórios. Observe na tabela 18 o resultado da operação FULL JOIN entre FUNCIONARIO e DEPENDENTE.

RESULTADO de $R \bowtie S$

CPF_Func	Func_Nome	Nome_Dep
		Joao Mendes
		Ricardo Filho
91738246654	Karine Yao	Susana Yao
88987785456	Jonas Wang	Seiko Wang
78699863428	Felipe Lima	Celio Lima
95236874195	Alana Peres	
78945563214	Barbara Pires	

Tabela 17. FULL JOIN entre FUNCIONARIO e DEPENDENTE
Fonte: Adaptado de Elmasri e Navathe (2011, p. 114).

Em SQL utiliza-se FULL OUTER JOIN, ou apenas FULL JOIN, para aplicação da JUNÇÃO EXTERNA COMPLETA:

```
select CPF_FUNC, FUNC_NOME, NOME_DEP
from FUNCIONARIO
full join DEPENDENTE on (funcionario.cpf_func = dependente.f_cpf);
```

2.20 Sistemas Gerenciadores de Banco de Dados Relacionais (SGBDR)

De acordo com Stair (2009), um sistema gerenciador de banco de dados relacional (SGBDR) – do inglês relational database management system (RDBMS), é um conjunto de softwares utilizados para gerenciar bancos de dados estruturados de acordo com o modelo relacional. São sistemas implementados com tabelas bidimensionais, ou seja, tabelas com dados organizados em linhas e colunas, também chamadas de relações. Cada relação é composta por um conjunto de tuplas e um mapa de atributos. Cada atributo pode aceitar apenas certos valores ou tipos de dados, os valores ou tipo de dados permitidos para esse atributo é chamado de domínio. Os meios de interagir com um SGBDR é escrever consultas em Structured Query Language (SQL). Os valores dos dados são digitados e podem ser numéricos, textos, datas ou de outros tipos. Os tipos são impostos pelo sistema e devem ser definidos na criação do modelo. É importante ressaltar que, devido sua base matemática na teoria relacional dos conjuntos, relações (tabelas) podem juntar-se e transformarem-se em novas relações mais complexas.

Conforme Powell (2006), às vezes Kits de desenvolvimento de software (SDK), ferramentas de front-end e kits completos de gerenciamento de pacotes estão incluídos em um SGBDR. Microsoft Access é um exemplo disso, ambas as ferramentas de banco de dados e front-end de desenvolvimento relacional, para a construção de telas de entrada, estão disponíveis dentro do mesmo software.

Segundo Redmond (2012), bancos de dados relacionais não são relacionais porque as tabelas "relacionam-se" umas com as outras através de chaves primárias e chaves estrangeiras, o poder do modelo é certamente a matemática. Esta característica permite aos usuários expressar consultas poderosas e, em seguida, permite que o sistema as aperfeiçoe, baseado em padrões predefinidos.

Os SGBDRs populares atuais incluem o DB2 e Informix Dynamic Server (da IBM), o Oracle e Rdb (da Oracle), o Sybase SGBD (da Sybase) e o SQLServer e Access (da Microsoft). Além disso, vários sistemas de código aberto, como MySQL e PostgreSQL, estão disponíveis.



Figura 28. Sistemas Gerenciadores de Bancos de dados Relacionais
Fonte: Autor.

2.20.1 Objetivos de Bancos de dados Relacionais

Conforme Sudarshan e Silberschatz (2006), em geral, o objetivo de um projeto de banco de dados relacional é um conjunto de esquemas de relação que nos permite armazenar informações sem redundância desnecessária, além de permitir recuperar informações facilmente. Isso é obtido projetando esquemas que estejam em uma forma normal adequada.

De acordo com Heuser (2008), uma forma normal é um conjunto de regras que devem ser obedecidas por uma tabela para evitar problemas indesejáveis, como por exemplo, repetição de informação e inconsistência de dados. Há diversas formas normais, isto é, diversas regras para verificar tabelas relacionais. No caso deste trabalho, vamos considerar as primeiras três formas normais. Na prática, considera-

se que uma base de dados está normalizada se aderir à terceira forma normal. Ao processo de análise e aplicação de testes para certificação das formas normais dá-se o nome de normalização.

2.20.2 Dependência funcional

A idéia Central para a definição de normalização é a de dependência funcional. Dependências funcionais são uma maneira de descrever a interdependência dos atributos ou campos nas tabelas.

Segundo Elmasri e Navathe (2011), uma dependência funcional é uma restrição entre dois atributos do banco de dados que descreve como eles se relacionam entre si. Por exemplo, quando os valores de uma tupla no relacionamento de um atributo Y dependem ou são determinados pelos valores de tuplas do atributo X, ou seja, os valores de um atributo X determinam exclusivamente ou funcionalmente os valores do atributo Y, dizemos que existe uma dependencia funcional de X para Y, ou que Y é funcionalmente dependente de X. A abreviação para dependencia funcional é DF.

Para exemplificar o entendimento, observe a seguir:

CLIENTE

ID_Cliente	Nome_Cliente	Cidade_Cliente
1001	Mateus Filipe Garcia	BALSA NOVA
1004	Leonor Sofia Menezes	CURITIBA
1012	Afonso Alexandre Leao	PARANAGUA
1018	Maria Amelia Cavalcante	COLOMBO
1031	Renata Filipa Alves	CURITIBA
1039	Gabriel Luis Oliveira	CURITIBA
1056	Mateus Filipe Garcia	LONDRINA

Tabela 18. Dependência funcional entre atributos de uma relação
Fonte: Adaptado de Churcher, (2007, p. 115).

Para a situação descrita na Figura 18, se eu sei que o identificador de um cliente é 1001, posso dizer que seu nome é “Mateus Filipe Garcia”. Agora observe, será que funciona o contrário? Se eu sei o nome de um cliente, posso dizer-lhe

exclusivamente o seu número de identificação? Repare que neste exemplo simulamos uma situação real, onde é possível haver vários clientes com o mesmo nome, de modo que saber apenas o nome não determina exclusivamente o seu ID. De uma maneira mais formal, Nome_cliente funcionalmente não determina ID_Cliente. ID_Cliente é o atributo determinante desta tabela, também conhecido como atributo chave. Se sabemos os valores dos atributos-chave, podemos encontrar uma linha única na tabela. Uma vez que temos essa linha, então nós sabemos o valor de todos os outros atributos.

Por exemplo, se eu sei o ID_cliente, eu posso encontrar uma linha única na tabela de clientes e assim ser capaz de determinar o Nome e Cidade. Ou, em termos de dependências funcionais, conforme relata Churcher (2007), os atributos chave funcionalmente determinam todos os outros campos na tabela, este atributo chamamos de chave primária.

Confome Elmasri e Navathe (2011), a normalização de dados pode ser considerada um processo de analisar os esquemas de relação com base em suas dependências funcionais (DFs) e chaves primárias para conseguir as propriedades desejadas de minimização da redundância e minimização das anomalias de inserção, exclusão e atualização de dados.

2.21 Normalização de Relações

De acordo com Elmasri e Navathe (2011), o processo de normalização proposto inicialmente por Codd em 1972, leva um esquema de relação por uma série de testes para certificar se ele satisfaz certa forma normal. O processo prossegue em um padrão de cima para baixo, avaliando cada relação em comparação com os critérios para as formas normais, ou seja para aplicar a 2NF primeiramente é necessário que seja aplicada a 1FN, da mesma forma, para aplicar a 3NF é necessário que a tabela esteja normalizada na 2NF. Esquemas de relação insatisfatórios, que não atendem a certas condições, os testes de forma normal, são decompostos em esquemas de relação menores, que atendem aos testes e, portanto, possuem as propriedades desejáveis.

Inicialmente, Codd propôs três formas normais, que ele chamou de primeira, segunda e terceira forma normal. Uma definição mais forte da 3FN, chamada Forma

Normal Boyce-Codd (FNBC), foi proposta posteriormente por Boyce e Codd. Todas essas formas normais estão baseadas em uma única ferramenta analítica: as dependências funcionais entre os atributos de uma relação. Depois, uma quarta forma normal (4FN) e uma quinta forma normal (5FN) foram propostas, com base nos conceitos de dependências multivaloradas e depéndências de junção, respectivamente. Por definição, a forma normal de uma relação refere-se à condição de forma normal mais alta a que ela atende e, portanto, indica o grau ao qual ela foi normalizada Elmasri e Navathe (2011).



Figura 29. Etapas de um processo de normalização
Fonte: Autor.

2.21.1 Uso das formas normais na prática

Conforme Elmasri e Navathe (2011), embora várias formas normais mais altas tenham sido definidas, como a 4FN e a 5FN, a utilidade prática dessas formas normais torna-se questionável quando as restrições sobre as quais elas estão baseadas são raras, e difíceis de entender ou detectar pelos projetistas e usuários de banco de dados que precisam descobrir essas restrições, desta forma, o projeto de banco de dados praticado na indústria hoje presta atenção particular à normalização apenas até a 3FN.

Uma observação importante é que os projetistas de banco de dados não precisam normalizar para a forma normal mais alta possível. As relações podem ser deixadas em um estado de normalização inferior, como a 2FN, por questões de desempenho, fazer isso gera as penalidades correspondentes de lidar com as anomalias suscetíveis ao grau de normalização aplicado Elmasri e Navathe (2011).

2.21.2 Primeira forma Normal (1FN)

Você notou na etapa de modelagem conceitual que o modelo ER permite que conjuntos de entidades e conjuntos de relacionamento tenham atributos com algum grau de subestrutura. Especificamente, ele permite atributos de valores múltiplos e atributos compostos.

Segundo Elmasri e Navathe (2011), a primeira forma normal foi definida para reprovar atributos multivvalorados, atributos compostos e suas combinações. Ela afirma que o domínio de um atributo deve incluir apenas valores atômicos (simples, indivisíveis) e que o valor de qualquer atributo em qualquer tupla deve ser um único valor do domínio desse atributo. Em outras palavras, a 1FN reprova grupos repetitivos de itens de dados.

Veja a seguir o exemplo de uma tabela não normalizada, inicialmente em um formato de arquivo departamental, em seguida, o passo a passo para entendimento do processo de normalização para a 1FN.

A tabela Cliente, demonstrada nesse exemplo possui seis atributos, o atributo ID_Cliente é a chave primária, ela identifica cada linha exclusivamente, no entanto, note que temos um atributo composto (Endereço) e dois atributos multivvalorados (Telefone e Dependente), observe que o domínio destes atributos possui conjuntos de valores e portanto não é atômico.

Esta tabela está suscetível a diversos problemas, por exemplo, suponha que precisamos atualizar um dos números de telefone, o cliente que possui dois números registrados pode perder um dos números já cadastrados, o mesmo serve para adicionar um novo número, além de que por um erro de digitação o cliente pode perder os dois telefones de contato.

Repare também que temos informações de dependentes nessa tabela, na estrutura atual fica inviável contar o número de dependentes por clientes, além de que teríamos problemas caso ocorra a necessidade de filtrar apenas um dos dependentes ou ainda futuramente adicionar detalhes dos dependentes, por exemplo sexo, data de nascimento, CPF, etc...

CLIENTE (Não normalizada)

ID_CLIENTE	NOME_CLIENTE	ENDERECO	TELEFONE	NOME_DEPENDENTE	GRAU_DEPENDENCIA
5	ROSA SILVA	AV. REP ARGENTINA 2350 Sala7 80240000 CURITIBA PR	4134554467 4132267778	FELIPE SILVA, AMANDA SILVA	FILHO
31	JOSE SOUZA	R. JOAO NEGRAO 280 80010200 CURITIBA PR	413044000	AMANDA SOUZA	FILHO
81	SIMIRAMYS SOUZA	R. JOAO BIENTINEZ 11 82810010 CURITIBA PR		JOAO SOUZA	FILHO
163	EDUARDO JACOB	MAJOR AMERICO GUIMARAES 224 82520260 CURITIBA PR	4132331783 4136697887	MIGUEL JACOB	IRMAO
198	LUCIANE SCHWALBE	R. MAL DEODORO DA FONSECA 520 89521700 JARAGUA DO SUL SC	4733710313	SUELEN SCHWALBE	FILHO
204	ARGEMIRO ALUISIO	R. MONTE CARLO 37 87010060 MARINGA PR	4432325310 4996587778	DANTE LARLING	FILHO

Tabela 19. 1FN e uma tabela não normalizada - Layout 1

Fonte: Autor.

O primeiro passo é trabalhar a tabela CLIENTE para identificar relações aninhadas. Para atributos compostos, deixamos cada componente ser um atributo por si só, para atributos de valores múltiplos, criamos uma tupla para cada item do conjunto de valores múltiplos, em alguns casos, se o número máximo de valores para o atributo for conhecido, é conveniente criarmos para cada componente um novo atributo. Por exemplo, o atributo telefone, se considerarmos um limite máximo de dois números por cliente, podemos separar este atributo multivalorado em Telefone1 e Telefone2, essa solução é necessária mas tem a desvantagem de inserir valores NULL se a maioria dos clientes possuir apenas um telefone de contato.

CLIENTE (Não normalizada)

ID_CLIENTE	NOME_CLIENTE	ENDERECO	TELEFONE1	TELEFONE2	NOME_DEPENDENTE	GRAU_DEPENDENCIA
5	ROSA SILVA	AV. REP ARGENTINA 2350 Sala7 80240000 CURITIBA PR	4134554467	4132267778	FELIPE SILVA	FILHO
					AMANDA SILVA	
31	JOSE SOUZA	R. JOAO NEGRAO 280 80010200 CURITIBA PR	413044000		AMANDA SOUZA	FILHO
81	SIMIRAMYS SOUZA	R. JOAO BIENTINEZ 11 82810010 CURITIBA PR			JOAO SOUZA	FILHO
163	EDUARDO JACOB	MAJOR AMERICO GUIMARAES 224 82520260 CURITIBA PR	4132331783	4136697887	MIGUEL JACOB	IRMAO
198	LUCIANE SCHWALBE	R. MAL DEODORO DA FONSECA 520 89521700 JARAGUA DO SUL SC	4733710313		SUELEN SCHWALBE	FILHO
204	ARGEMIRO ALUISIO	R. MONTE CARLO 37 87010060 MARINGA PR	4432325310	41996587778	DANTE LARLING	FILHO

Tabela 20. 1FN e uma tabela não normalizada - Layout 2

Fonte: Autor.

De acordo com Heuser (2008), podemos dizer que a tabela CLIENTE é um conjunto de relações aninhadas, pois temos a relação CLIENTE e uma nova relação na mesma tabela, que podemos chamar de DEPENDENTE, observe que temos atributos com grupos de repetição, (NOME_DEPENDENTE e GRAU_DEPENDENCIA). Por questões de espaço na demonstração, o atributo Endereço não foi separado em vários atributos. Veja abaixo como fica a estrutura da tabela ajustando os atributos Endereço e Telefone:

CLIENTE (Não normalizada)

CLIENTE
ID_CLIENTE
NOME_CLIENTE
LOGR_CLIENTE
NUMERO_LOGR_CLIENTE
COMPL_LOGR_CLIENTE
CEP_CLIENTE
CIDADE_CLIENTE
UF_CLIENTE
DDD_CLIENTE
FONE1_CLIENTE
FONE2_CLIENTE
NOME_DEPENDENTE
GRAU_DEPENDENCIA

Tabela 21. 1FN e uma tabela não normalizada - Layout 3
Fonte: Autor.

Neste ponto, para normalizarmos a tabela CLIENTE na primeira forma normal (1FN), precisamos criar uma tabela separada para a informação de dependentes, a ordem das linhas e colunas não importa, o importante é relacionarmos a tabela CLIENTE com DEPENDENTE na cardinalidade correta, como um cliente pode ter mais de um dependente, a cardinalidade máxima para este tipo de relacionamento deve ser (1:N), ou seja, um para muitos.

A tabela DEPENDENTE precisa ter um identificador para cada dependente, por este motivo criamos COD_DEPENDENTE, se um cliente tem dois dependentes, o código cliente terá associado um código para cada Dependente, este novo campo será parte da chave primária composta de ID_CLIENTE e COD_DEPENDENTE.

De acordo com Elmasri e Navathe (2011), em alguns casos, mesmo sendo um atributo composto, o campo nome pode ser considerado um atributo atômico, por

este motivo optamos por não separar o atributo NOME_DEPENDENTE em nome e sobrenome.

DEPENDENTE (Nova relação Criada)

COD_DEPENDENTE	ID_CLIENTE	NOME_DEPENDENTE	GRAU_DEPENDENCIA
1	5	FELIPE SILVA	FILHO
2	5	AMANDA SILVA	FILHO
1	31	AMANDA SOUZA	FILHO
1	81	JOAO SOUZA	FILHO
1	163	MIGUEL JACOB	IRMAO
1	198	SUELEN SCHWALBE	FILHO
1	204	DANTE LARLING	FILHO

Tabela 22. 1FN nova relação criada

Fonte: Autor.

Abaixo a tabela 23 mostra a tabela CLIENTE normalizada na primeira forma normal, observe que agora temos duas relações onde cada relação possui atributos atômicos.

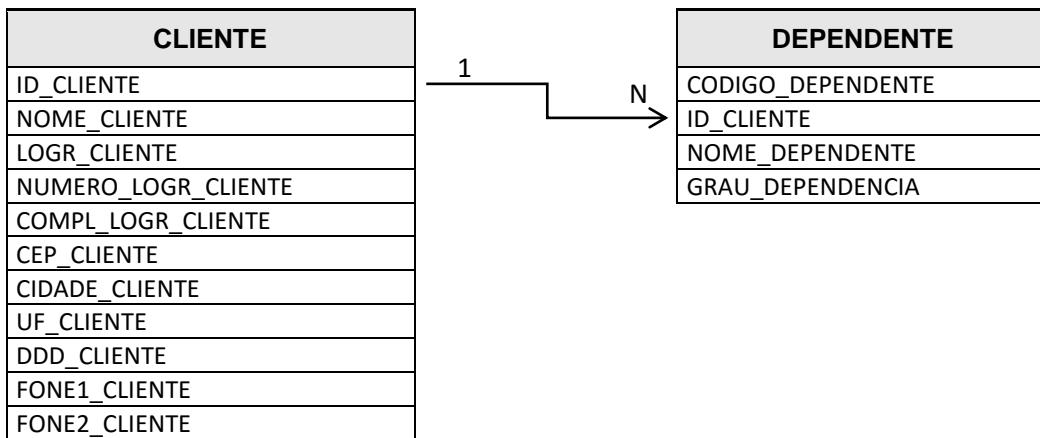


Tabela 23. Normalização na 1FN aplicada sobre a tabela CLIENTE

Fonte: Autor.

2.21.3 Segunda forma Normal (2FN)

De acordo com Elmasri e Navathe (2011) a segunda forma normal (2FN) é baseada no conceito de dependência funcional total. Uma relação está na 2FN se cada atributo não-chave for total e funcionalmente dependente da chave primária. Uma dependência funcional caracteriza-se como uma dependência parcial se algum atributo da chave for desconsiderado e a dependência ainda se mantiver.

Conforme menciona Churcher (2007), a definição da 1FN e 2FN obriga-nos a conhecer a chave primária da tabela que estamos avaliando, quando uma tabela está na 1FN e possui apenas uma coluna como chave primária, não há dependências parciais entre seus atributos não-chave, obviamente que é impossível uma coluna depender de uma parte da chave primária quando a chave primária possui apenas uma coluna, ou seja, não é uma chave composta, desta forma, toda relação que está na 1FN e que possui apenas um atributo como chave primária já está na 2FN.

De acordo com Elmasri e Navathe (2011), quando a chave primária da tabela é a combinação de dois ou mais atributos, ou seja, é uma chave primária composta, precisamos avaliar a dependência funcional total dos atributos não-chave em relação a todos os campos que compõem a chave primaria, desta forma a tabela estará na 2FN se estiver na 1FN e precisarmos de todos os campos da chave primaria composta para determinarmos os valores dos campos não-chave. Se algum campo depender somente de parte da chave composta então este campo deve pertencer a outra tabela.

Por exemplo, considere a tabela LOTEAMENTO, a chave primária da tabela é a combinação dos atributos CODIGO_LOTE e CODIGO_LOTEAMENTO. Note que esta tabela já está na primeira forma normal (1FN), isso porque se solicitarmos informações pela identificação do CODIGO_LOTE e do CODIGO_LOTEAMENTO, (por exemplo 2 e 5), teremos valores exclusivos para todos os outros campos não-chave, conforme mostrado na tabela 25.

LOTEAMENTO

CODIGO LOTE	CODIGO LOTEAMENTO	AREA LOTE	PRECO LOTE	NOME LOTEAMENTO	CIDADE LOTEAMENTO	UF LOTEAMENTO
2	1	100	40000	JARDIM DAS CAMELIAS	GUARATUBA	PR
2	2	400	40000	JARDIM DOS IPES	MATINHOS	PR
2	3	300	30000	JARDIM DAS PALMEIRAS	PRAIA DE LESTE	PR
2	4	1200	50000	JARDIM DAS TULIPAS	ITAPOA	SC
2	5	300	60000	JARDIM DAS ROSAS	FLORIANOPOLIS	SC
4	1	100	40000	JARDIM DAS CAMELIAS	GUARATUBA	PR
4	2	400	40000	JARDIM DOS IPES	MATINHOS	PR
4	3	300	30000	JARDIM DAS PALMEIRAS	PRAIA DE LESTE	PR
4	4	1000	40000	JARDIM DAS TULIPAS	ITAPOA	SC
4	5	300	60000	JARDIM DAS ROSAS	FLORIANOPOLIS	SC

Tabela 24. Tabela na 1FN para aplicação da 2FN

Fonte: Autor

CODIGO_LOTE	CODIGO_LOTEAMENTO	AREA_LOTE	PRECO_LOTE	NOME LOTEAMENTO	CIDADE LOTEAMENTO	UF LOTEAMENTO
2	5	300	60000	JARDIM DAS ROSAS	FLORIANOPOLIS	SC

Tabela 25. Valores exclusivos para atributos não chave em uma relação na 1FN
Fonte: Autor.

Para testar a tabela LOTEAMENTO precisamos validar a dependência funcional total dos campos não chave em relação à chave composta por CODIGO_LOTE e CODIGO_LOTEAMENTO. Observe que a tabela LOTEAMENTO viola a definição geral da segunda forma normal, isso porque embora possamos descobrir o valor de todos os campos não-chave por saber a chave primária, não precisamos de ambos os campos da chave primária para fazer isso, se quisermos obter o valor do atributo AREA_LOTE, precisamos saber qual o CODIGO_LOTE e o CODIGO_LOTEAMENTO, no entanto, se quisermos saber apenas o nome de um loteamento, só precisamos conhecer o valor do CODIGO_LOTEAMENTO, note que NOME_LOTEAMENTO, CIDADE_LOTEAMENTO e UF_LOTEAMENTO são parcialmente dependentes da chave primária composta, ao contrário de AREA_LOTE e PRECO_LOTE que são totalmente dependentes da chave primaria composta.

Observe abaixo na tabela 26, para saber a área do lote 2, fica a questão: “De qual loteamento? ”, ou seja, não é possível saber a área do lote 2 conhecendo apenas o CODIGO_LOTE. Já na tabela 27, para saber o nome de um loteamento basta apenas o informar o valor do atributo CODIGO_LOTEAMENTO.

CODIGO_LOTE	CODIGO_LOTEAMENTO	AREA_LOTE	PRECO_LOTE	NOME_LOTEAMENTO	CIDADE_LOTEAMENTO	UF_LOTEAMENTO
2	1	100	40000	JARDIM DAS CAMELIAS	GUARATUBA	PR
2	2	400	40000	JARDIM DOS IPES	MATINHOS	PR
2	3	300	30000	JARDIM DAS PALMEIRAS	PRAIA DE LESTE	PR
2	4	1200	50000	JARDIM DAS TULIPAS	ITAPOA	SC
2	5	300	60000	JARDIM DAS ROSAS	FLORIANOPOLIS	SC

Tabela 26. Área do lote 2 conhecendo apenas o Código_lote
Fonte: Autor.

CODIGO_LOTE	CODIGO_LOTEAMENTO	AREA_LOTE	PRECO_LOTE	NOME_LOTEAMENTO	CIDADE_LOTEAMENTO	UF_LOTEAMENTO
2	5	300	60000	JARDIM DAS ROSAS	FLORIANOPOLIS	SC
4	5	300	60000	JARDIM DAS ROSAS	FLORIANOPOLIS	SC

Tabela 27. Tabela com atributo caracterizando uma dependência parcial
Fonte: Autor.

Observe que a 2FN está ligada a ocorrência de chaves primárias compostas e a relação de dependência dos atributos não-chave em relação a esta chave, se uma tabela não está na segunda forma normal, removemos os campos não-chave que não são dependentes de toda a chave primária, criamos outra tabela com esses campos e a parte da chave primária na qual eles dependem será o campo chave da nova tabela (Elmasri e Navathe 2011).

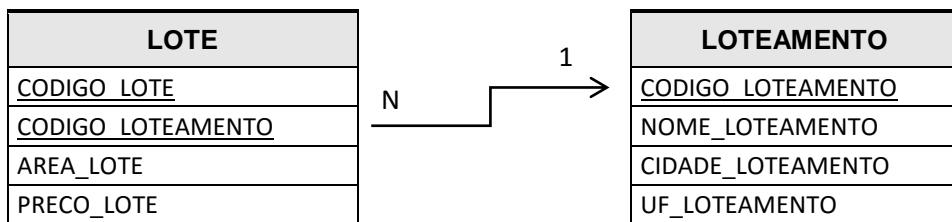


Tabela 28. Uma tabela normalizada na 2FN

Fonte: Autor.

2.21.4 Terceira forma normal (3FN)

Conforme Elmasri e Navathe (2011), a terceira forma normal é baseada no conceito de dependência funcional transitiva. Uma dependência funcional em um esquema de relação é uma dependência transitiva se houver um conjunto de atributos não chave que sejam diretamente dependentes de outros atributos não chave da tabela.

Heuser (2008), afirma que na definição de colunas de uma tabela pode haver casos em que uma coluna não seja diretamente dependente da chave primária, ou de parte dela, mas sim, dependente de outra coluna da entidade, coluna esta que também não é chave primária, quando isso ocorre, o autor afirma que há uma dependência funcional transitiva.

Observe a tabela 29, o campo ID_CLIENTE é a chave primária, esta tabela está na primeira forma normal porque não possui atributos multivalorados e todas os outros atributos não chave dependem da chave primária, se eu sei um valor de ID_CLIENTE, eu posso saber um valor único para todos os outros campos. Se quisermos saber a profissão de um cliente, basta informarmos um valor para ID_CLIENTE (por exemplo, "1009"), isso será suficiente para obtermos a profissão ("Telefonista") em uma única linha retornada.

E na segunda forma normal? Será que esta tabela está na 2FN?. Observe que a chave primária não é composta, a chave primária é apenas um campo e nada pode depender de parte da chave, portanto, sim, está na 2 FN.

Embora esta tabela esteja normalizada na 2FN ainda há um problema, repare que na situação em que essa tabela está, o valor de DESC_PROFESSAO é determinado por mais de um campo e este campo não é um campo que faz parte da chave primária. Se sabemos que o valor de ID_CLIENTE, que é a chave primária, é 1009 então sabemos que o valor do atributo DESC_PROFESSAO é “Telefonista”. No entanto, se sabemos que o valor de COD_PROFESSAO é 125, também obtemos que a profissão do cliente é Telefonista. Há dois campos diferentes que determinam o valor do atributo profissão e neste caso dizemos que há uma dependência transitiva, que, conforme já mencionado, é quando o valor de um campo não-chave depende do valor de outro campo não-chave, por este motivo dizemos que esta tabela não está na terceira forma normal.

ID_CLIENTE	NOME	DT_NASC	SEXO	COD_PROFESSAO	DESC_PROFESSAO
1009	Reginaldo Manuel Barreto	06/02/80	M	125	TELEFONISTA
1012	Afonso Alexandre Leao	25/07/91	M	153	CABELEIREIRO
1124	Laryssa Keller Freitas	03/11/94	F	122	ASSISTENTE DE LABORATORIO
1134	Luis Santiago Bernardes	02/08/80	M	106	SUPERVISOR DE VENDAS
1135	Lucas Gabriel Amaral	21/10/74	M	107	ESTETICISTA
1136	Raquel Sofia Paes	25/09/73	F	142	FARMACEUTICO
1162	Laura Patricia Barcellos	14/08/65	F	154	VENDEDOR
1174	David Luis Ferrari	04/03/69	M	195	ANALISTA DE MARKETING
1198	Nuna Alexandra Galisteu	27/04/75	F	154	VENDEDOR
1200	Madalena Maria Junqueira	03/02/90	F	195	ANALISTA DE MARKETING

Tabela 29. Tabela com uma dependência funcional transitiva
Fonte: Autor.

Conforme Churcher (2007), se uma tabela não está na terceira forma normal, deve-se remover os campos não-chave que são dependentes de outro campo que não seja a chave primária e criar uma tabela com estes campos. As duas tabelas resultantes estarão na 3FN.

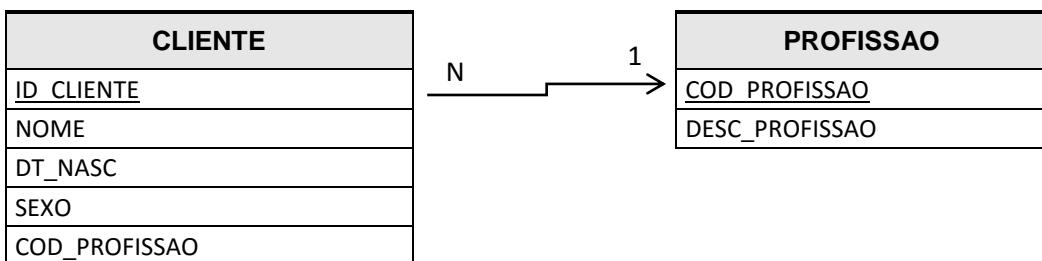


Tabela 30. Uma tabela normalizada na 3FN
Fonte: Autor.

De acordo com Heuser (2008, p. 147), “uma tabela encontra-se na 3FN quando, além de estar na 2FN, toda coluna não chave depende diretamente da chave primária, isto é, quando não há dependências funcionais transitivas ou indiretas”.

2.22 O ponto de partida

Conforme relata Churcher (2007), uma grande quantidade de trabalho é necessária para garantir que o banco de dados seja conveniente para os utilizadores. Isto significa projetar uma interface de usuário com uma lógica clara, boas instalações de entrada, capacidade para encontrar rapidamente os dados para editar ou apagar, realizar consultas precisas, oferecer recursos para relatórios, capacidade de importação e exportação de dados, e boas instalações de manutenção, como backups e arquivamento. Não subestime o tempo e os conhecimentos necessários para completar uma aplicação útil até mesmo para o menor banco de dados.

Como vimos anteriormente um dos critérios de classificação de um SGBD é quanto à sua finalidade que pode ser de uso geral ou de uso específico.

De acordo com Elmasri e Navathe (2011), um SGBD é de uso específico quando ele é projetado e construído para uma determinada aplicação, estes SGBDs são chamados de sistemas de processamento de transação on-line (OLTP – Online Transaction Processing). Neste ambiente a aplicação envia a solicitação, o SGBD executa a instrução e retorna o resultado para o usuário final. Por exemplo um sistema de vendas onde você digita um documento e a aplicação retorna os dados cadastrados caso estes já existam.

Conforme Elmasri e Navathe (2011), estes bancos de dados devem executar com eficiência processamentos de transação on-line (OLTP), que inclui inserções, atualizações e exclusões, enquanto também tem suporte para requisitos de consulta de informação. Ninguém quer um sistema de vendas lento, onde o cadastramento de um cliente ou atualização de dados cadastrais do mesmo torne-se um martírio.

De acordo com Kimball (2002), o objetivo e as características de um ambiente de armazenamento de dados é fornecer dados em um formato facilmente comprehensível para a comunidade do negócio, neste sentido nem o mais moderno sistema OLTP pode oferecer relatórios operacionais adequados, estes relatórios são caracterizados por um conjunto limitado de consultas fixas executadas na própria aplicação, por este motivo há os bancos de dados operacionais.

Os bancos de dados operacionais geralmente são bancos de dados relacionais, possuem estrutura de dados normalizada e são otimizados para processar consultas e transações que lidam com exclusões, inserções ou atualizações de registros nas tabelas. Geralmente armazenam informações do ambiente OLTP e que são convenientes ao negócio.

De acordo com Kimball (2002), este armazenamento de dados operacional, também chamado de ODS, (Operational Data Store), é mais comumente implementado para entregar relatórios operacionais e desempenhar o papel de fonte de dados para um data warehouse. Os bancos de dados operacionais geralmente são projetados para atender as necessidades dos procedimentos regulares sobre o uso dos dados do ambiente OLTP. Por exemplo, caso você queira uma base de dados que contenha todos os clientes de sua empresa que fazem aniversário em um determinado mês.

Conforme Inmon (2007), o ODS é um ambiente híbrido em que os dados de aplicações são transformados, geralmente por ETL, em um formato integrado. Uma vez colocado em ODS, os dados são então disponibilizados para o processamento de alto desempenho, incluindo processamento de consultas e atualizações. De certa forma o ODS protege o data warehouse clássico da sobrecarga de processamento de transações de dados que vem com a realização de processamentos e atualizações em tempo real do ambiente OLTP.

De acordo com Elmasri e Navathe (2011) um data warehouse também é um sistema de informações, bem como um sistema de suporte, no entanto, enquanto os bancos de dados tradicionais são transacionais, os data warehouses têm a

característica distintiva de servir principalmente para aplicações de apoio à decisão, são bancos de dados otimizados para recuperação rápida de dados, e não para processamento de transação de rotina. Oferecem acesso a dados para análises complexas, descoberta de conhecimento e tomada de decisão, são bancos de dados que dão suporte a aplicações OLAP, DSS e aplicações de mineração de dados.

Muito se divulga data warehouses com a finalidade de armazenagem de dados, no entanto, conforme Elmasri e Navathe, (2011), data warehouses são muito distintos dos bancos de dados tradicionais em sua estrutura, funcionamento, desempenho e finalidade. Data warehouses são projetados exatamente para dar suporte à extração, processamento e apresentação eficientes para fins analíticos e de tomada de decisão.

“Em comparação com os bancos de dados tradicionais, os data warehouses em geral contêm quantidades muito grandes de dados de várias fontes, que podem incluir bancos de dados de diferentes modelos de dados e, às vezes, arquivos adquiridos de sistemas e plataformas independentes”. Elmasri e Navathe (2011, p. 721).

Característica	OLTP	Data Warehouse
Propósito	Suporte a processamento de Transações	Apoio à tomada de decisão
Fonte de Dados	Transações de Negócios	Múltiplos arquivos, bancos de dados, dados internos e externos à empresa
Acesso a dados permitido aos usuários	Leitura e escrita	Leitura apenas
Modo de acesso a dados primários.	Atualização simples de banco de dados e consulta.	Consultas de dados simples e complexos com o aumento do uso de mineração de dados para reconhecer padrões nos dados
Principal modelo de dados empregado	Relacional	Relacional ou Dimensional
Nível de Detalhe	Transações Detalhadas	Muitas vezes dados resumidos
Disponibilidade de dados históricos	Muito limitado, geralmente, algumas semanas ou meses	Vários Anos
Processo de atualização	Online - Os dados são atualizados quando transações comerciais ocorrem	Processo periódico, uma vez por semana ou uma vez por mês
Facilidade de processo	Rotineira e fácil	Complexa deve combinar dados de muitas fontes; os dados devem entrar através de um processo de limpeza de dados

Problemas de integridade de dados	Cada transação deve ser intimamente editada.	Grande esforço para "limpar" e integrar dados de múltiplas fontes
--	--	---

Tabela 31. Comparação entre OLTP e Data Warehouse
Fonte: Adaptado de Stair (2009, p. 206).

2.23 Data warehouses

Um armazém de dados, também conhecido como data warehouse, é um banco de dados destinado ao suporte a sistemas de apoio a decisão, cujas informações são armazenadas em estruturas lógicas dimensionais possibilitando o seu processamento analítico através de ferramentas especiais (OLAP e data mining).

De acordo com Inmon (2005), um data warehouse é uma coleção de dados integrados, orientados por assunto, concebido para suportar sistemas de apoio à decisão, onde cada unidade de dados é relevante para algum momento no tempo. O armazém de dados contém dados atômicos e dados levemente resumidos.

Conforme Kimball (2002), um data warehouse é uma cópia dos dados do ambiente de transações especificamente estruturados para consultas e análises.

Stair (2009), relata ainda que um data warehouse é um banco de dados que contém informações de negócios a partir de muitas fontes na empresa, cobrindo todos os aspectos da organização e voltado principalmente para apoiar a tomada de decisão.

Conforme Elmasri e Navathe (2011), diferentemente da maioria dos bancos de dados transacionais, que são projetados para atender as necessidades dos sistemas de processamento de transações, um data warehouse é projetado especificamente para apoiar o processo de tomada de decisão da gestão, costumam apoiar análises de série temporal e de tendência, ambas exigindo mais dados históricos do que geralmente é mantido nos bancos de dados transacionais.

Inmon (2002), afirma que o data warehouse é orientado para as principais áreas da corporação, geralmente são definidas no modelo de dados corporativos de alto nível. Assuntos típicos incluem Cliente, Produto, Operação, Reclamação e Finanças. Cada grande área de assunto é fisicamente implementada como uma série de tabelas relacionadas no data warehouse onde cada área de assunto pode conter inúmeras ou até mesmo centenas de tabelas relacionadas.

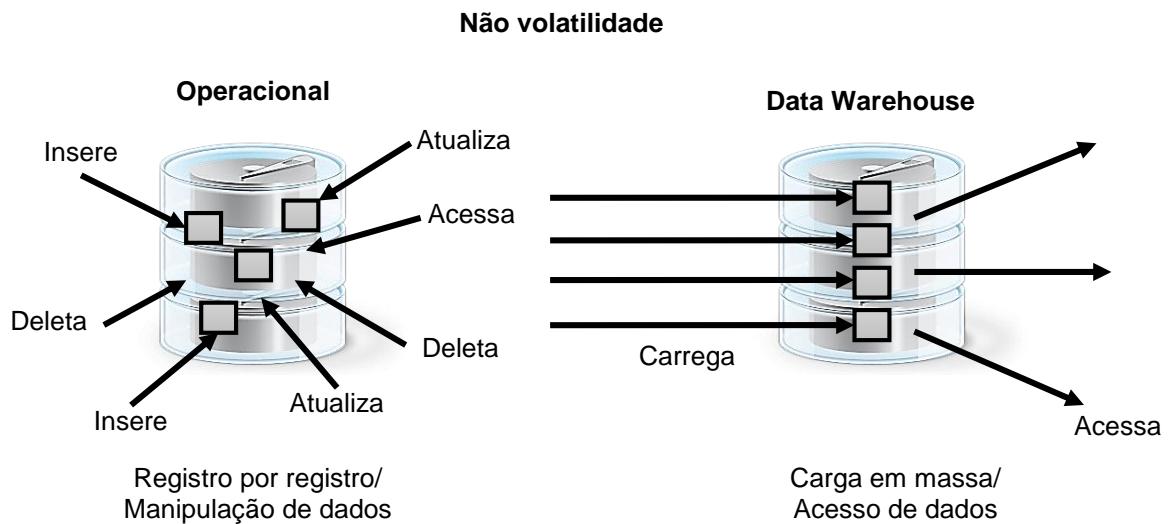
Segundo Inmon (2005. p. 30), “de todos os aspectos de um data warehouse, a integração é o mais importante”.

Essa integração de informações é uma característica marcante em um data warehouse, isso porque os dados são alimentados a partir de inúmeras fontes através de processos de extração, transformação e carga (ETL), nestes processos os dados são convertidos, formatados, ordenados, resumidos e trabalhados de acordo com a necessidade da modelagem. O resultado é que uma vez que os dados residem no data warehouse forma-se uma única imagem corporativa, fornecendo aos usuários de negócios uma visão multidimensional dos dados que eles precisam para analisar as condições de administração.

Em comparação com os bancos de dados operacionais, as informações no data warehouse mudam com muito menos frequência, por este motivo dizemos que um DW é não volátil, esta é outra importante característica de um Data warehouse. É normal que ao serem acessados e manipulados os dados sejam atualizados no ambiente operacional, no entanto, os dados no armazém de dados apresentam um conjunto diferente de características. Por exemplo, se um cliente não mora mais em um endereço, obviamente que seus dados precisam ser atualizados, neste caso, o atributo endereço deste cliente no banco de dados operacional será alterado, já no Data warehouse, o novo endereço será adicionado, guardando desta forma o histórico de locais em que o cliente residiu.

Conforme Inmon (2005), o data warehouse geralmente, mas nem sempre, é carregado em massa e acessado, mas não é atualizado (no sentido geral). Em vez disso, quando o dado no data warehouse é carregado, ele é carregado em um instantâneo, um formato estático. Quando ocorrem alterações posteriores, um novo registro instantâneo é gravado. Ao fazê-lo, um registro histórico de dados é mantido no armazém de dados. No nosso exemplo teremos o endereço atual e os endereços que nosso cliente residiu ao longo de algum período no tempo.

A última característica do data warehouse, citada por Inmon (2005), é que ele é variante no tempo. A variação no Tempo implica que cada unidade de dados no data warehouse existe em algum momento no tempo. Em alguns casos, um registro possui a data e hora marcada, em outros, um registro tem uma data de transação. Mas, em todos os casos, existe alguma forma de marcação de tempo para mostrar o momento no tempo durante o qual o registro foi disposto.



Variante no Tempo

- Operacional**

Data Warehouse
- Variante no Tempo – atual de 60 a 90 dias.
 - Atualização de registros.
 - Estrutura chave pode ou não conter elemento de tempo.
 - Horizonte Temporal – 5 a 10 anos.
 - Sofisticada fotografia de dados.
 - Estrutura chave contém um elemento de tempo.

Figura 30. Características de um Data warehouse
Fonte: Adaptado de Inmon (2005, p. 32).

2.24 Granularidade de Dados

De acordo com Inmon (2005), o aspecto mais importante de design de um data warehouse é a questão de granularidade. Na verdade, a questão da granularidade permeia toda a arquitetura que envolve o ambiente de data warehouse. Granularidade refere-se ao nível de detalhe ou resumo das unidades de dados em um data warehouse ou data mart, quanto mais detalhe existir, menor será o nível de granularidade e quanto menos detalhe existir, mais alto o nível de granularidade.

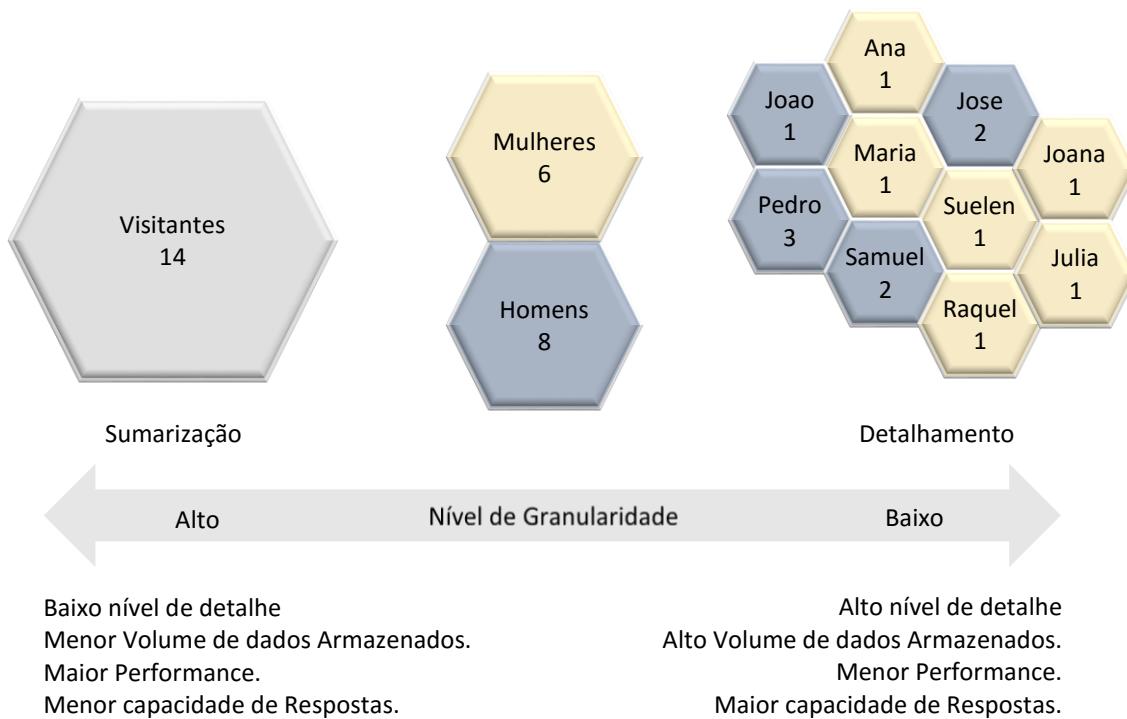


Figura 31. Níveis de Granularidade
Fonte: Autor.

Conforme relata Elmasri e Navathe (2011), na construção de um data warehouse, os responsáveis deverão ter uma visão ampla do uso antecipado do armazém, não há como prever todas as consultas ou análises possíveis, porém o projeto deve suportar especificamente as consultas ocasionais que serão aplicadas sobre o ambiente. O grande desafio é manter um equilíbrio entre eficiência no processamento de consultas e suporte aos níveis de requisitos das consultas, não podemos esquecer que um data warehouse é otimizado para acesso com base nas necessidades de um tomador de decisão.

Neste sentido, como afirma Sauter (2010), os projetistas precisam firmar um compromisso com os tomadores de decisão bem como alinhar as necessidades para que o ambiente dê flexibilidade suficiente para ver o problema em níveis distintos e controlar o escopo de processamento do banco de dados. Definir uma granularidade baixa, ou seja, manter um alto nível de detalhe, aumenta o volume de dados, exigindo maior capacidade de armazenamento e maior tempo de processamento. Além disso a escolha de um baixo nível de granularidade aumenta as chances da incidência de fatores irrelevantes ou inapropriados para os tomadores de decisão. Por outro lado, ao aumentar os níveis de granularidade, a compactação de dados é maior e torna-se predominante o desempenho de resposta do ambiente, no entanto, a capacidade dos decisores para responder perguntas diminui. Ou seja, a granularidade ideal depende da necessidade do projeto.

Conforme cita Inmon (2005), quanto menor o nível de granularidade, maior será a versatilidade das consultas a serem emitidas, por outro lado, quanto maior o nível de granularidade menor será a versatilidade das consultas, isso porque a granularidade define o menor nível de detalhe em que uma análise pode chegar.

2.25 Data Mart

Um data mart é um subconjunto de um data warehouse, conforme Inmon (2007), é o lugar onde o usuário final tem acesso direto e controle de seus dados analíticos. É moldado em torno de um conjunto de expectativas gerais dos utilizadores para a forma como os dados devem ser vistos, isso porque ao invés de armazenar todos os dados corporativos, data marts contêm um subconjunto de dados para um único aspecto da empresa, isso faz com que muitas vezes os utilizadores formem grupos de usuários por departamento, por exemplo, Finanças tem o seu próprio data mart, marketing tem um data mart, vendas tem um data mart, e assim por diante.

O data mart é normalmente implementado usando uma tecnologia diferente da empregada no data warehouse, geralmente utiliza-se de softwares que permitem aos usuários explorar os dados a partir de um variado número de perspectivas.

Conforme cita Inmon (2002), os data marts são SGBDs multidimensionais, fornecem um sistema de informação com uma estrutura que permite que a

organização tenha um acesso muito flexível aos dados, para separar e filtrar dados de várias maneiras, e de explorar de forma dinâmica a relação entre os dados resumidos e detalhados. SGBDs multidimensionais, oferecem flexibilidade e controle para o usuário final, e como tal, eles se encaixam bem em um sistema de suporte a decisão (DSS).

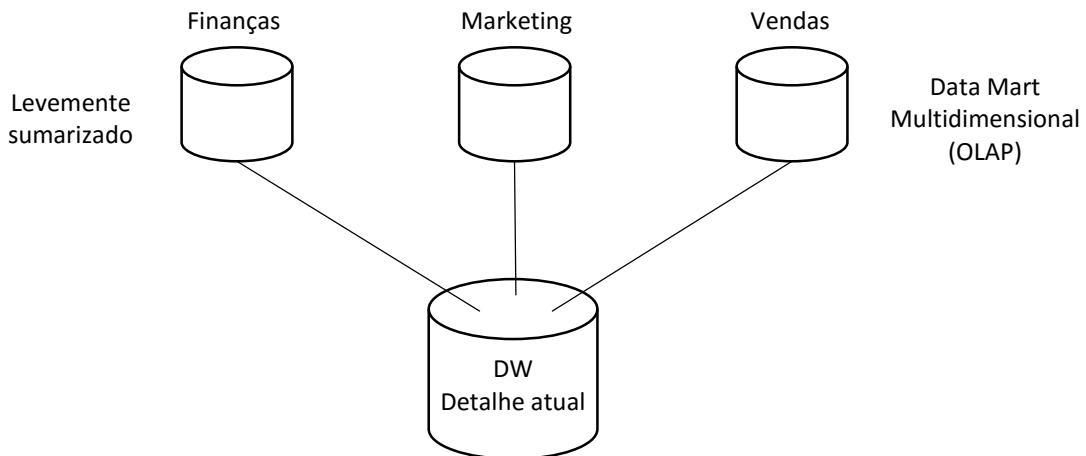


Figura 32. Interação entre um armazém de dados e Data Marts
Fonte: Adaptado de Inmon (2005. p. 176).

A figura 32 mostra a estrutura clássica do armazém de dados e como os dados de detalhe atual e dados departamentais ou data marts, (OLAP), se interagem.

Segundo Inmon (2007), a principal fonte de dados para um data mart é o data warehouse. Cada data mart geralmente detém consideravelmente menos dados do que o data warehouse e pode também conter um significante montante de dados resumidos ou agregados, no entanto, embora não seja a melhor prática, data marts também podem conter dados mais detalhados do que os fornecidos pelo armazém de dados, estes dados são provenientes de fontes particulares existentes fora do data warehouse geral, ou mesmo diretamente dos sistemas de origem, dependendo do que é necessário perguntar para responder consultas específicas. Por exemplo, um local de produção pode necessitar de compilar alguns dados especializados e únicos para o processo usado para criar um produto em particular.

Não podemos prever todas as perguntas feitas pelos usuários de negócios, por este motivo, conforme relata Kimball (2002), algumas vezes precisamos fornecer-lhes acesso a dados mais detalhados para que possam estudá-los com base na pergunta do negócio na mão. Dados com o menor nível de detalhe são

praticamente imunes a surpresas ou mudanças. Estes dados resumidos, de maior nível granular, servem para proporcionar um desempenho melhorado para consultas comuns, mas não devem tentar servir como um substituto para os detalhes, os dados de menor nível granular são fundamentais para responder perguntas mais específicas.

Conforme Withee (2010), a estrutura de dados encontrados em uma data mart é diferente da estrutura de dados para qualquer outro data mart. Por exemplo, a estrutura de um data mart de vendas será diferente de um data mart de marketing, obviamente que o nível de detalhamento e as informações contidas neste data mart não são úteis para todos os departamentos como um todo, por este motivo dizer que um data mart é um data warehouse não faz sentido. Diferentemente de um data warehouse, data marts criam estruturas que não são reutilizáveis, exceto por alguém que opera no departamento para o qual ele é otimizado. As estruturas de data mart são tipicamente conhecidas como estruturas multidimensionais, podem ser junção estrela, também chamadas de star schema, ou snow flake, ambas contêm tabelas de fatos e dimensões e são servidas pela tecnologia OLAP.

De acordo com Withee (2010), dados de data warehouse devem ser selecionados, acessados, e depois reformulados para atender às necessidades do data mart. Muitas vezes os dados de um data mart residem em cubos. Cubo é o nome que se dá para uma plataforma de banco de dados com uma estrutura tridimensional para processamento analítico (OLAP), é formado por uma tabela Fato e suas tabelas de dimensão. Os cubos precisam ser formados, e muitos cálculos diferentes precisam ser executadas nos dados detalhados que residem no data warehouse. Em suma, um processo não-trivial ocorre à medida que os dados são transmitidos a partir de um mundo normalizado para um mundo multidimensional. Uma das questões importantes aqui é quantos dados deverão ser acessados e como geralmente o processo de atualização será executado.

Conforme Elmasri e Navathe (2011), data marts em geral são voltados para um subconjunto da organização e possuem um foco mais acentuado em colocar informações de negócios direcionadas para os tomadores de decisão.

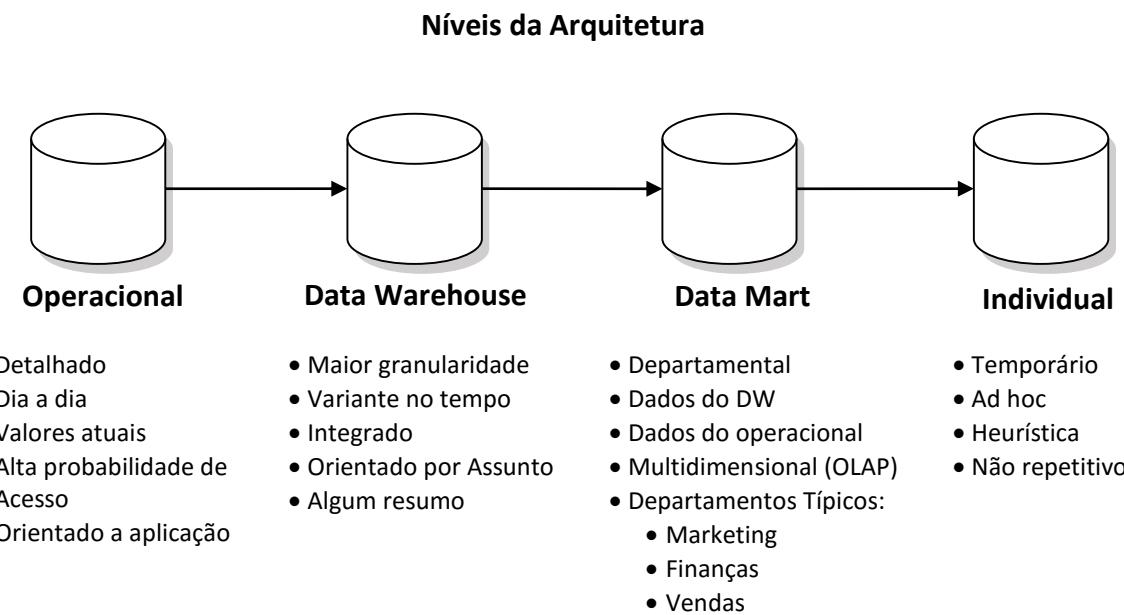


Figura 33. Níveis de Arquitetura de um BI System
Fonte: Adaptado de Inmon (2002. P. 16).

O dado detalhado alojado em um armazém de dados fornece uma fonte muito robusta e conveniente de dados para os data marts, também conhecidos como bancos de dados multidimensionais ou bancos de dados OLAP, observe na figura 34, os tipos de consultas para as quais os diferentes níveis de dados podem ser utilizados.

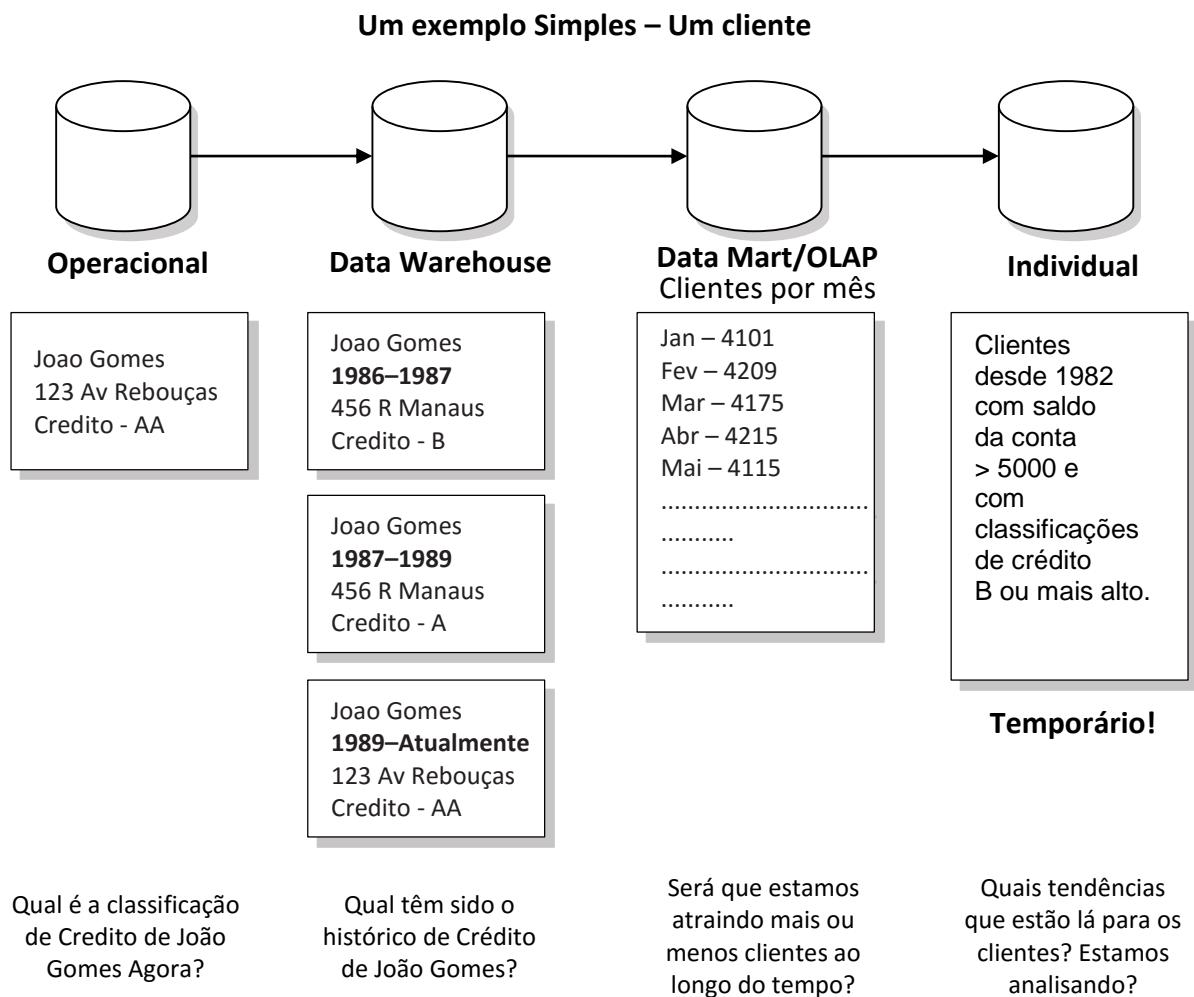


Figura 34. Tipos de consultas para diferentes níveis de dados
Fonte: Adaptado de Inmon (2002. P. 18).

2.26 DSS - Decision Support System

Segundo Inmon (2005), em meados da década de 1970, o surgimento do processamento de transações online (OLTP), abriu possibilidades de acesso a mais dados, tantos quantos fossem possíveis, abrindo novas perspectivas integrais para o negócio e processamento. O computador podia então ser usado para tarefas não possíveis anteriormente, incluindo sistemas de reservas de condução, sistemas de caixa de banco, sistemas de controle de produção, e assim por diante. Na década de 1980, mais as novas tecnologias, tais como PCs e linguagens de quarta geração (4GLs), começaram a surgir. O usuário final começou a assumir um papel anteriormente insondável, controlar diretamente os dados e sistemas, um papel anteriormente reservado para o processador de dados. Com PCs e tecnologia 4GL

veio a noção de que mais poderia ser feito com os dados do que simplesmente o processamento de transações on-line. MIS (sistemas de informação de gestão), como era chamado nos primeiros dias, também poderia ser implementado. Hoje também conhecido como DSS, o MIS foi usado para processamento de dados com a finalidade de orientar as decisões de gestão.

Conforme Sauter (2010), um DSS, do inglês Decision Support System, é um sistema baseado em computador que suporta análises para auxiliar o tomador de decisão na organização da informação e modelagem de resultados. Ajudam os principais tomadores de decisão com dados de nível mais alto em decisões complexas e importantes. Normalmente DSS envolve a análise de muitas unidades de dados de forma heurística. Como regra geral, o processamento do DSS não envolve a atualização dos dados.

2.27 Tecnologia Multidimensional OLAP

De acordo com Withee (2010), em 1993, Edgar F. Codd, conhecido como o pai do banco de dados moderno, propôs um novo tipo de projeto de banco de dados, o Online Analytical Processing (OLAP). Em essência, OLAP armazena dados em uma forma multidimensional, é projetado para rápida recuperação e análise de dados. Esses bancos de dados são otimizados para análises, isso porque você pode fatiar os dados e analisá-los de diferentes ângulos. Por exemplo, você pode ver o valor das vendas totais dividido por loja, por período de tempo, e por caixa e, em seguida, saltar para olhar a margem de lucro dividido pelo produto, grupo de produtos, compra, e hora do dia.

OLAP é um conceito que se refere a aplicações que permitem efetuar de forma rápida e partilhada, a análise de informação multidimensional, originária de diferentes fontes de dados, geralmente utilizadas por usuários finais na geração de consultas e relatórios. As tecnologias OLAP permitem reestruturar os dados de uma base de dados relacional em uma perspectiva multidimensional, possibilitando consultar essa estrutura, com vista a detectar tendências e a tirar conclusões.

Conforme cita Sudarshan e Silberschatz (2010), um sistema de processamento analítico online (OLAP) é um sistema interativo que permite que um analista veja diferentes resumos de dados multidimensionais. A palavra on-line

indica que um analista deve ser capaz de solicitar novos resumos e obter respostas on-line, em poucos segundos, e não deve ser forçado a esperar por um longo tempo para ver o resultado de uma consulta.

De acordo com Inmon (2005, p. 175):

Uma das tecnologias frequentemente discutidas no contexto do armazém de dados é o processamento de bancos de dados multidimensional, às vezes chamado de processamento OLAP, no entanto, esta tecnologia é adequada para o processamento de Data Marts e não processamento de Data Warehouses. Apesar de tentador pensar que a tecnologia multidimensional OLAP deva ser a tecnologia de banco de dados para o data warehouse, isso é um erro. As propriedades que fazem um banco de dados multidimensional uma ótima tecnologia para o que ele faz não são as propriedades de importância primordial para o armazém de dados, e as propriedades que são as mais importantes em um data warehouse não são aquelas encontradas em tecnologia de bancos de dados multidimensionais (OLAP).

Os principais fornecedores de software OLAP atualmente incluem Microsoft BI, IBM Cognos, SAP Business Objects, MicroStrategy, SQL Server e Oracle.

Considere algumas diferenças entre aplicações OLAP, sustentadas por um banco de dados multidimensional (Data Mart) e um Armazém de dados na tabela a seguir.

Aplicações OLAP – (Data Mart)	Data Warehouse
Mantém uma ordem de magnitude menor de dados.	Contém grandes quantidades de dados.
Voltado para muito pesado e imprevisível acesso e análise de dados.	Voltado para uma quantidade limitada de acesso.
Detém um curto horizonte temporal de dados.	Contém dados com um horizonte de tempo muito longo (de 5 a 10 anos).
Softwares de Análise	Infra-Estrutura de dados
Dados relevantes para o Departamento	Dados relevantes para a Organização.

Tabela 32. Diferenças entre Data Mart (OLAP) e Data Warehouse
Fonte: Adaptado de Inmon 2005.

2.28 Modelos Multidimensionais

Segundo Elmasri e Navathe (2011), modelos multidimensionais tiram proveito dos relacionamentos inerentes nos dados para preencher os dados em matrizes multidimensionais, chamadas cubos de dados. Estes podem ser chamados de hipercubos, se tiverem mais de três dimensões. Para dados que se prestam à formatação dimensional, o desempenho da consulta nas matrizes multidimensionais pode ser muito melhor do que no modelo de dados relacional.

Uma planilha padrão, composta por linhas e colunas, é uma matriz bidimensional. Um exemplo seria uma planilha de vendas regionais por produto para determinado período. Os produtos poderiam ser mostrados como linhas com as receitas de vendas para cada região compreendendo as colunas.

	Região			
	Reg1	Reg2	Reg3	Reg4
Produto	P123			
	P124			
	P125			
	P126			
	.			
	.			

Figura 35. Um modelo de matriz bidimensional
Fonte: Adaptado de Elmasri e Navathe (2011, p. 723).

Ao acrescentar uma dimensão de tempo, como por exemplo os trimestres fiscais de uma organização, seria produzida uma matriz tridimensional que poderia ser representada usando um cubo de dados onde cada célula teria dados para um produto específico, trimestre fiscal específico e região específica. Os dados podem ser consultados diretamente em qualquer combinação de dimensões, evitando consultas de bancos de dados complexas (Elmasri e Navathe 2011).

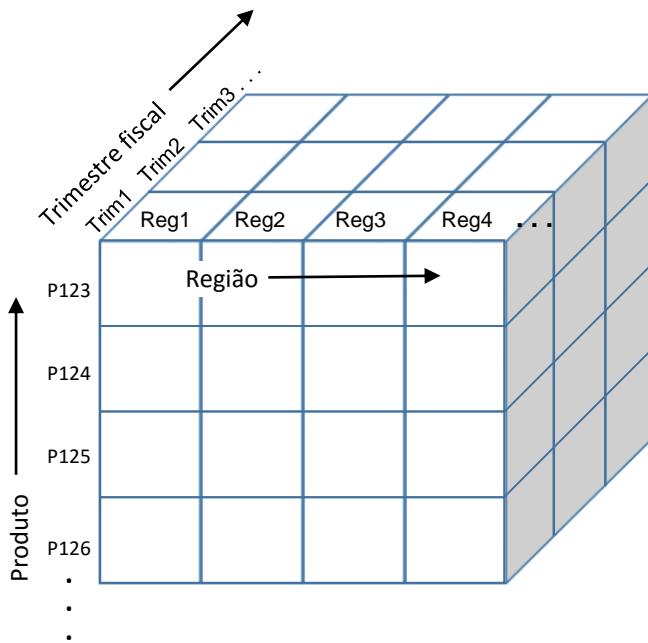


Figura 36. Um modelo de Cubo de dados Tridimensional
Fonte: Adaptado de Elmasri e Navathe (2011, p. 724).

2.28.1 Dimensão

Conforme Reeves (2009), dimensões são as principais categorias de informação de negócios ou agrupamentos para descrever dados de negócios. Dimensões contêm informações utilizadas para restringir consultas, títulos de relatório e definir caminhos de pesquisa. A relação entre os atributos dimensionais cria caminhos de pesquisa e a capacidade de navegar para cima e para baixo em uma hierarquia descreve vários aspectos de uma medida.

Ao descrever a necessidade de comunicar informações por região, por produto, e por trimestre, os atributos a seguir "por" são dimensões. Em nosso cubo, cada um destes é uma dimensão, poderíamos chama-las de dimensão localidade, dimensão produto e dimensão tempo.

Cada dimensão contém todas as informações relativas a um determinado assunto do negócio, por exemplo, clientes, produtos, tempo, e pode ser ligada a uma ou mais tabelas de fatos contendo itens mensuráveis (receitas, custos, números de vendas e assim por diante). O resultado é que o usuário pode consultar em seu data mart informações sobre as receitas de vendas por produto, por região e durante um período específico no tempo, isso de forma dinâmica e rápida.

Segundo Kimball (2002), tabelas de dimensões muitas vezes representam relações hierárquicas no negócio. Por exemplo, em uma tabela dimensão Produto, os produtos podem ser agrupados por marcas e em seguida por categorias. Para cada linha na dimensão do produto, armazenamos descrição da marca e da categoria associada a cada produto. A informação descritiva hierárquica é armazenada de forma redundante, mas o motivo é a facilidade de uso e desempenho da consulta. Também poderíamos armazenar apenas o código da marca na dimensão produto e criarmos uma pesquisa de marca em uma tabela separada. Isso seria chamado de um esquema snowflake ou floco de neve.

2.28.2 Fatos

Conforme Reeves (2009), fatos são as medições fundamentais do negócio. Estes são capturados como informações específicas sobre um evento de negócios ou transação. Eles são medidos, monitorados e rastreados ao longo do tempo. Fatos são os atributos numéricos, são normalmente os valores e as contagens mostradas no corpo de relatórios e que são usados por todos e quaisquer cálculos que possam ser necessários.

Tabelas fatos não necessariamente precisam ser compostas apenas por atributos numéricos, mas sim por atributos quantificáveis, por exemplo, a identificação de uma ordem de venda não é por via de regra um atributo numérico, no entanto a quantidade de ordens de vendas é um item quantificável e, portanto, vai na tabela de fatos. Dependendo do nível de granularidade aplicado na modelagem, por exemplo, se a ordem de venda for o menor nível de detalhe, então na fato haverá um quantificador numérico de 1 para cada produto, a descrição deste produto reside na dimensão Produtos, que é relacionada a esta tabela Fato pela chave primária correspondente.

2.28.3 Hierarquia de Detalhes

De acordo com Kimball (2002), uma hierarquia é composta pelos atributos que se relacionam entre si em uma dimensão, possibilitando ordenar e relacionar as

visões hierárquicas de seus dados em níveis maiores ou menores, dependendo do nível de detalhe no qual eles estão armazenados e disponíveis no ambiente.

Segundo Reeves (2009), cada nível hierárquico é composto por membros, os membros são um subconjunto de uma dimensão e são de domínio do nível hierárquico ao qual pertencem. Por exemplo, em uma dimensão Tempo, o nível hierárquico Ano é composto por quatro membros, Trimestre 1, Trimestre 2, Trimestre 3 e Trimestre 4, onde cada membro é de domínio da hierarquia Trimestre. Consequentemente como O menor nível de detalhe de uma hierarquia é chamado de granularidade.

Observe na imagem abaixo a dimensão tempo, suas hierarquias e seus membros:

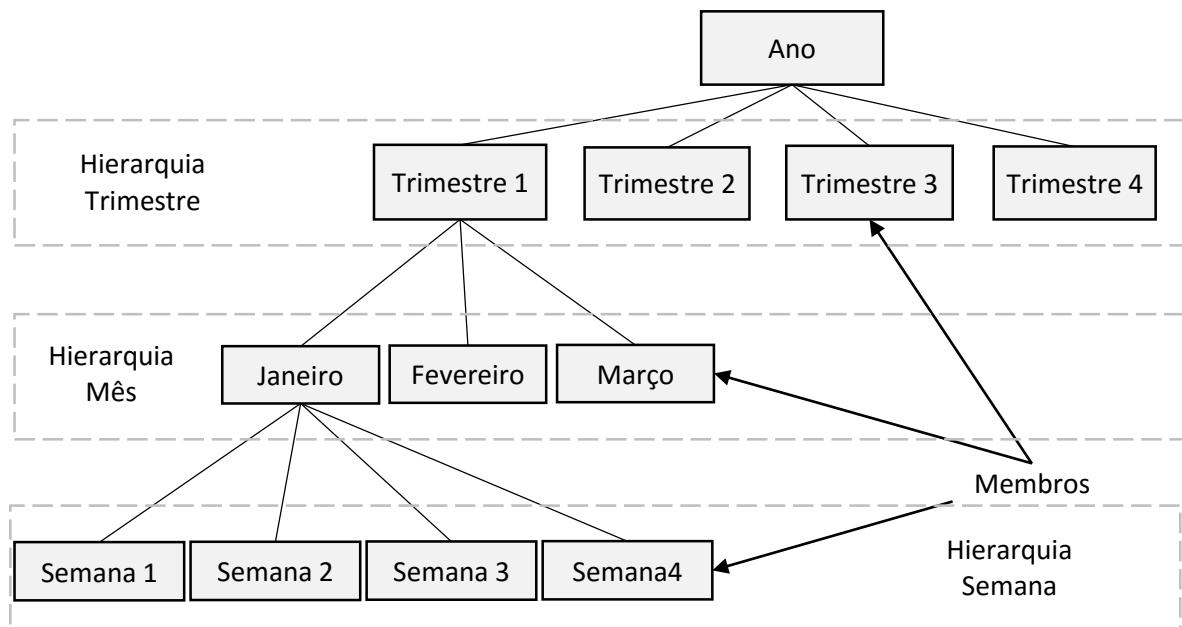


Figura 37. Hierarquias de uma Dimensão Tempo e seus membros

Fonte: Autor.

Conforme Elmasri e Navathe (2011), os modelos multidimensionais atendem prontamente a visões hierárquicas no que é conhecido como exibição roll-up e exibição drill-down. Uma exibição roll-up sobe na hierarquia, agrupando em unidades maiores ao longo de uma dimensão, por exemplo, somando dados semanais por mês, ou somando dados mensais por trimestre, ou somando dados trimestrais por ano. Uma exibição drill-down oferece a capacidade oposta, fornecendo uma visão mais detalhada acerca da visão atual analisada, esta desce até o menor nível de detalhe disponível na tabela de dimensão consultada.

2.29 Star Schema (Modelo Estrela)

Conforme Inmon (2002), o esquema estrela é uma estrutura de dados, onde os dados são desnormalizados para otimizar o acesso aos dados, é a base do projeto de data mart multidimensional.

A estrutura de dados de junção em estrela é assim chamada porque sua representação descreve uma "Estrela" com um centro e diversas estruturas periféricas de dados. No centro da junção em estrela está uma tabela denominada Fato. A tabela de fatos é uma estrutura que contém muitas ocorrências de dados quantitativos, também conhecidos como métricas. Em torno da tabela fato estão as dimensões, compostas por dados qualitativos que descrevem algum aspecto importante da tabela fato.

De acordo com Inmon (2005. P. 137):

A junção em estrela é uma técnica de design de banco de dados que é por vezes erradamente aplicada ao ambiente de data warehouse. A abordagem multidimensional Star Schema é uma abordagem onde o design de banco de dados é baseado nas ocorrências de dados dentro de uma área temática e como esses dados serão acessados. Projeto Star Schema se aplica ao mundo da data marts, não ao mundo dos data warehouses. É um erro construir um data warehouse com um Star Schema, isso porque o data warehouse vai acabar sendo ideal para um conjunto de usuários, em detrimento de todos os outros.

A figura 38 mostra um modelo dimensional em star schema e as dimensões Tempo, Localidade, Cliente, Venda e Produto.

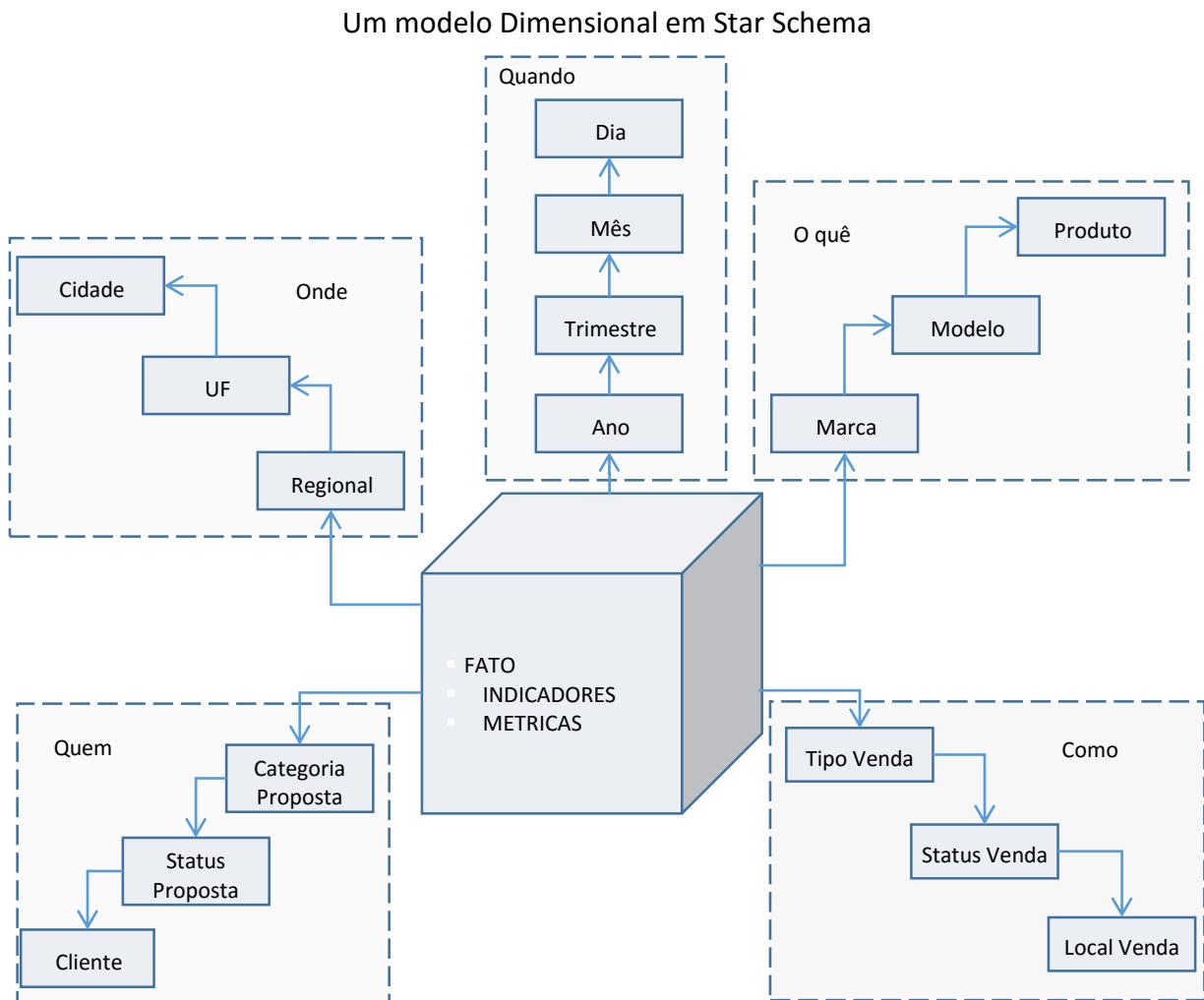


Figura 38. Um modelo Dimensional em Star Schema

Fonte: Autor.

2.30 BI

BI é uma abreviação para Business Intelligence, é todo processo que envolve a habilidade de estruturar, acessar e explorar informações armazenadas em um banco de dados, data warehouse ou data mart, com o objetivo de desenvolver percepções, entendimentos e conhecimentos para produção de um melhor processo de análise e tomada de decisão. É importante não confundir Business Intelligence com ferramenta. Business Intelligence refere-se à interação entre gestão e tecnologia, é um processo produtivo em que a matéria prima é a informação e o produto final o conhecimento. Tudo se baseia, portanto, em planejar, gerir e controlar a informação de maneira a criar e distribuir conhecimento de forma otimizada.

3 ESTUDO DE CASO

Nesta etapa será desenvolvido um estudo prático para que se possa evidenciar de forma prática os conceitos que foram apresentados na conceituação teórica. Muitos dos conceitos apresentados até aqui, alguns deles propositalmente de forma repetitiva, foram para que neste momento possa demonstrar todo o entendimento necessário para a realização da exemplificação prática.

A apresentação prática refere-se a uma empresa fictícia, com dados trabalhados e gerados para atender as características de situações encontradas no real cotidiano empresarial. Primeiramente será apresentada a empresa, destacando quais suas necessidades, e principalmente quais os pontos abordados para suprir suas necessidades. Posteriormente todas as etapas necessárias para o desenvolvimento de um sistema de informações desde o início.

3.1 Características da Empresa

A nossa empresa é uma imobiliária que atua em Curitiba e Região metropolitana na intermediação de venda ou locação de imóveis, tais como casas, apartamentos, salas, escritórios, lotes e terrenos. Pela venda ou locação de um imóvel a imobiliária ganha uma comissão, a qual é devida pelo dono do imóvel. Um percentual da comissão é repassado ao vendedor responsável pela venda ou locação.

O gerente da imobiliária sabe que o mercado apresenta tendências de alta na produção de imóveis, devido a políticas de redução de taxas anuais, aumento de recursos disponibilizados pelos bancos e a força da demanda devido ao déficit habitacional brasileiro. Diante deste cenário, nosso gestor decide que é o momento de apresentar serviços de melhor qualidade ao cliente, com práticas de gestão cada vez mais apuradas, para isso ele toma a decisão de implementar um banco de dados estruturado e criar um sistema de gestão da informação.

Os principais envolvidos no projeto são o gerente e o auxiliar administrativo. Atualmente a empresa possui um sistema de apoio a operações que processa todos os dados gerados a partir das operações de vendas e locações de imóveis, é um sistema contratado e implementado por uma empresa de TI (terceirizada) que cuida

da manutenção desta ferramenta. Basicamente é um terminal onde os dados operacionais são inseridos. As atividades relacionadas com gestão de pessoas, compras, contratações, contas a receber, contas a pagar, bancos e controles financeiros são sustentadas por inúmeras planilhas de dados em Excel, e vários pequenos bancos de dados em Access.

3.2 Proposta de uma solução de BI

Este trabalho irá focar apenas no sistema de informação, no banco de dados e sua arquitetura, com breves referências aos contextos do negócio em que a tecnologia será inserida e supostamente irá apoiar.

O banco de dados proposto será do tipo relacional, utilizando um SGBD da Oracle, será construído a partir dos dados provenientes do sistema transacional da empresa. Após a implementação do banco de dados operacional será implementado um data mart (modelagem multidimensional).

Como a confiabilidade da informação está diretamente ligada à sua utilização, o acesso deverá ser de forma amigável e interativa com o usuário, para isso, será utilizada a ferramenta da TIBCO, o Spotfire, podendo se conectar a um banco de dados relacional ou dimensional, é otimizada para análise de dados e geração de consultas em memória.

Por decisão da Gerência, inicialmente apenas um módulo do sistema da imobiliária será utilizado, o que se refere à venda de loteamentos, os demais módulos serão implementados posteriormente em outra etapa. As principais atividades e objetivos do projeto incluem documentar e divulgar as informações disponíveis da arquitetura atual, validar e estabelecer regras de negócio, identificar as necessidades de dados e fomentar a utilização das novas ferramentas de acesso aos dados reduzindo, desta forma, a utilização de ferramentas intermediárias (Access, Excel). Relatórios analíticos de consumo corporativo, devem ser alimentados exclusivamente pelo data warehouse.

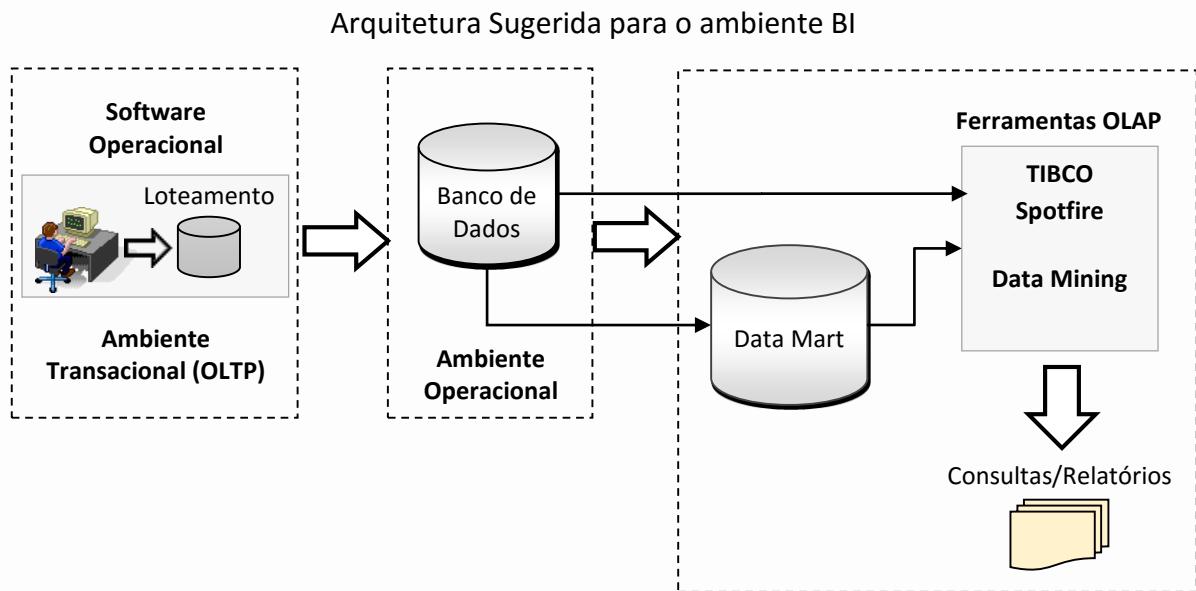


Figura 39. Arquitetura do ambiente BI proposto
Fonte: Autor.

3.3 Análise de Requisitos

Após algumas reuniões com o gerente da imobiliária e alguns dias acompanhando a rotina dos usuários do sistema, foram mapeados o conjunto de requisitos necessários para criação do modelo conceitual do banco de dados.

O sistema operacional da imobiliária é suportado por um software, uma aplicação. Nesta aplicação os dados pessoais do visitante são inseridos no sistema, caso ele tenha interesse em algum imóvel, o vendedor cria uma proposta. Após acordo entre as partes, a proposta é efetivada e gera um número de contrato. O contrato é o compromisso de financiamento, que será acompanhado conforme o pagamento das parcelas do contrato. Um cliente pode ter várias propostas, cada proposta, caso efetivada, cria um contrato distinto e cada contrato pode ter um ou mais itens.

A aplicação é essencial para o processo operacional da imobiliária, no entanto, o gerente deseja aplicar análises sobre seus dados, deste modo ele deseja extrair dados, obter informações e consolidar conhecimentos para minimizar perdas, aumentar seu lucro e principalmente antecipar mudanças no mercado em que atua.

A aplicação possui várias telas ou seções, embora todas sejam interligadas, cada seção é acessada separadamente. Por exemplo, após o usuário entrar no

sistema digitando sua identificação e senha de acesso, dependendo do perfil do usuário, a seção correspondente aparece disponível para transações. Os níveis de privilégio, ou permissões, podem variar desde simples consultas a alterações de dados cadastrais. Cada tela ou seção é respectiva a um assunto, por exemplo, a seção do cliente oferece todas as telas com informações e campos que podem ser consultados ou devem ser preenchidos e alterados com dados do cliente, assim como a tela referente a propostas, que possui todos os dados da proposta bem como seu status e pendências.

Ao todo serão extraídas do banco de dados da aplicação (OLTP) e carregadas para o novo banco de dados (operacional), informações de quatro seções da aplicação, Cliente, Propostas, Contrato e Parcelas. Foi definido que o processo de atualização do banco operacional será através do consumo de arquivos simples de dados, estes arquivos serão gerados em formato de texto, com dados do ambiente OLTP e processados no período da noite a partir das 21:30.

Observe que não estamos aqui mencionando um data warehouse, ou data mart, mas sim um BD operacional que será alimentado com dados do ambiente OLTP, deste modo, as operações de DDL e DML neste banco não impactam a aplicação da imobiliária. Este banco de dados será útil para relatórios operacionais e para suprir o futuro data warehouse e, ou, data marts. Abaixo os arquivos da aplicação com os dados de Clientes, Propostas, Contratos e Parcelas.

FILE_CLNT	FILE_PROP	FILE_CON	FILE_PAR
ID_CLNT	COD_VEND	NUM_CON	NUM_CON
NM_CLNT	NUM_PROP	ID_CLNT	NUM_PAR
LOGR_CLNT	ID_CLNT	NUM_PROP	COD_VEND
NUM_LGR_CLNT	DT_INC_PROP	COD_VEND	NUM_PROP
CPL_LGR_CLNT	VLR_PRIN	DT_EMI	ID_CLNT
CEP_CLNT	VLR_ENT	VLR_TT_CON	DT_VCNOTO
UF_CLNT	QT_PRST	QT_PARS	DT_PGNTO
DDD_CLNT	VLR_DSP_CART	VLR_PAR	VLR_PAR
FN_CLNT	REND_DECL	TAXA_CON	VLR_DSC_PAR
DT_NSC_CLNT	STS_PROP	SEQ_ITEM_CON	VLR_MLT_PAR
GR_INSTR_CLNT	NM_VEND	CD_LTNTO	VLR_TT_PAGO
COD_PROF_CLNT	FNE_VEND	CD_LOTE	
PROF_CLNT	PERC_COMISS	VLR_ITEM_CON	
RENDÀ	DT_ALT	AREA_LOTE	
NM_DPNTE		PRC_LOTE	
GR_DPNC		SIT_LOTE	
DT_INC		NM_LTNTO	
DT_ALT		CID_LTNTO	
		UF_LTNTO	

Tabela 33. Arquivos da Aplicação OLTP
Fonte: Autor.

3.3.1 Metadados (Dicionário de dados) I

Como você deve ter notado, os quatro arquivos gerados possuem os campos de cada seção da aplicação, o nome dos atributos é o mesmo definido no banco de dados transacional (OLTP), e embora possamos fazer suposições, não conseguimos base concreta sobre o que temos disponível, a menos que tenhamos um metadados claro e consistente. Observe abaixo o metadados dos arquivos.

CLIENTE				
Atributo	Descrição Atributo	Domínio	Tamanho	
ID_CLNT	Identificador do Cliente	Número	19	
NM_CLNT	Nome Cliente	Texto	100	
LOGR_CLNT	Logradouro Cliente	Texto	100	
NUM_LGR_CLNT	Número Logradouro Cliente	Número	10	
CPL_LGR_CLNT	Complemento Logradouro Cliente	Texto	20	
CEP_CLNT	Cep do Cliente	Número	8	
UF_CLNT	Uf Cliente	Texto	2	
DDD_CLNT	DDD Cliente	Número	2	
FN_CLNT	Fone Cliente	Número	12	
DT_NSC_CLNT	Data Nascimento do Cliente	Data	-	
GR_INSTR_CLNT	Grau Instrução Cliente	Texto	50	
COD_PROF_CLNT	Código Profissão Cliente	Número	8	
PROF_CLNT	Profissão Cliente	Texto	50	
RENDAS	Renda do Cliente	Número	17	
NM_DPNTE	Nome Dependentes	Texto	100	
GR_DPNC	Grau de Dependência	Texto	20	
CID_CLNT	Cidade Cliente	Texto	50	
COD_CID_CLNT	Código Cidade Cliente	Número	3	
GEN_CLI	Sexo do Cliente	Texto	1	

Tabela 34. Metadados do arquivo de Clientes

Fonte: Autor.

PROPOSTA				
Atributo	Descrição Atributo	Domínio	Tamanho	
COD_VEND	Código Vendedor	Número	9	
NUM_PROP	Número Proposta	Número	10	
ID_CLNT	Identificador do Cliente	Número	10	
DT_INC_PROP	Data Inclusão Proposta	Data	-	
VLR_PRIN	Valor Principal	Número	17	
VLR_ENT	Valor Entrada	Número	17	
QT_PRST	Quantidade Prestações	Número	3	
VLR_DSP_CART	Valor Despesa Cartório	Número	17	
REND_DECL	Renda Declarada	Número	17	
STS_PROP	Status Proposta	Número	1	
NM_VEND	Nome Vendedor	Texto	100	
FNE_VEND	Telefone Vendedor	Número	15	
PERC_COMISS	Percentual De Comissão	Número	6	
DT_ALT	Data Alteração Proposta	Data	-	

Tabela 35. Metadados do arquivo de Proposta

Fonte: Autor.

CONTRATO			
Atributo	Descrição Atributo	Domínio	Tamanho
NUM_CON	Número Contrato	Número	15
ID_CLNT	Identificador do Cliente	Número	10
NUM_PROP	Número Proposta	Número	10
COD_VEND	Código Vendedor	Número	9
DT_EMI	Data Emissão do Contrato	Data	-
VLR_TT_CON	Valor Total do Contrato	Número	17
QT_PARS	Quantidade de Parcelas	Número	3
VLR_PAR	Valor da Parcela	Número	17
TAXA_CON	Taxa Contrato	Número	17
SEQ_ITEM_CON	Sequencia Item Contrato	Número	2
CD_LTNTO	Código Loteamento	Número	5
CD_LOTE	Uf Loteamento	Texto	2
VLR_ITEM_CON	Valor Item Contrato	Número	17
AREA_LOTE	Área do Lote	Número	10
PRC_LOTE	Preço Lote	Número	17
SIT_LOTE	Situação do Lote	Texto	20
NM_LTNTO	Nome do Loteamento	Texto	100
CID_LTNTO	Cidade Loteamento	Texto	30
UF_LTNTO	Uf Loteamento	Texto	2

Tabela 36. Metadados do arquivo de Contrato

Fonte: Autor.

PARCELA			
Atributo	Descrição Atributo	Domínio	Tamanho
NUM_CON	Número Contrato	Número	15
NUM_PAR	Número Parcela	Número	3
COD_VEND	Código Vendedor	Número	9
NUM_PROP	Número Proposta	Número	10
ID_CLNT	Identificador do Cliente	Número	10
DT_VCNTO	Data Vencimento	Data	-
DT_PGNTO	Data Pagamento	Data	-
VLR_PAR	Valor Parcela	Número	17
VLR_DSC_PAR	Valor Desconto Parcela	Número	17
VLR_MLT_PAR	Valor Multa Parcela	Número	17
VLR_TT_PAGO	Valor Total Pago	Número	17

Tabela 37. Metadados do arquivo de Parcelas

Fonte: Autor.

3.4 Modelagem conceitual

De acordo com o levantamento de requisitos, e com o dicionário de dados dos arquivos recebidos, podemos criar o modelo conceitual, neste modelo definiremos como os dados serão tratados e guardados no banco de dados. Observe que não trataremos do banco de dados em si, mas sim a forma como as estruturas serão

criadas para armazenar os dados. O modelo conceitual é a base para o modelo lógico.

Obviamente que você já tenha notado que temos quatro entidades, cliente, proposta, contrato e parcela, no entanto, conforme já visto anteriormente, uma entidade representa um objeto ou conceito do mundo real que possui uma existência independente, nesta afirmação, observando melhor, note que dentro do arquivo de propostas temos também a entidade vendedor. Há mais entidades nos outros arquivos, mas trataremos delas mais adiante. Todas as entidades definidas no modelo conceitual, ou seja, todos os objetos que queremos tratar de uma forma distinta, serão uma tabela no banco de dados físico.

3.4.1 BrModelo

O BrModelo é um software livre, desenvolvido em Delphi, que surgiu como um projeto de conclusão do curso do autor. É um programa gratuito e tem seu código fonte aberto, o autor não oferece suporte, por este motivo ela pode apresentar algumas inconsistências em seu funcionamento, no entanto, nada que impeça sua utilização neste trabalho.

A ferramenta BrModelo facilita a criação de modelos conceituais e lógicos para bancos de dados relacionais, ela implementa o modelo conceitual de Chen e com ela criaremos o nosso DER, onde representaremos nossas entidades, seus atributos, relacionamentos, e utilizando a notação (min, max), a razão de cardinalidade e a restrição de participação.

BrModelo é uma ferramenta freeware voltada para ensino de modelagem em banco de dados relacional com base na metodologia defendida por Carlos A. Heuser no livro "Projeto de Banco de Dados". Esta ferramenta foi desenvolvida por Carlos Henrique Cândido sob a orientação do Prof. Dr. Ronaldo dos Santos Mello (UFSC), como trabalho de conclusão do curso de pós-graduação em banco de dados (UNVAG - MT e UFSC). (CANDIDO, 2007).

Para utilização neste trabalho foi utilizada a versão 2.0, de junho, 2007. Não é necessária instalação, basta fazer download do arquivo e executá-lo no disco local.

A figura 40 mostra a tela inicial do BrModelo.

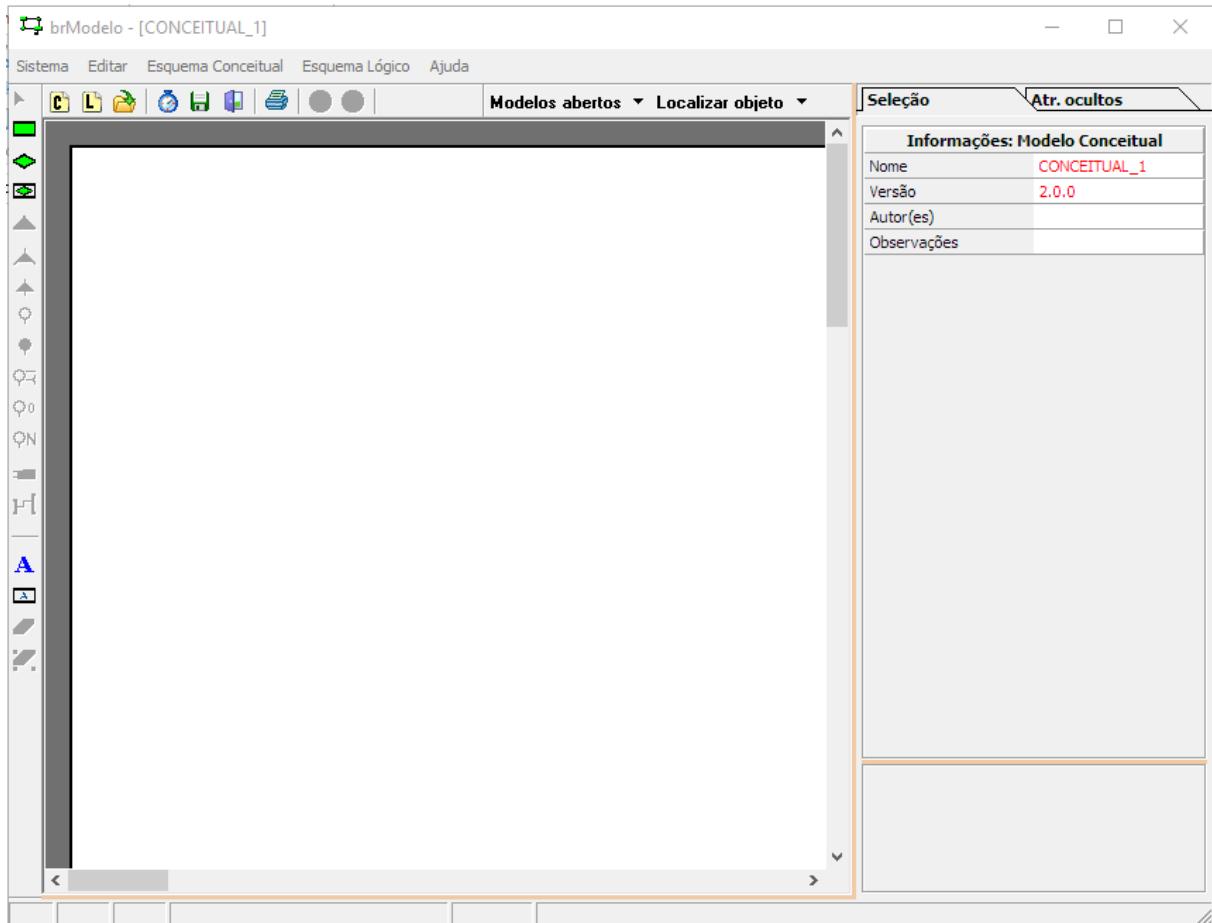


Figura 40. Tela inicial do BrModelo
Fonte: Autor.

Na barra de tarefas, localizada ao lado esquerdo da tela, estão disponíveis os ícones para criação dos objetos do esquema conceitual. Ao passar o mouse sobre cada ícone, é exibida a descrição do que cada objeto faz. Os objetos inativos serão habilitados após a criação da primeira entidade.

Para criar uma entidade, basta clicar no ícone retangular, conforme mostrado na figura 41, e em seguida clicar no espaço em branco da área de trabalho para que ela seja criada.

O BrModelo atribui um nome padrão para cada novo objeto criado. Para configurar o nome, basta clicar sobre o objeto, e no painel de edição localizado ao lado direito da tela, alterar sua identificação, conforme mostrado na figura 42.

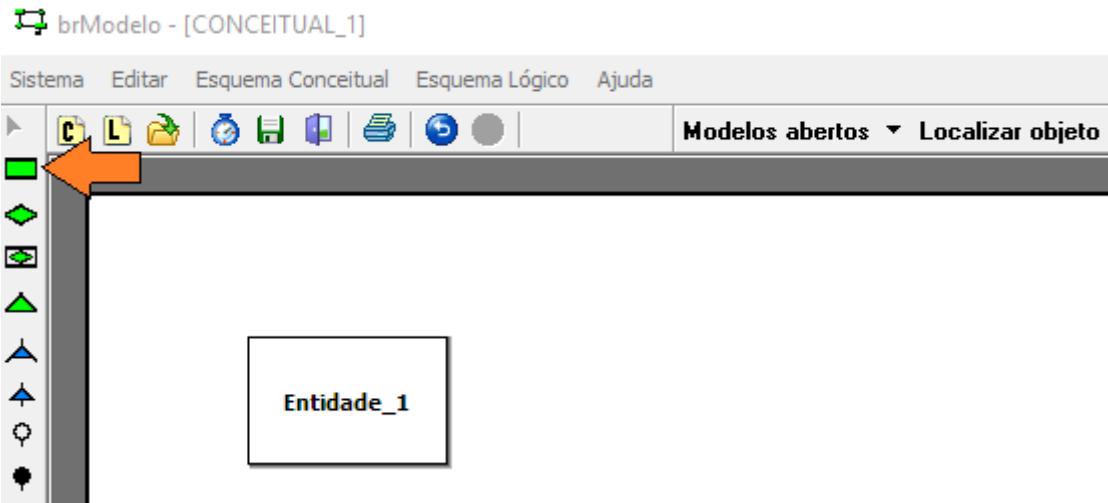


Figura 41. Criando uma entidade no BrModelo
Fonte: Autor.



Figura 42. Configurando a identificação de um objeto
Fonte: Autor.

Para adicionar um atributo identificador, basta clicar no seu respectivo ícone, e em seguida clicar sobre a entidade a qual desejamos adicioná-lo.

Em nosso exemplo, a entidade CLIENTE receberá um atributo identificador que será a representação do campo Id_cliente, o atributo chave da entidade, conforme mostra a figura 43 e 44.

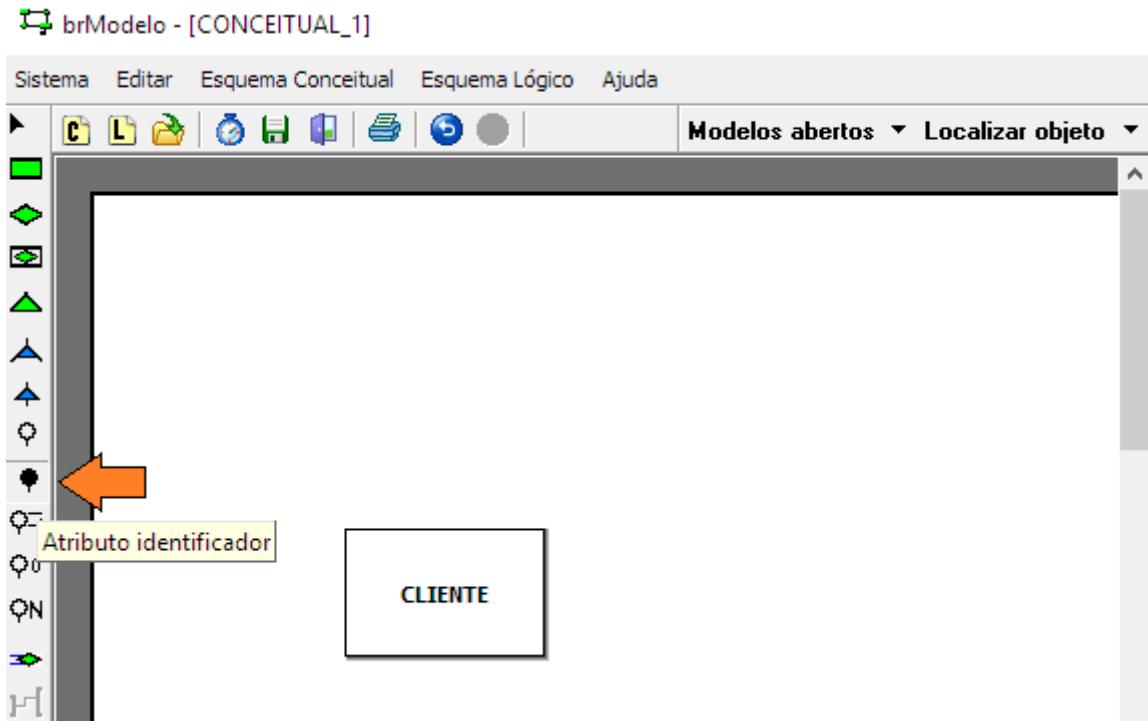


Figura 43. Adicionando um atributo identificador
Fonte: Autor

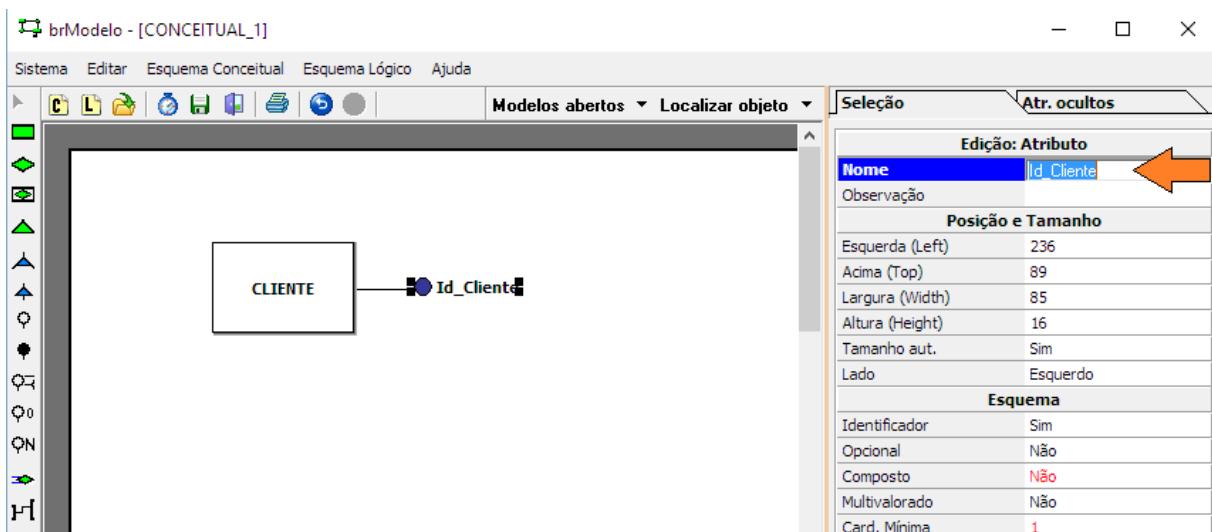


Figura 44. Configurando o atributo identificador
Fonte: Autor

Após a criação e configuração do atributo identificador da entidade CLIENTE, criamos a entidade PROPOSTA.

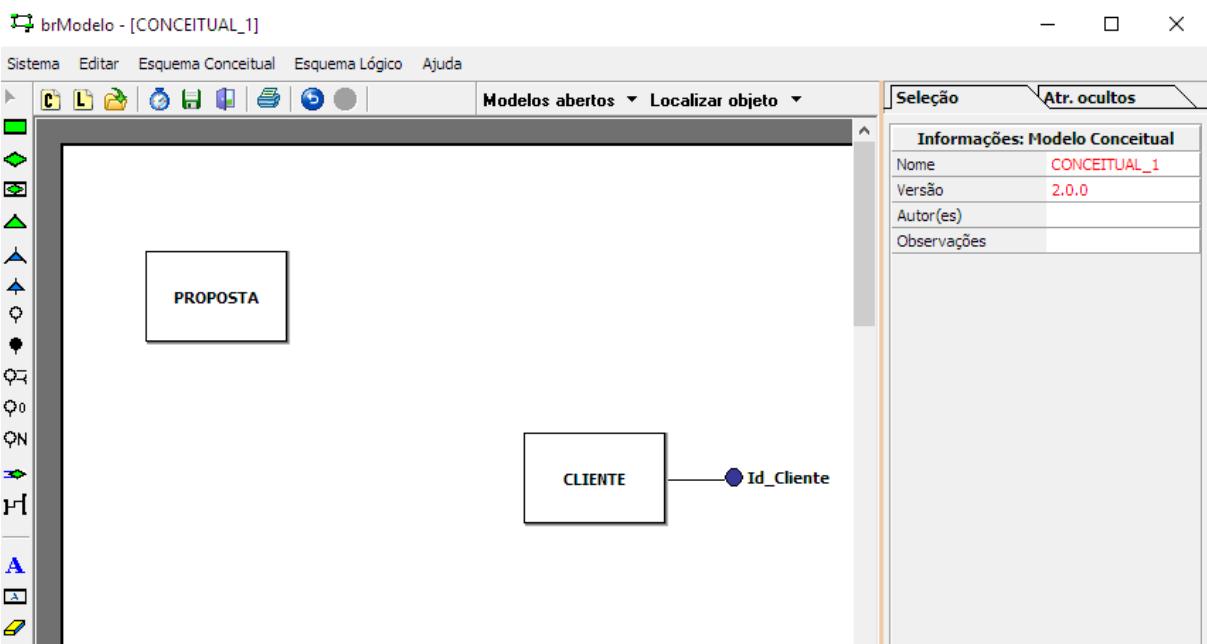


Figura 45. Nova entidade PROPOSTA criada
Fonte: Autor.

Para relacionar a entidade CLIENTE com a entidade PROPOSTA, basta clicar sobre o ícone “ligar objetos”, conforme mostra a figura 46, em seguida clicar sobre a entidade CLIENTE e sobre a entidade PROPOSTA. Um relacionamento será criado.

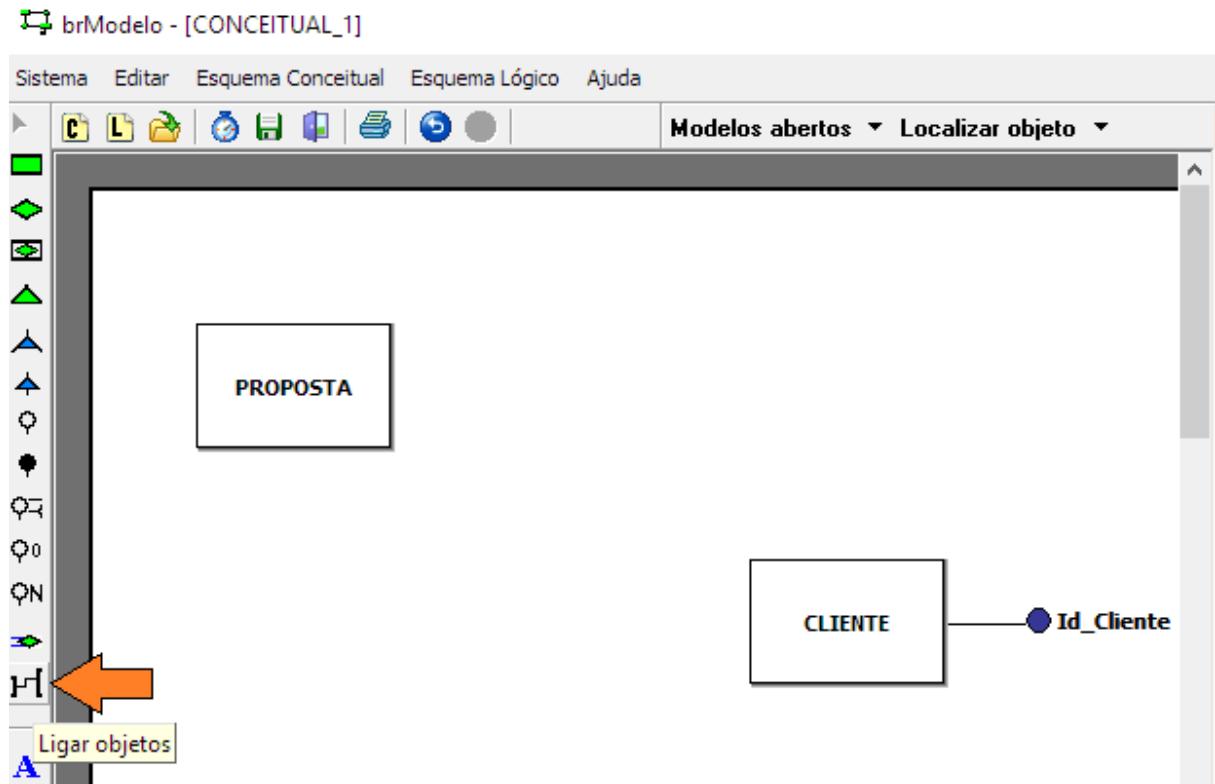


Figura 46. Adicionando relacionamento entre entidades no BrModelo
Fonte: Autor.

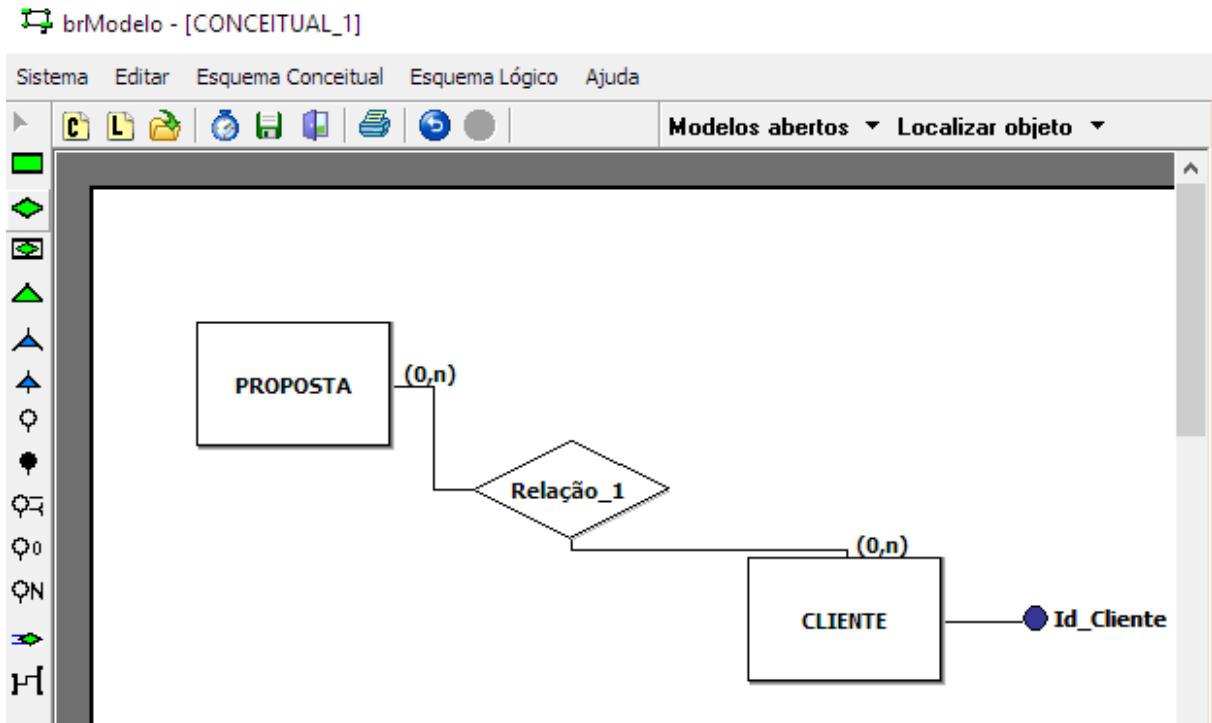


Figura 47. Criado um relacionamento entre duas entidades
Fonte: Autor.

Para configurar as propriedades, basta clicar sobre o objeto, e no painel de edição localizado ao lado direito da tela, alterar suas configurações. A figura 48, mostra a alteração no nome do relacionamento, na razão de cardinalidade e na restrição de participação entre as entidades CLIENTE E PROPOSTA.

Para salvar o trabalho, para clicar em “Sistema” e “Salvar”.

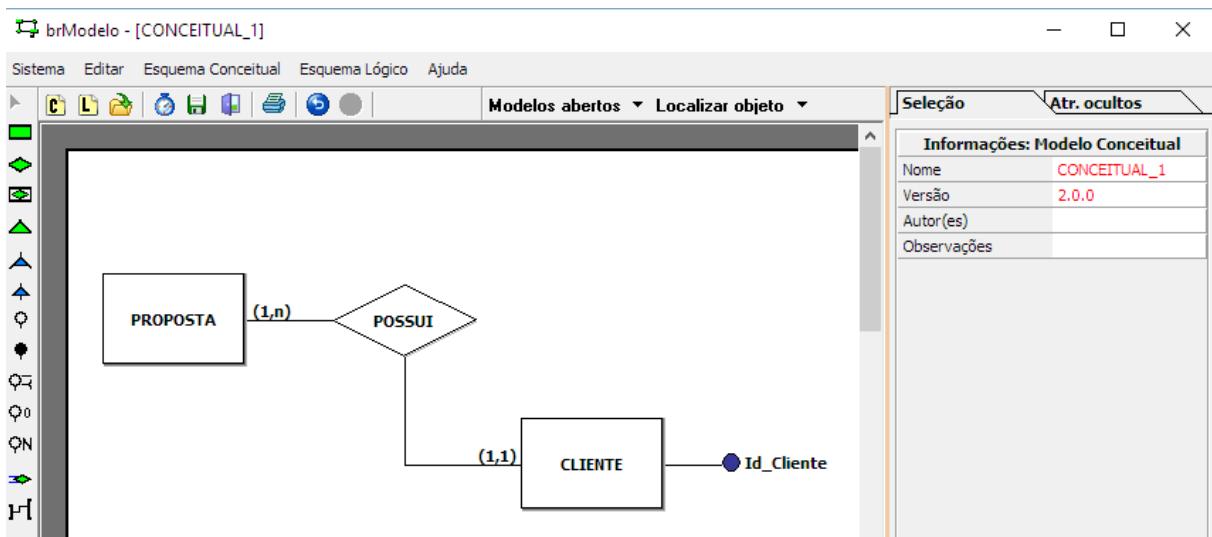


Figura 48. Representação conceitual de duas entidades
Fonte: Autor.

3.4.2 Diagrama Entidade Relacionamento

Observe a figura 49, criamos a entidade CLIENTE, identificando como atributo chave o campo Id_Cliente e como atributos compostos o Telefone e Endereco_Cliente. A restrição de participação da entidade CLIENTE no relacionamento com a entidade PROPOSTA é 1 enquanto que a razão de cardinalidade é N, lembre-se que a restrição de participação, ou cardinalidade mínima, especifica se a existência de uma entidade depende dela estar relacionada a outra por meio do tipo de relacionamento, neste caso, um cliente deve ter no mínimo uma proposta associada, isso porque nesse modelo, conforme foi esclarecido nas reuniões iniciais, um cliente só é registrado no sistema após o interesse em algum lote, nesta fase é criada uma proposta para que todos as etapas processuais sejam acompanhadas, desde visitas até simulações de financiamento. A razão de cardinalidade é n porque um cliente pode ter várias propostas associadas ao seu cadastro, isso porque se um cliente visitar vários lotamentos e não gostar de nenhum, é preciso ter a informação de que ele já visitou e não teve interesse.

Conforme mostra a figura 50, o atributo chave da entidade PROPOSTA é Num_proposta, isso quer dizer que, conforme demonstrado na figura 49, embora um cliente possa ter várias propostas, cada proposta terá um número distinto, por este motivo, conforme a notação indicada ao lado da entidade CLIENTE, cada número de proposta deve ter no mínimo 1 e no máximo 1 cliente associado. Em outras palavras, não pode haver o mesmo número de proposta associado a mais de um cliente.

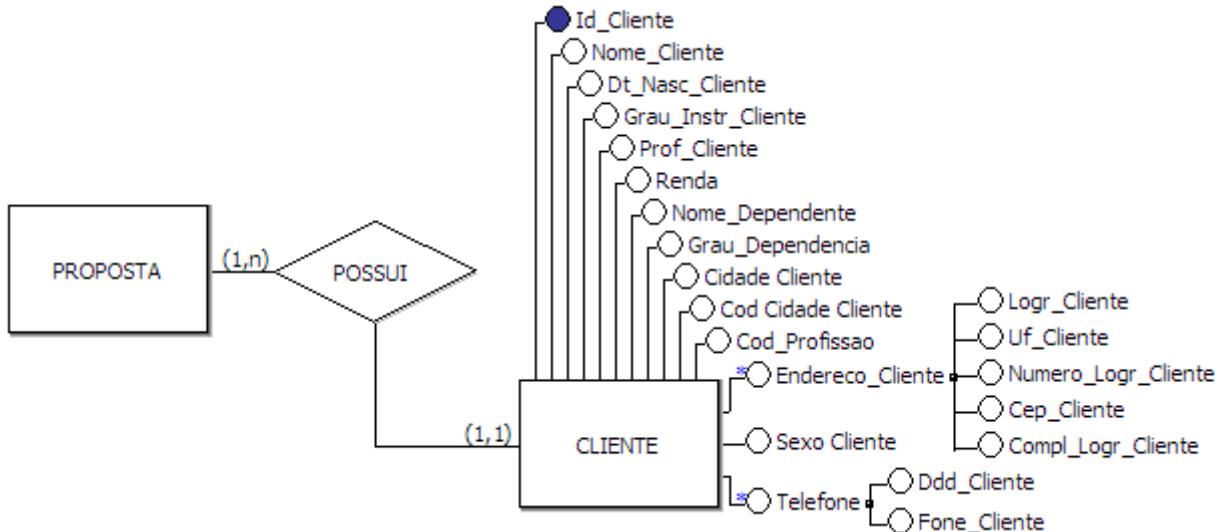


Figura 49. DER das entidades Cliente e Proposta
Fonte: Autor.

No arquivo de propostas, conforme já mencionado, há atributos que em nosso modelo serão transformados em uma nova entidade, conforme mostra a figura 50, será chamada de VENDEDOR. Embora o objetivo desta fase não seja a normalização, de certa forma com este procedimento é justamente isso que estamos fazendo.

Observe a figura 50, note que a entidade VENDEDOR possui como atributo chave Cod_Vendedor, note também que o atributo Telefone é um atributo composto, seus atributos componentes são DDD_Vendedor e Telefone_Vendedor. Observe que a notação para a entidade vendedor é (0,n), isso significa que um vendedor pode não ter propostas associadas a ele, cenário comum em início de mês, mas também não há limite máximo de propostas para cada vendedor. A entidade VENDEDOR está vinculada à entidade PROPOSTA por meio do relacionamento “Criado por”.

Conforme documentado durante o levantamento de requisitos, embora qualquer vendedor possa efetivar uma proposta, neste modelo, apenas o vendedor responsável pela proposta é quem ganha a comissão da venda. É por este motivo que uma proposta não pode existir sem um Cod_Vendedor, bem como não pode haver mais de um vendedor associado a uma mesma proposta. É importante ressaltar que este é o comportamento deste minimundo. Esta característica está representada no modelo através da notação ao lado da entidade VENDEDOR, (1,1).

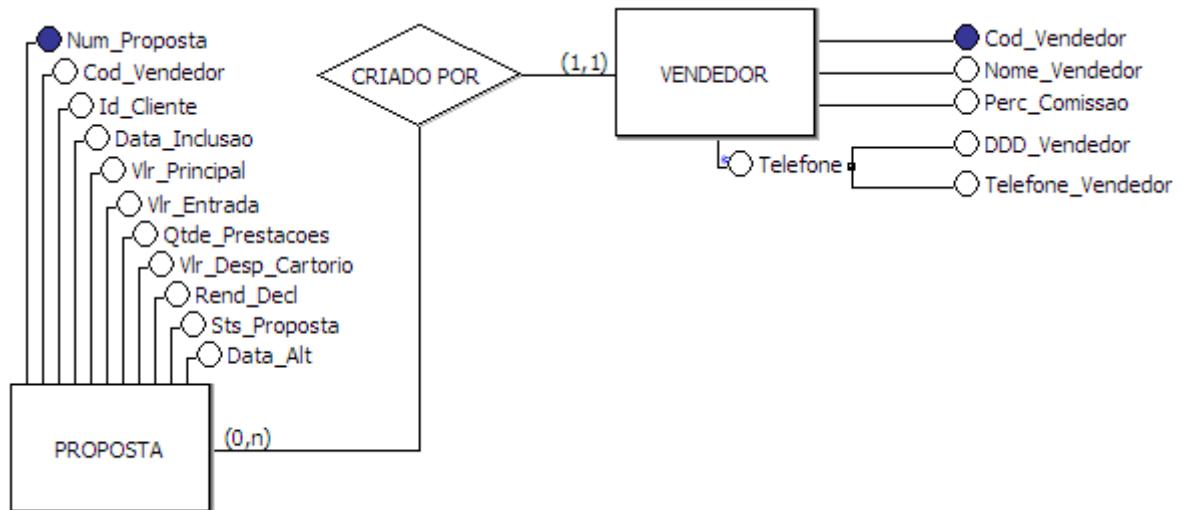


Figura 50. DER das entidades Proposta e Vendedor

Fonte: Autor.

A figura 51 descreve o relacionamento entre as entidades PROPOSTA e CONTRATO. Uma proposta não necessariamente precisa gerar um contrato, no entanto, caso efetivada, deve estar associada a no máximo 1 número de contrato. Conforme descreve o modelo, um contrato é originado a partir da efetivação de uma proposta, desta forma, um contrato deve ter no mínimo 1 e no máximo 1 proposta associada a ele. Observe que a entidade CONTRATO assim como a entidade PROPOSTA possui atributos que indicam a criação de novas entidades, os atributos Código_Loteamento e Código_Lote remetem à criação de duas novas entidades.

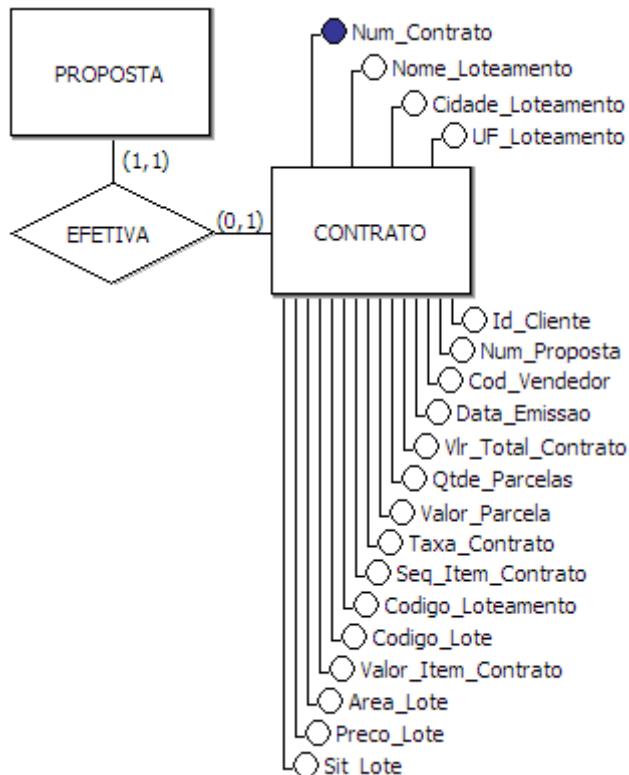


Figura 51. DER das entidades Proposta e Contrato

Fonte: Autor.

Antes de tratarmos do desmembramento da entidade CONTRATO, analisaremos o último arquivo, representado aqui pela entidade PARCELA.

Na figura 52, observe que a chave da entidade PARCELA é uma chave composta, esta chave é a combinação dos atributos Num_Contrato e Num_Parcela.

Cada contrato e sua respectiva parcela da entidade PARCELA, deve estar associado a no mínimo 1 e no máximo 1 contrato da entidade CONTRATO, conforme representado na notação min, max ao lado da entidade contrato.

Um contrato é um acordo de financiamento, este acordo de financiamento gera um determinado número de parcelas que pode variar de 1 a N parcelas por contrato, por este motivo esta é a representação indicada ao lado da entidade PARCELA.

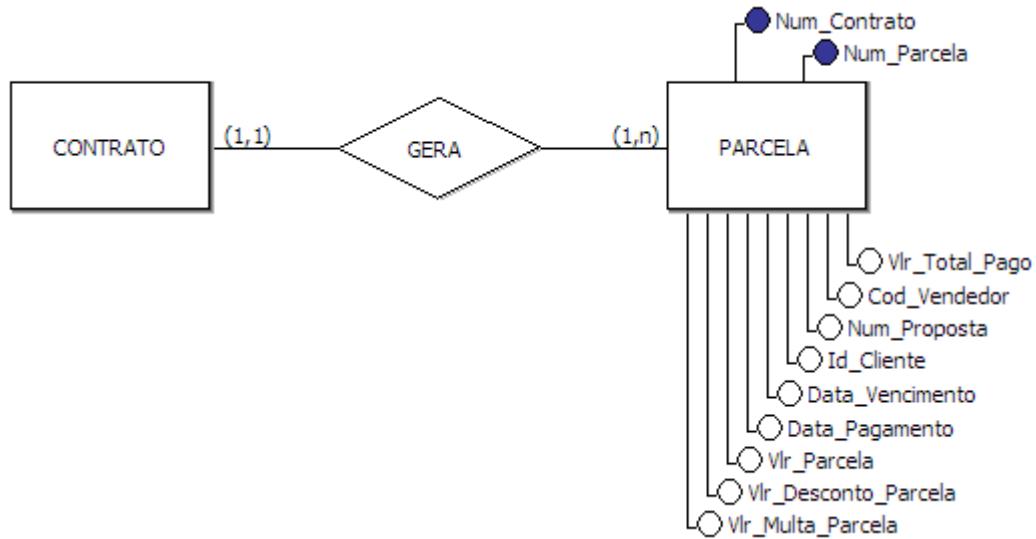


Figura 52. DER das entidades Contrato e Parcela

Fonte: Autor.

A figura 53 mostra o primeiro modelo conceitual do projeto. Para seguir um padrão, utilizamos apenas letras maiúsculas para descrever entidades e relacionamentos, o nome dos atributos possui a primeira letra de cada palavra em letra maiúscula, e o atributo chave é destacado dos demais atributos.

A modelagem de dados vista pelo modelo conceitual é mais fácil de ser compreendida, o analista de negócios pode entender o que está relacionado com o que, bem como quais entidades e atributos estão disponíveis no modelo. Obviamente que em entidades extensas a descrição de todos atributos pode ficar apenas no modelo lógico.

Conforme citado neste trabalho, o modelo conceitual é uma descrição concisa dos requisitos de dados dos usuários, como não incluem descrições detalhadas de implementação, este modelo é mais fácil de ser entendido e pode ser usado para a comunicação com usuários não técnicos.

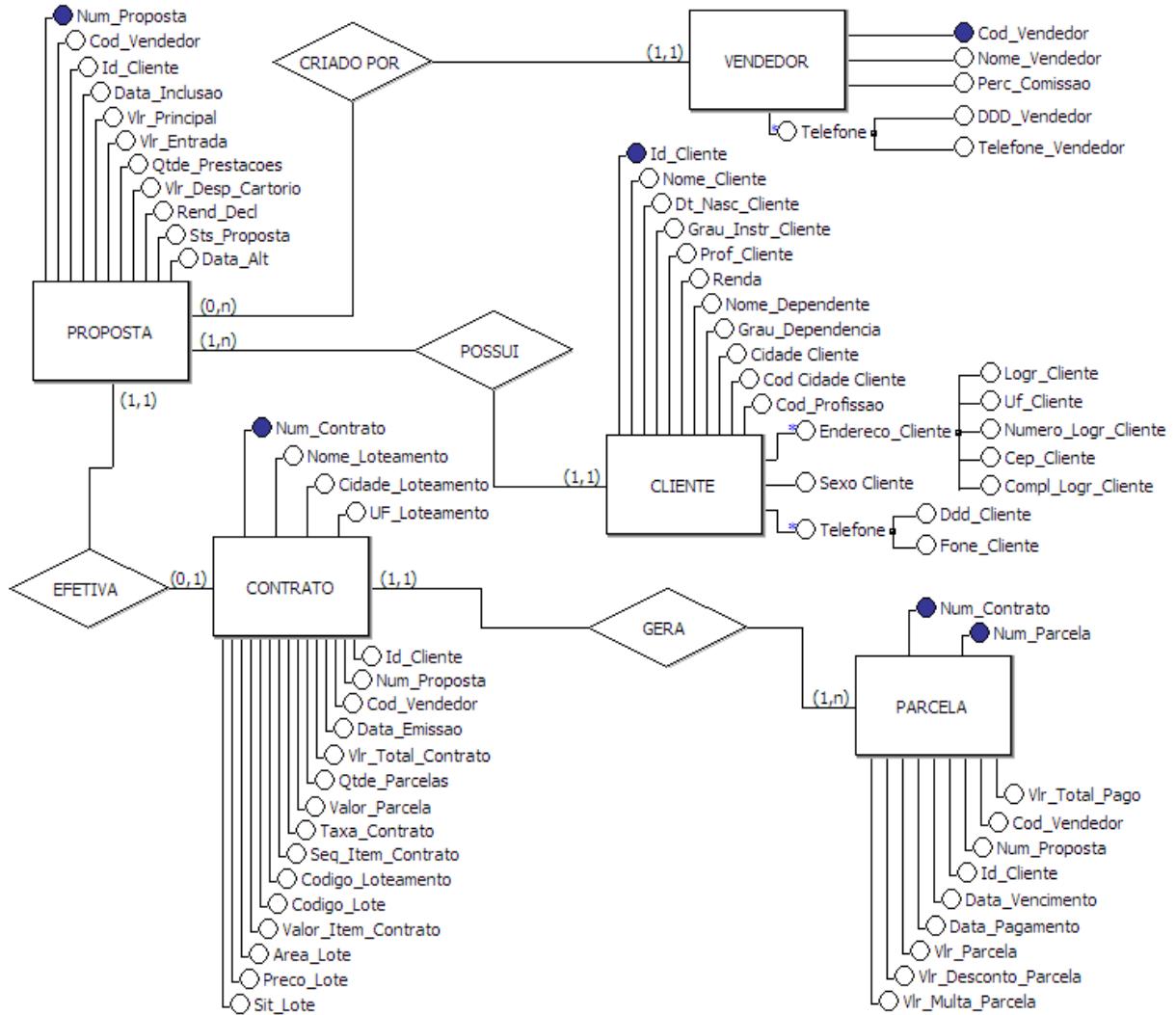


Figura 53. DER Inicial do Projeto da Imobiliária
Fonte: Autor.

Criamos neste trabalho a estrutura conceitual inicial, mas ainda há alguns ajustes necessários.

Por exemplo, definimos como atributo chave da entidade CONTRATO o atributo Num_contrato, se sabemos que um atributo chave de uma entidade é composto por um campo que possui valores únicos, ou seja, distintos, não podemos inserir combinações que gerem duplicidade, caso isso ocorra, haverá problemas no processo de carga justamente pela violação da restrição de existência deste atributo.

No mínimo modelado, uma característica importante a ser considerada é que um contrato pode conter um ou mais lotes, no mesmo ou em loteamentos diferentes.

Por exemplo, observe a seguir na figura 54.

Um contrato de financiamento pode referir-se apenas ao lote 3 do loteamento 1, ou aos lotes 2 e 3 do loteamento 1, ou ainda, caso haja disponibilidade e o cliente preferir, ele pode em um único contrato, financiar o lote 3 do loteamento 1 e o lote 1 do loteamento 2.

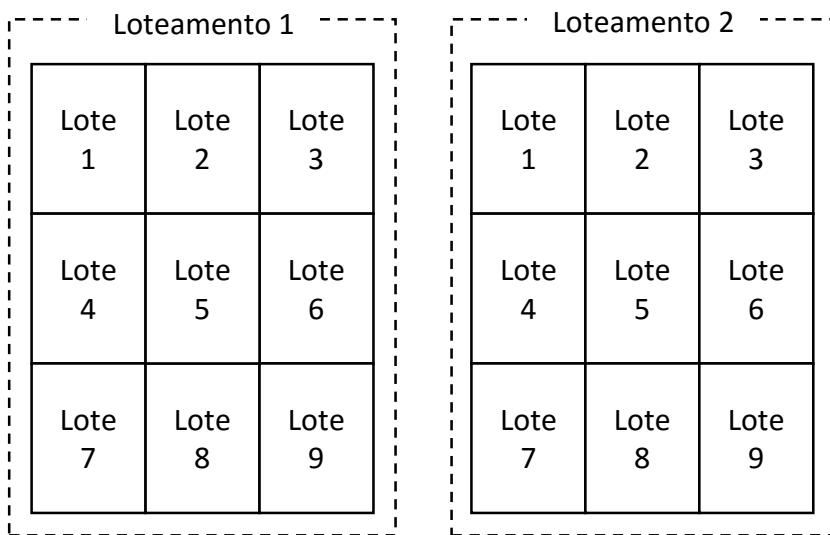


Figura 54. Característica do minimundo modelado

Fonte: Autor.

Conforme você já tenha notado, esta característica do negócio inviabiliza, na nossa entidade CONTRATO, mantermos como atributo chave apenas o número do contrato, isso porque o contrato aparecerá em várias linhas quando o cliente financiar mais de um lote. Os atributos Código_Loteamento e Código_Lote, embora estejam na nossa entidade contrato, remetem à criação de duas novas entidades.

Você pode questionar se poderíamos usar chaves compostas para a entidade CONTRATO, no entanto, observe que temos mais de um assunto nesta entidade. Não faz sentido termos uma entidade para falar de contratos e dentro dela guardarmos informações de lote e loteamento, além disso, o objetivo é um banco de dados normalizado, não queremos um banco de dados com informações redundantes.

Separamos os subgrupos de informação contidos no arquivo de contratos, criando três novas entidades, ITEM CONTRATO, LOTE e LOTEAMENTO, o mesmo fizemos com a entidade CLIENTE, separando os dados de dependentes em um nova entidade.

Observe na figura 55 o modelo conceitual com estas últimas alterações.

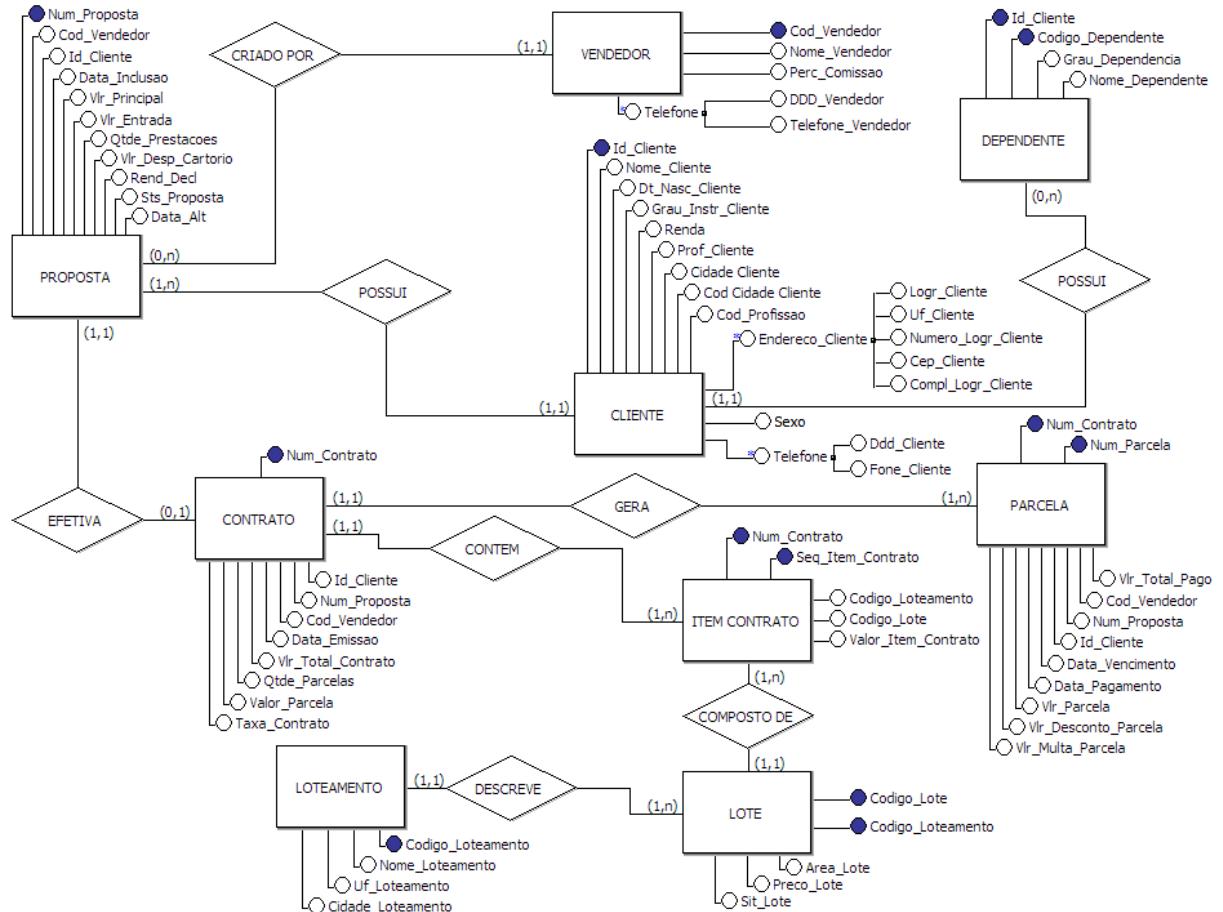


Figura 55. DER atual do Projeto da Imobiliária
Fonte: Autor.

Este será o modelo conceitual do projeto e também o ponto de partida para criação do modelo lógico onde de fato aplicaremos os testes para adequação às formas normais. Conforme demonstrado até aqui, a criação do modelo conceitual foi realizada considerando regras do minimundo em questão, por este motivo, precisamos muitas vezes envolver o cliente, pois o foco nesta etapa é discutir os aspectos do negócio e não da tecnologia. Esta é a tarefa inicial na implementação de um banco de dados, e pode ser a responsável por muitos imprevistos caso não seja abordada corretamente.

3.5 Modelo Lógico

Após a criação do modelo conceitual, agora vamos para a criação do modelo lógico. Diferentemente do modelo conceitual, o modelo lógico leva em consideração algumas características do SGBD. Conforme mencionado, utilizaremos um banco de

dados relacional, neste banco de dados as entidades criadas no modelo conceitual serão organizadas na forma de tabelas. É no modelo lógico que faremos a adequação de padrão e nomenclatura, definiremos as chaves primárias, as chaves estrangeiras, restrições de domínio e os testes das formas normais para o processo de normalização.

3.5.1 DBDesigner 4 e DBDesigner Fork

DBDesigner 4 é uma ferramenta de design de banco de dados gratuita que permite a modelagem, criação e manutenção do banco de dados em um único ambiente, foi desenvolvida pela empresa Fabulous Force Database Tools sob a licença GNU GPL (General Public License). É uma ferramenta CASE, (Computer-Aided Software Engineering), que trabalha com o modelo lógico para a modelagem de dados, ou seja, ela não gera o modelo conceitual proposto por Chen.

Com esta ferramenta você cria o modelo lógico, define chaves primárias, chaves estrangeiras, índices, tipos de domínio dos atributos e até mesmo restrições de integridade, por fim você gera o script de criação em um documento de texto, o qual você utiliza para criar os objetos no banco de dados. A ferramenta possibilita também a engenharia reversa, gerando o modelo de dados a partir de um banco existente e possui a função de sincronização que é utilizada para alterar automaticamente a base de dados quando o modelo lógico for alterado.

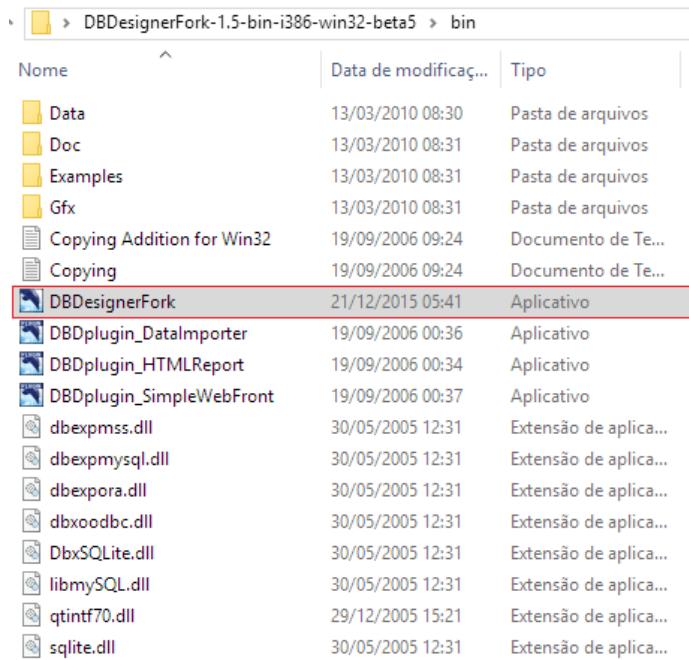
DBDesigner 4 é um software desenvolvido para rodar em plataforma Windows ou Linux, no entanto a ferramenta foi construída e otimizada para oferecer suporte apenas ao banco de dados MySQL, ou seja, ela não exporta scripts SQL para outros bancos de dados (Oracle, SQL Server, etc.) (DbDesigner4).

Neste trabalho utilizaremos o DBDesigner Fork, que possui as mesmas funcionalidades que o DBDesigner 4, praticamente é o mesmo software, no entanto, o DB Designer Fork é otimizado para os demais bancos de dados, ou seja, sincroniza e gera scripts SQL para Oracle, SQL Server, MySQL, Firebird, SQLite e PostgreSQL.

O DBDesigner Fork não precisa ser instalado, basta fazer o download da ferramenta em <http://sourceforge.net/projects/dbdesigner-fork/> e descompactar o software no disco local (DBDesignerFork).

3.5.2 Utilizando o DBDesigner Fork

Após baixar o aplicativo e descompactá-lo no disco local, acesse a pasta bin, e clique no primeiro ícone do aplicativo, DBDesignerFork, conforme a figura 56.



Nome	Data de modificação	Tipo
Data	13/03/2010 08:30	Pasta de arquivos
Doc	13/03/2010 08:31	Pasta de arquivos
Examples	13/03/2010 08:31	Pasta de arquivos
Gfx	13/03/2010 08:31	Pasta de arquivos
Copying Addition for Win32	19/09/2006 09:24	Documento de Texto
Copying	19/09/2006 09:24	Documento de Texto
DBDesignerFork	21/12/2015 05:41	Aplicativo
DBDplugin_DataImporter	19/09/2006 00:36	Aplicativo
DBDplugin_HTMLReport	19/09/2006 00:34	Aplicativo
DBDplugin_SimpleWebFront	19/09/2006 00:37	Aplicativo
dbexpmss.dll	30/05/2005 12:31	Extensão de aplicação
dbexpmysql.dll	30/05/2005 12:31	Extensão de aplicação
dbxpora.dll	30/05/2005 12:31	Extensão de aplicação
dbxoodbc.dll	30/05/2005 12:31	Extensão de aplicação
DbxSQLite.dll	30/05/2005 12:31	Extensão de aplicação
libmySQL.dll	30/05/2005 12:31	Extensão de aplicação
qtintf70.dll	29/12/2005 15:21	Extensão de aplicação
sqlite.dll	30/05/2005 12:31	Extensão de aplicação

Figura 56. Diretório da aplicação DBDesignerFork

Fonte: Autor.

Ao clicar no item destacado na figura 56, abra o DBDesigner Fork e a tela inicial será exibida, conforme na figura 57. No centro da tela aparece uma pequena janela com algumas dicas referentes a plataforma. Se você não quiser exibir esta janela toda vez que o aplicativo é iniciado selecione a opção “Do not show tips on startup”.

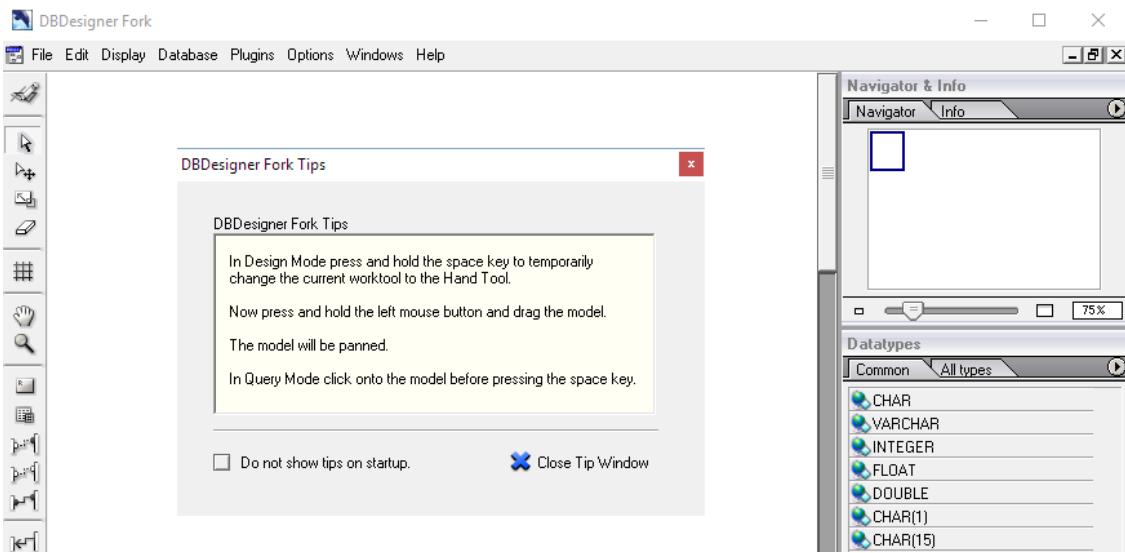


Figura 57. Tela inicial do DBDesigner Fork

Fonte: Autor.

Antes de iniciarmos a criação do modelo, escolheremos o tipo de notação para representar o relacionamento entre as tabelas, podemos alterar a forma que a aplicação apresenta a notação indo no menu “Display” e escolher a opção “Notation”, conforme mostra a Figura 58.

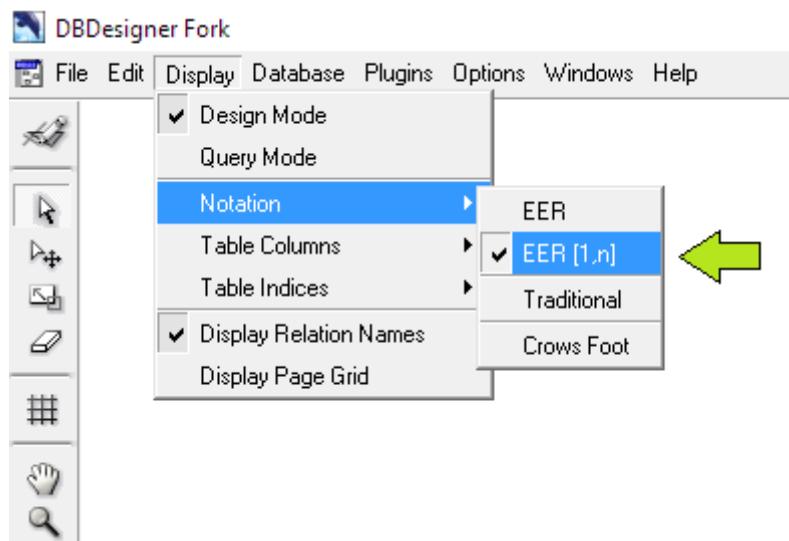


Figura 58. Configuração do tipo de notação

Fonte: Autor.

Na área de trabalho em branco é onde criaremos o modelo. Para criar as tabelas correspondentes às entidades do nosso modelo conceitual, basta clicar no botão “New Table” na barra de ferramentas vertical que está localizada à esquerda da janela, conforme mostra a figura 59.



Figura 59. Ícone para criação de nova tabela

Fonte: Autor.

Após selecionar o ícone de nova tabela basta clicar em qualquer parte da área de trabalho em branco.

Conforme mostra a Figura 60, será apresentada a notação gráfica da tabela.

Para excluir a tabela, basta selecioná-la e pressionar o botão delete.

Para configurar a tabela basta dar um duplo clique sobre a mesma e surgirá a janela “Table Editor”, conforme mostra a Figura 61.

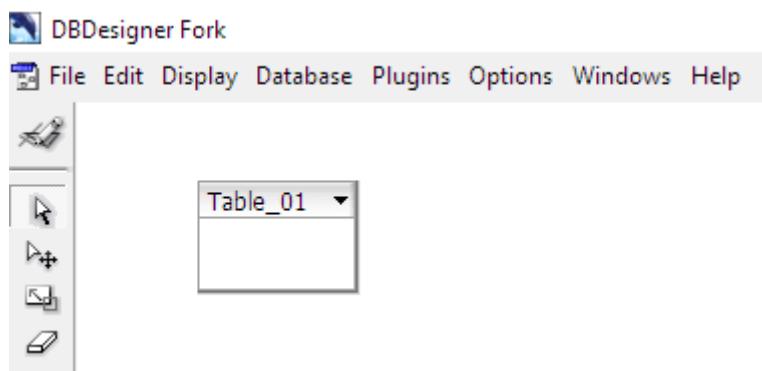


Figura 60. Tabela criada na área de trabalho

Fonte: Autor.

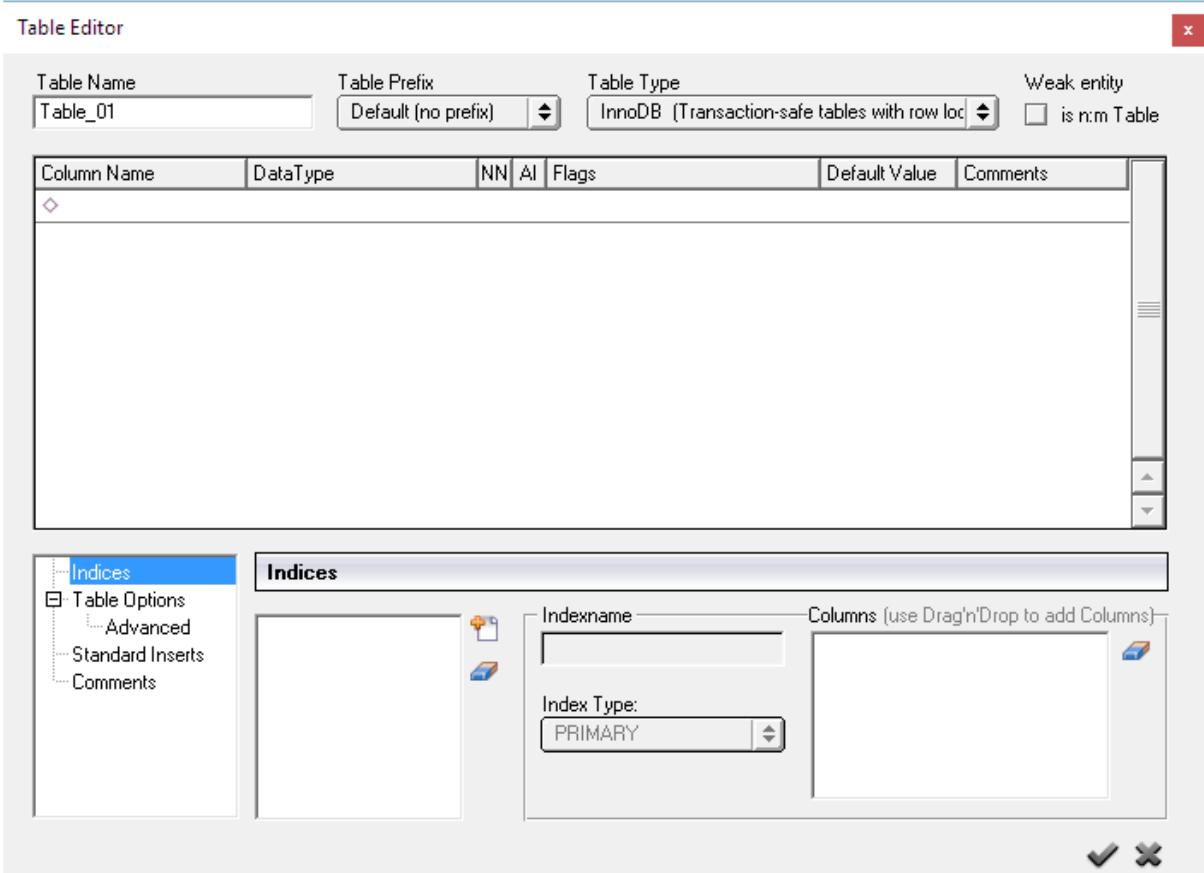


Figura 61. Configurando a tabela selecionada no Table Editor
Fonte: Autor.

A primeira configuração que devemos fazer é relacionada ao nome da tabela, para isso devemos utilizar o campo "Table Name" e alterar o nome automaticamente criado, note que apenas para seguir um padrão, utilizaremos caixa alta no nome das tabelas.

A primeira entidade que modelaremos é a entidade CLIENTE. Após a alteração do nome, devemos configurar as colunas da tabela, para isso, conforme a figura 62, basta clicar no campo destacado abaixo de "Column Name" e criar o atributo Id_cliente. Observe na figura 63, a tabela com o atributo Id_cliente criado, note que o DBDesigner Fork atribui a primeira coluna criada como sendo a chave primária.

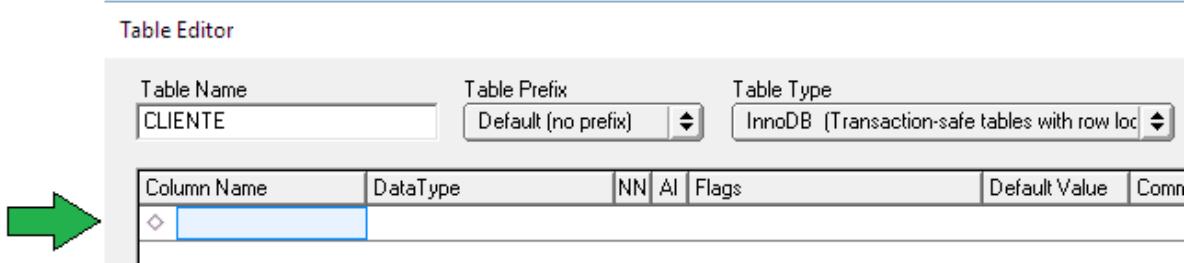


Figura 62. Configurando as colunas da tabela Cliente
Fonte: Autor.

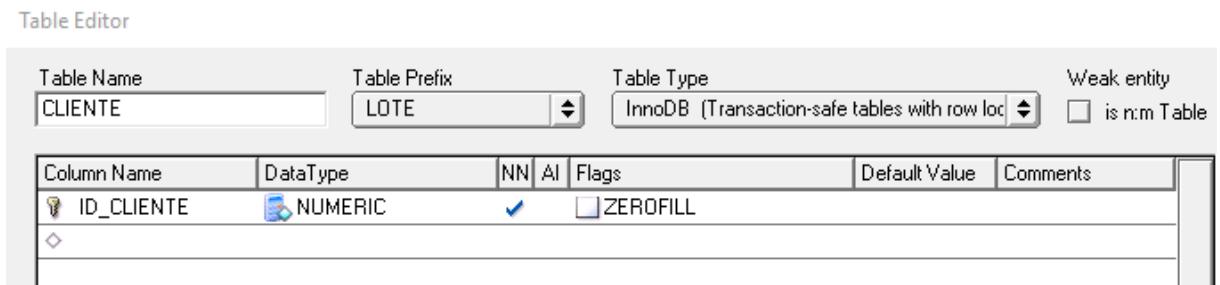


Figura 63. Configuração da coluna que será a chave primária da tabela
Fonte: Autor.

Após digitar o nome da coluna basta dar um ENTER para que ele possa ir para o próximo campo que é o “Data Type”. O “Data Type” é uma caixa drop-down que tem tipos de dados predefinidos. Entre os diversos campos temos o “INTEGER” para valores numéricos inteiros, “FLOAT” para valores numéricos reais, “VARCHAR” para caracteres de texto de qualquer tamanho. O “VARCHAR” ajusta o campo de forma dinâmica, porém podemos definir um tamanho máximo para o “VARCHAR”. Para determinar um tamanho máximo temos as opções “Varchar(20)”, “Varchar(45)” e “Varchar(255)”. Caso seja necessário outro valor basta clicar em um desses valores e editar o valor dentro dos parênteses para o valor desejado.

Podemos verificar ainda outras características que podem ser configuradas após o “Data Type”.

A opção "NN", (Not Null), significa que, quando selecionado, não é permitido deixar a coluna vazia, atributos definidos como chave primária sempre terão essa opção selecionada. A opção "AI", (Auto Increment), significa que os valores da coluna serão criados automaticamente pelo banco de dados. A opção "Flags" é alterada automaticamente de acordo com o tipo de dado escolhido. A opção "Default Value" é utilizada quando queremos definir um valor padrão para uma coluna, mesmo quando nenhum dado seja configurado para ela. Por fim, a opção

"Comments" serve para que possamos colocar comentários sobre as colunas criadas.

Finalizada a configuração do primeiro atributo, podemos adicionar os demais abaixo deste. Na tela “Table Editor” podemos notar também a presença de uma chave bem à esquerda da coluna “Column Name”, isto indica que o atributo é uma chave primária, para os demais atributos podemos também configurar outras chaves-primárias, basta clicar no símbolo à esquerda, observe na figura 64, com isso podemos configurar e desconfigurar como chave-primária quantos atributos forem necessários.

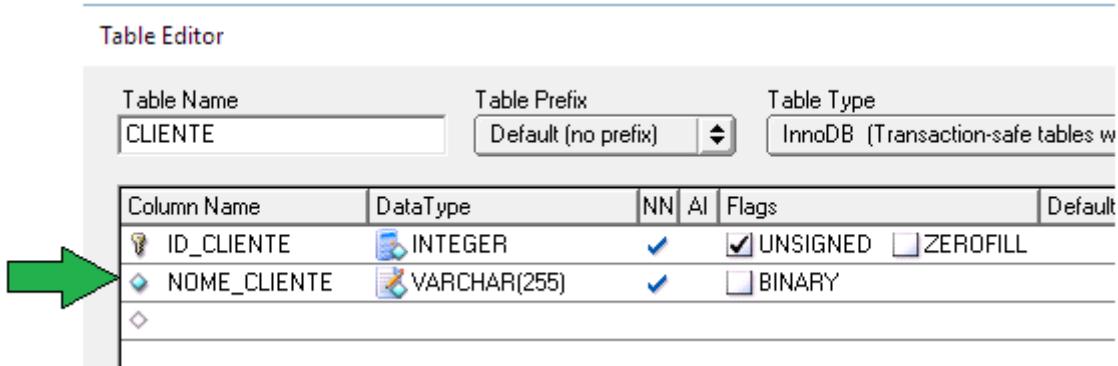


Figura 64. Símbolo à esquerda do Atributo, indicando sua função
Fonte: Autor.

CLIENTE
ID_CLIENTE
NOME_CLIENTE
DTA_NASC_CLIENTE
GRAU_INSTR_CLIENTE
VLR_RENDA
COD_PROFISSAO
PROFISSAO_CLIENTE
LOGR_CLIENTE
NUM_LOGR_CLIENTE
COMPL_LOGR_CLIENTE
NUM_CEP
UF_CLIENTE
NUM_DDD_CLIENTE
NUM_FONE_CLIENTE
CIDADE_CLIENTE
COD_CIDADE_CLIENTE
SEXO

Figura 65.Tabela Cliente e seus Atributos
Fonte: Autor.

Conforme mencionado, no modelo lógico faremos a adequação de padrão e nomenclatura. De acordo com o metadados apresentado na tabela 33, identificamos o tipo de domínio dos atributos. Os atributos numéricos serão representados com prefixos nas iniciais, sendo eles, COD, NUM e VLR, observe que ID é um identificador exclusivo da entidade CLIENTE e também é numérico, por este motivo optamos por não alterá-lo. Os atributos cujo tipo de domínio é data, receberão o prefixo DTA e os atributos sem algum dos prefixos mencionados serão textuais. Observe que desta forma o próprio nome do campo já indica o tipo de domínio ao qual ele pertence, se é numérico, data ou textual.

No DBDesigner Fork também podemos alterar a exibição das tabelas na área de trabalho através do menu “Display”, que permite configurarmos a aparência da tabela, como por exemplo exibir ou ocultar o domínio dos atributos.

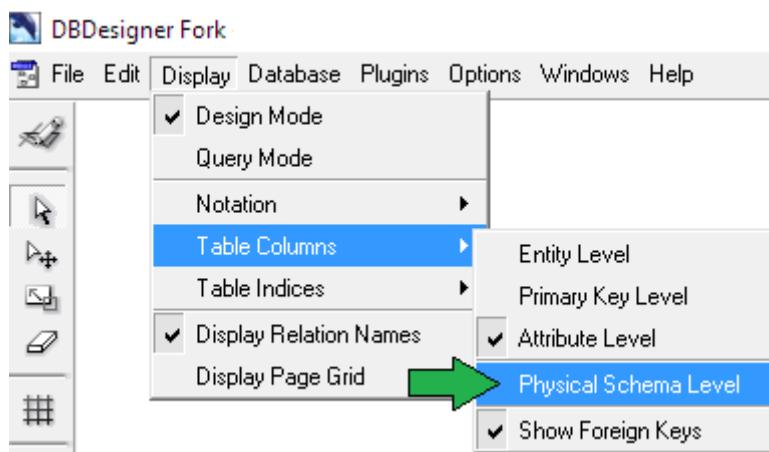


Figura 66. Alterando aparência das Tabelas para exibir domínio dos atributos
Fonte: Autor.

Para salvar o trabalho basta ir em "File" e "Save As", conforme mostra a Figura 67, salvamos o modelo da tabela CLIENTE com seus atributos e tipo de exibição alterada.

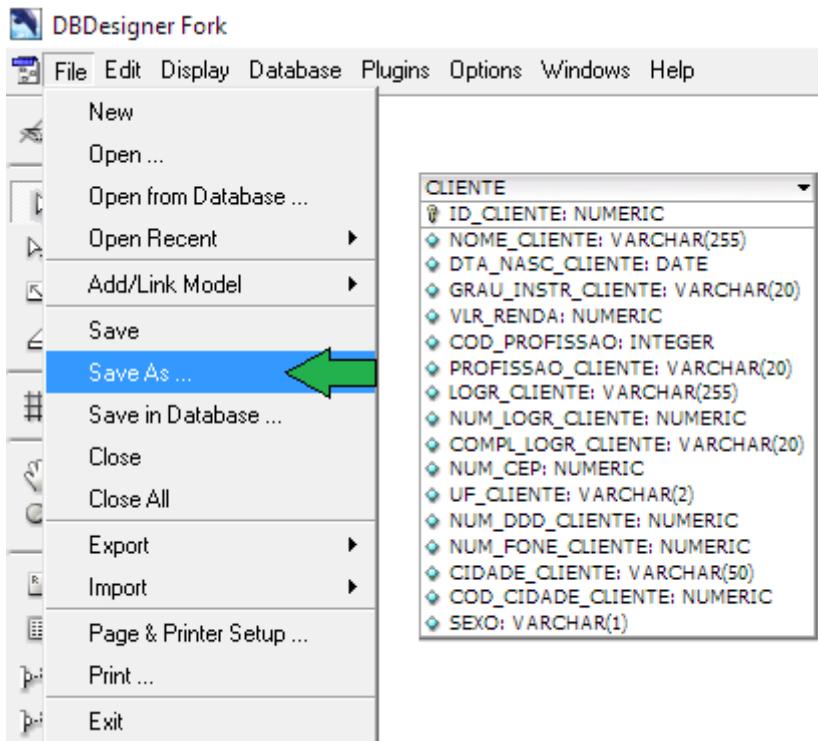


Figura 67. Salvando modelo

Fonte: Autor.

Para criar outras tabelas o processo é o mesmo, basta clicar no ícone correspondente na barra de ferramentas e executar as configurações conforme demonstrado anteriormente. De acordo com nosso DER, criamos a tabela DEPENDENTE.

The screenshot shows two tables side-by-side. On the left is the 'CLIENTE' table with its 15 columns listed. On the right is the newly created 'DEPENDENTE' table, which has three columns: COD_DEPENDENTE (NUMERIC), NOME_DEPENDENTE (VARCHAR(50)), and GRAU_DEPENDENCIA (VARCHAR(20)).

CLIENTE	
•	ID_CLIENTE: NUMERIC
•	NOME_CLIENTE: VARCHAR(255)
•	DTA_NASC_CLIENTE: DATE
•	GRAU_INSTR_CLIENTE: VARCHAR(20)
•	VLR_RENDAS: NUMERIC
•	COD_PROFISSAO: INTEGER
•	PROFISSAO_CLIENTE: VARCHAR(20)
•	LOGR_CLIENTE: VARCHAR(255)
•	NUM_LOGR_CLIENTE: NUMERIC
•	COMPL_LOGR_CLIENTE: VARCHAR(20)
•	NUM_CEP: NUMERIC
•	UF_CLIENTE: VARCHAR(2)
•	NUM_DDD_CLIENTE: NUMERIC
•	NUM_FONE_CLIENTE: NUMERIC
•	CIDADE_CLIENTE: VARCHAR(50)
•	COD_CIDADE_CLIENTE: NUMERIC
•	SEXO: VARCHAR(1)

DEPENDENTE	
•	COD_DEPENDENTE: NUMERIC
•	NOME_DEPENDENTE: VARCHAR(50)
•	GRAU_DEPENDENCIA: VARCHAR(20)

Figura 68. Tabela Cliente e nova tabela Dependente Criada

Fonte: Autor.

Nesta etapa podemos criar os relacionamentos entre as tabelas, para isso utilizamos os botões na barra de ferramentas conforme mostrado na figura 69. Note que basta colocar o mouse sobre o ícone e o nome e tipo de relacionamento será apresentado.

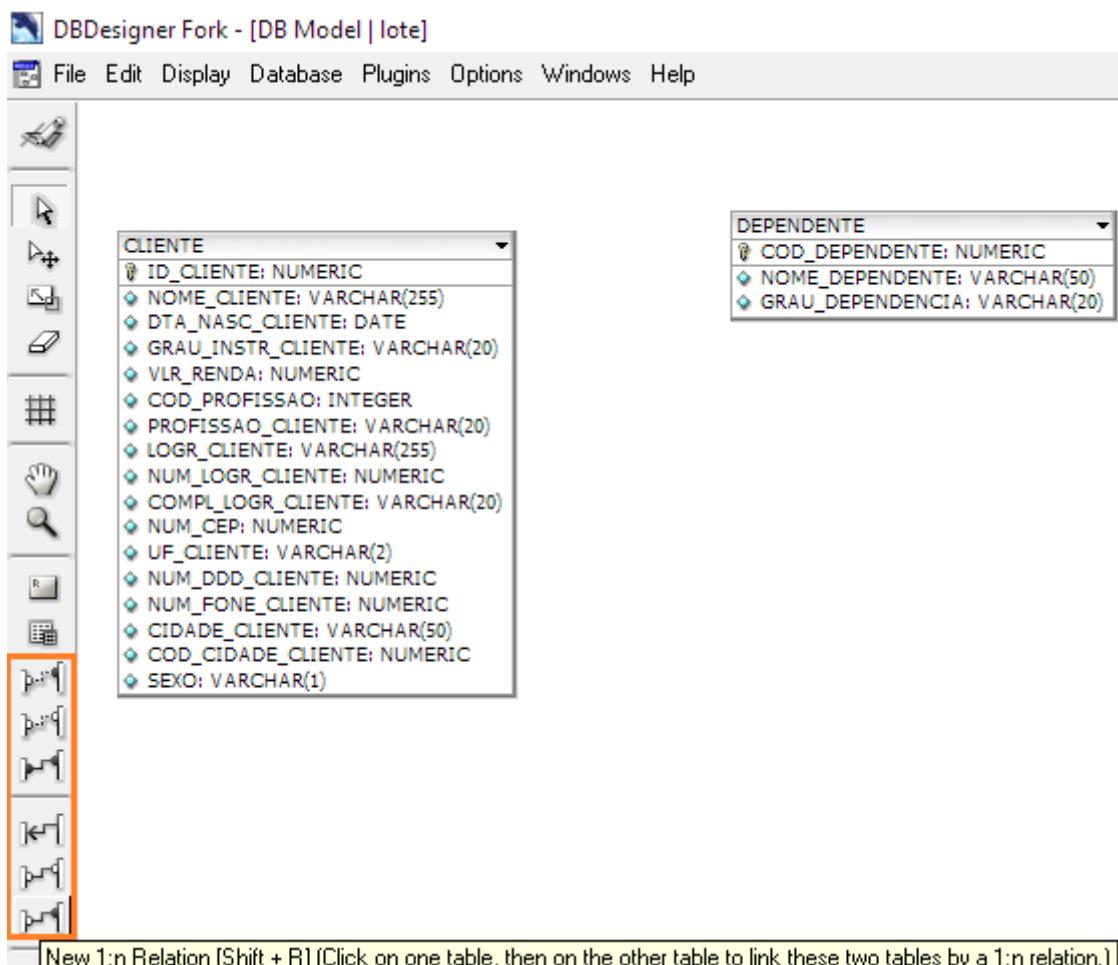


Figura 69. Relacionamentos que podem ser utilizados
Fonte: Autor.

O relacionamento que utilizaremos entre cliente e dependente é do tipo 1:N, representado pelo ícone "New 1:N Relation". Para relacionar duas tabelas, basta clicar no ícone de relacionamento, clicar na tabela desejada e em seguida clicar na outra tabela que queremos relacionar. É importante considerar que devemos obrigatoriamente clicar em primeiro lugar na tabela com cardinalidade 1 do relacionamento e somente depois clicar na tabela com cardinalidade N. Se não fizermos isso o relacionamento será criado com a cardinalidade invertida.

Cada Cliente pode ter N dependentes, deste modo, selecionamos o ícone do relacionamento, clicamos na tabela CLIENTE e em seguida na tabela DEPENDENTE, nosso modelo fica conforme a figura 70.

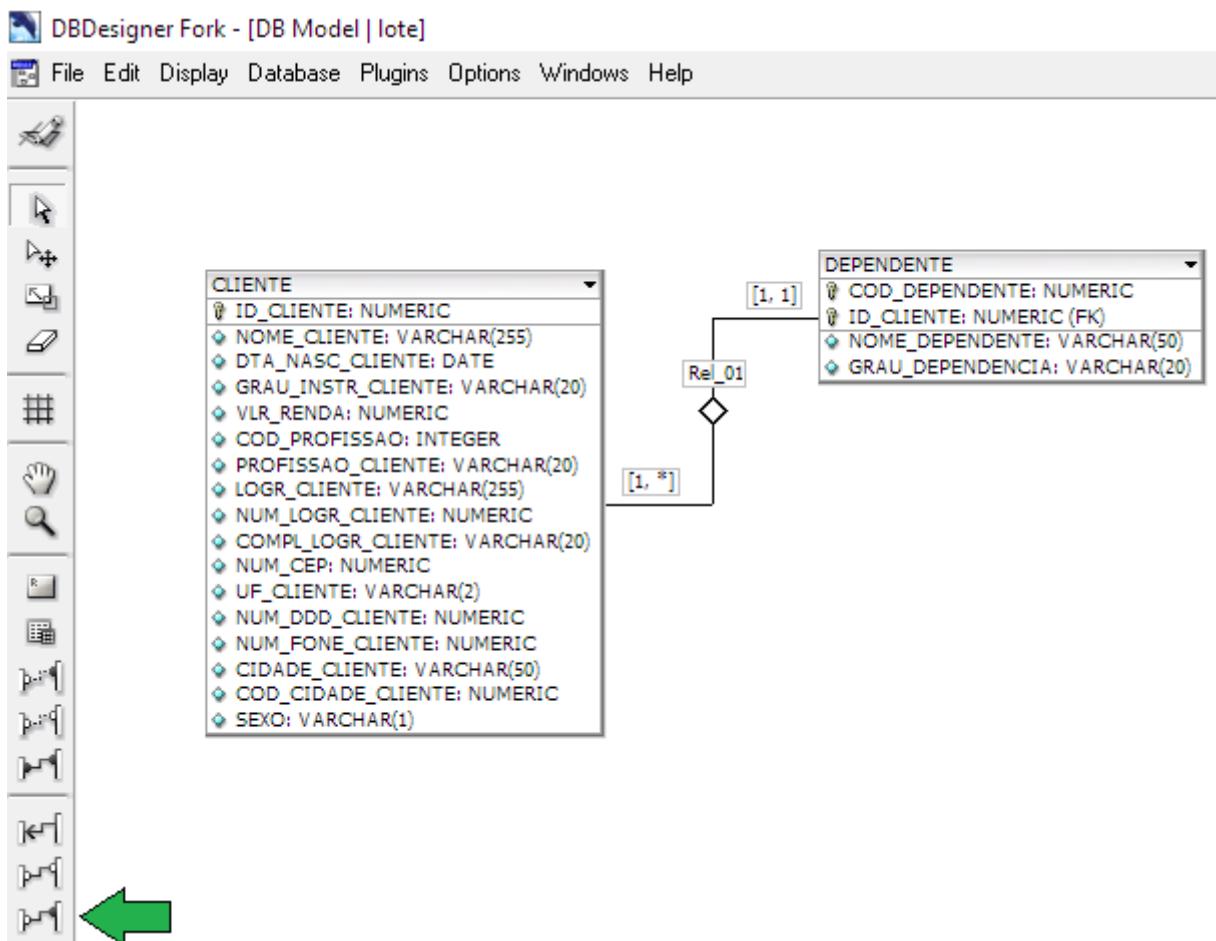


Figura 70. Utilizando o relacionamento New 1:N Relation
Fonte: Autor.

Na figura 70 temos o relacionamento criado, observe que o DBDesigner Fork atribuiu por padrão o nome “Rel_01”. Para alterar, devemos dar um duplo clique sobre do nome do relacionamento e na tela “Relation Editor” alteramos o nome para “POSSUI”, conforme a figura 71.

Utilizamos o relacionamento 1:N, observe na figura 71 que o tipo de relacionamento é exibido ao lado, em “Relation Kind”, este pode ser alterado ao clicar sobre a caixa de listagem, que em seguida exibirá os tipos de relacionamentos disponíveis. No modelo relacional, os tipos de relacionamentos podem ser 1-1, 1-N ou N-M.

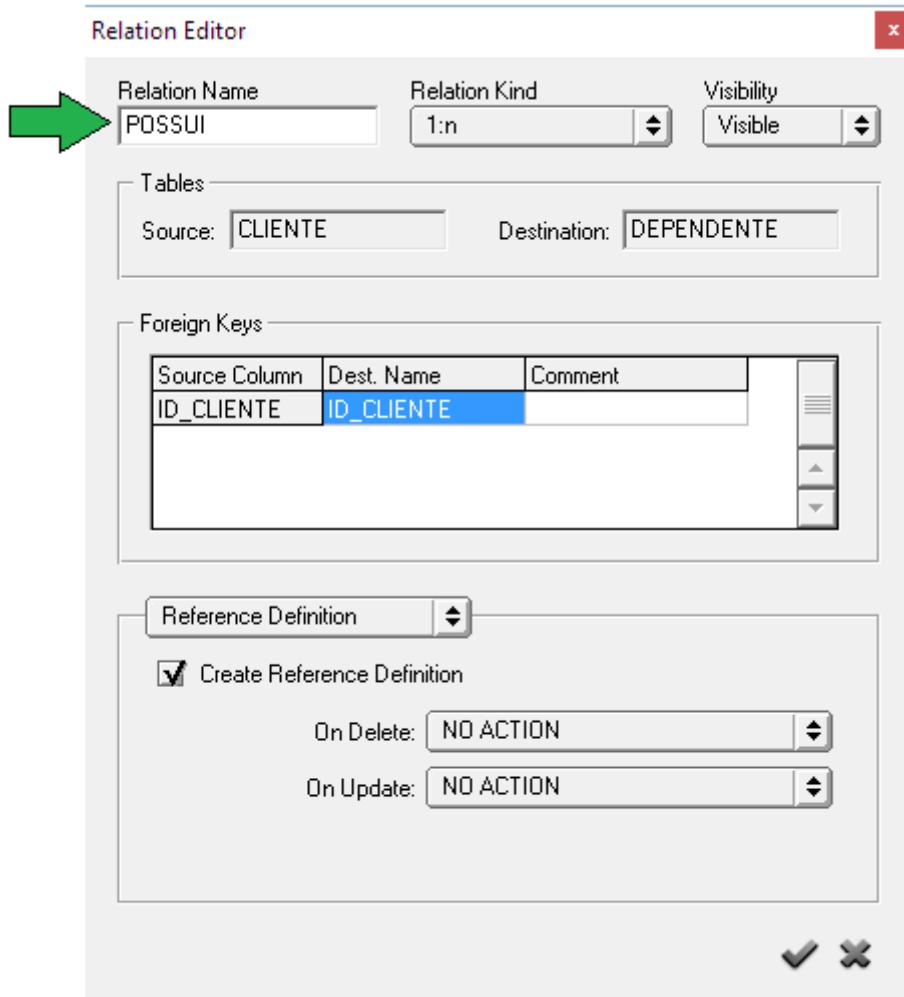


Figura 71. Configurando as propriedades do relacionamento
Fonte: Autor.

Em relacionamentos 1-n, deve ser incluída uma coluna na tabela do lado N do relacionamento que será chave estrangeira e representará a chave primária da tabela do lado de cardinalidade 1. Observe que o DBDesigner Fork automaticamente fez isso na tabela DEPENDENTE, adicionando o atributo Id_Cliente e identificando-o como Foreign Key (FK) e parte da chave primária composta, conforme mostrado na figura 72.

A chave estrangeira determina a integridade referencial do modelo, isso porque ela garante que nenhum valor para o atributo Id_Cliente seja adicionado na tabela DEPENDENTE sem que ele exista na tabela CLIENTE. Em outras palavras, ela obriga que o registro inserido na tabela exista na tabela correspondente.

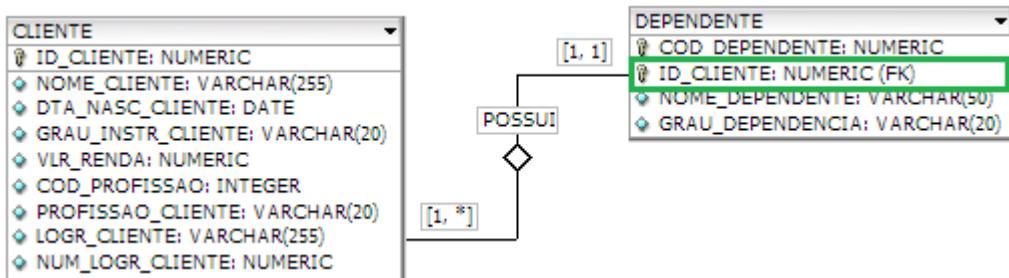


Figura 72. Foreign Key criada automaticamente pelo DBDesigner Fork
Fonte: Autor.

Conforme mostra a figura 72, para que as entidades cliente e dependente participem no conjunto de relacionamento “Possui”, cada entidade cliente deve ter no mínimo uma entidade dependente associada, no entanto, também pode ter mais de uma entidade dependente associada a ele, conforme descreve a notação ao lado da tabela cliente ([1, *]). Conforme a notação representada ao lado da tabela dependente, cada entidade dependente para participar no conjunto de relacionamento deve estar associada a no mínimo e no máximo uma entidade no conjunto de entidades cliente ([1, 1]). Esta restrição é a mesma descrita no modelo conceitual, aqui mencionada apenas para evidenciar que a leitura da notação é feita de forma inversa à descrita anteriormente no modelo conceitual de Chen.

Temos o modelo lógico das tabelas CLIENTE e DEPENDENTE, agora, criamos as tabelas PROPOSTA e VENDEDOR, padronizamos os nomes dos atributos e configuramos seus atributos chave como chave primária, conforme descreve a figura 73. Observe que na tabela PROPOSTA não criamos os atributos Id_Cliente e Cod_Vendedor, isso porque o DBDesigner Fork criará estes atributos como chave estrangeira, ao configurarmos o relacionamento entre as tabelas.

PROPOSTA
NUM_PROPOSTA: INTEGER
DTA_INCLUSAO: DATE
VLR_PRINCIPAL: NUMERIC
VLR_ENTRADA: NUMERIC
QTDE_PRESTACOES: NUMERIC
VLR_DESP_CARTORIO: NUMERIC
VLR_RENDER_DECL: NUMERIC
STS_PROPOSTA: NUMERIC
DTA_ALT: DATE

VENDEDOR
COD_VENDEDOR: NUMERIC
NOME_VENDEDOR: VARCHAR(100)
NUM_DDD_VENDEDOR: NUMERIC
NUM_FONE_VENDEDOR: NUMERIC
NUM_PERC_COMISSAO: NUMERIC(6,2)

CLIENTE
ID_CLIENTE: NUMERIC
NOME_CLIENTE: VARCHAR(255)
DTA_NASC_CLIENTE: DATE
GRAU_INSTR_CLIENTE: VARCHAR(20)
VLR_RENDER: NUMERIC
COD_PROFISSAO: INTEGER
PROFISSAO_CLIENTE: VARCHAR(20)
LOGR_CLIENTE: VARCHAR(255)
NUM_LOGR_CLIENTE: NUMERIC
COMPL_LOGR_CLIENTE: VARCHAR(20)
NUM_CEP: NUMERIC
UF_CLIENTE: VARCHAR(2)
NUM_DDD_CLIENTE: NUMERIC
NUM_FONE_CLIENTE: NUMERIC
CIDADE_CLIENTE: VARCHAR(50)
COD_CIDADE_CLIENTE: NUMERIC
SEXO: VARCHAR(1)

Figura 73. Configuração das novas tabelas Proposta e Vendedor
Fonte: Autor.

Na figura 74, utilizamos o relacionamento “New 1:n Non-Identifying-Relation”. Clicamos no ícone do relacionamento, clicamos na tabela CLIENTE e em seguida clicamos na tabela PROPOSTA. Logo após, relacionamos VENDEDOR com PROPOSTA, para isso, clicamos no ícone do relacionamento, clicamos na tabela VENDEDOR e em seguida clicamos na tabela PROPOSTA. Mantemos os mesmos nomes de relacionamento descritos no modelo conceitual.

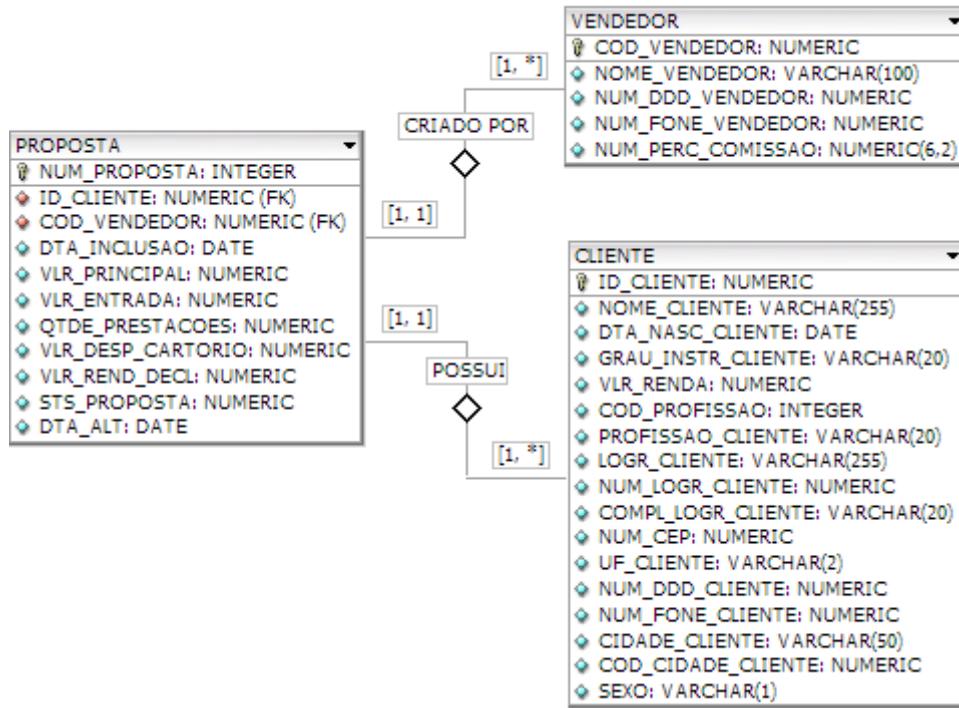


Figura 74. Relacionamento entre Cliente, Proposta e Vendedor

Fonte: Autor.

Conforme mostra a figura 75, criamos a tabela CONTRATO, e configuramos sua chave primária Num_Contrato. Relacionaremos PROPOSTAS com CONTRATO, desta vez utilizando “New 1:1 Non-Identifying-Relation”, conforme mostrado na figura 76.

CONTRATO
NUM_CONTRATO: NUMERIC
DTA_EMISSAO: DATE
VLR_TOTAL_CONTRATO: NUMERIC(14,2)
QTDE_PARCELAS: NUMERIC
VLR_PARCELA: NUMERIC(14,2)
NUM_TX_CONTRATO: NUMERIC

Figura 75. Tabela Contrato e sua chave primária Num_contrato

Fonte: Autor.

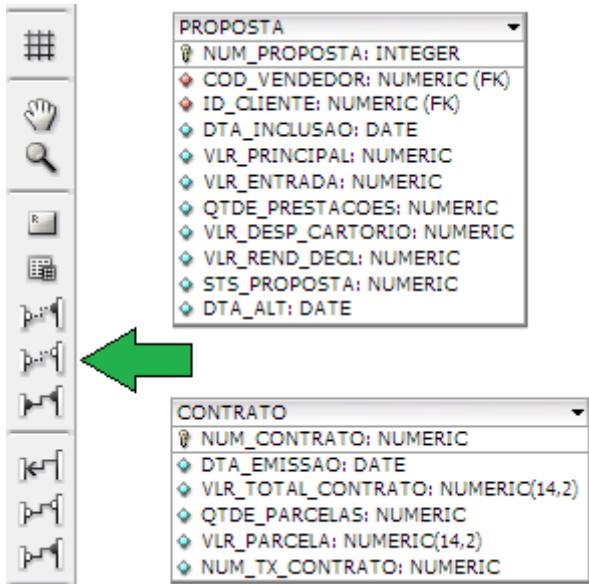


Figura 76. Ícone de relacionamento “New 1:1 Non-Identifying-Relation”

Fonte: Autor.

A entidade CONTRATO possui sua própria chave, Num_Contrato, portanto não é uma entidade fraca, no entanto, cada entidade contrato depende de outra entidade para existir, essa dependência de existência está relacionada à entidade PROPOSTA.

Neste minimundo, sabemos que uma proposta ao ser efetivada gera um contrato, logo, esperamos que toda entidade do conjunto de entidades CONTRATO esteja relacionada a uma entidade do conjunto de entidades PROPOSTA por meio do relacionamento “Efetiva”. Deste modo, a participação da entidade CONTRATO no conjunto de relacionamento “Efetiva” é total. Por outro lado, um cliente pode optar por não efetivar uma proposta, consequentemente, é possível que apenas algumas das entidades no conjunto de entidades PROPOSTA estejam relacionadas ao conjunto de entidades CONTRATO por meio do relacionamento “Efetiva” e, portanto, a participação de PROPOSTA no conjunto de relacionamento “Efetiva” é parcial.

Nos relacionamentos 1-1 a chave primária da tabela que tem participação parcial na relação será a chave estrangeira da entidade que tem participação total no conjunto de relacionamento. Como estamos utilizando na criação do relacionamento o tipo de relacionamento "New 1:1 Non-Identifying-Relation", o DBDesigner Fork coloca uma cópia da chave primária de uma das tabelas como chave estrangeira da outra. Portanto, determinamos que na tabela CONTRATO será criada a chave estrangeira, em seguida clicamos na tabela PROPOSTA, que cederá

a cópia da chave primária e clicamos na tabela Contrato, que receberá a chave estrangeira, conforme a figura 77.

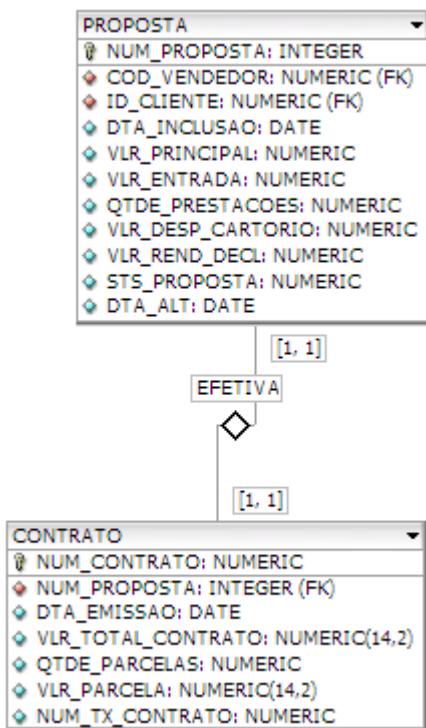


Figura 77. Relacionamento entre Proposta e Contrato
Fonte: Autor.

Conforme mostra a figura 78, criamos a tabela PARCELA, sua chave primária será uma chave composta por Num_Contrato e Num_Parcela.

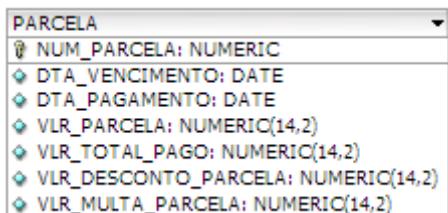


Figura 78. Tabela Parcera e seus atributos
Fonte: Autor.

Relacionamos PARCELA com CONTRATO, para isso clicamos no ícone "New 1:N Relation", em seguida clicamos na tabela CONTRATO e logo após, na tabela PARCELA. A chave primária da tabela CONTRATO é adicionada como chave estrangeira na tabela PARCELA, conforme mostrado na figura 79.

Observe que agora a tabela PARCELA possui uma chave primária composta por Num_Contrato e Num_Parcela.

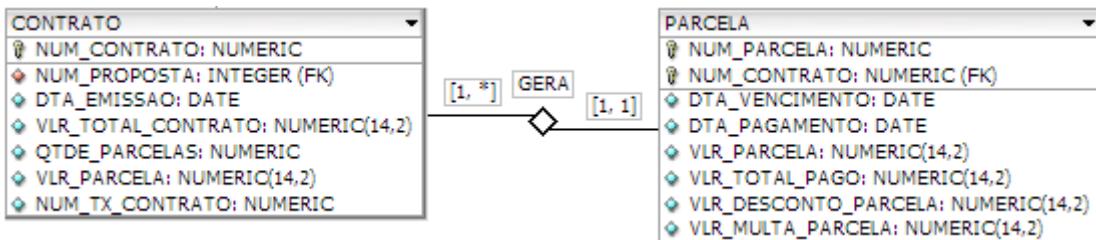


Figura 79. Relacionamento entre Contrato e Parcela
Fonte: Autor.

De acordo com o modelo conceitual, conforme mostra a figura 80, criamos as tabelas ITEM CONTRATO, LOTE e LOTEAMENTO. Os campos que são chave primária de outras tabelas serão criados ao configurarmos o relacionamento entre elas.

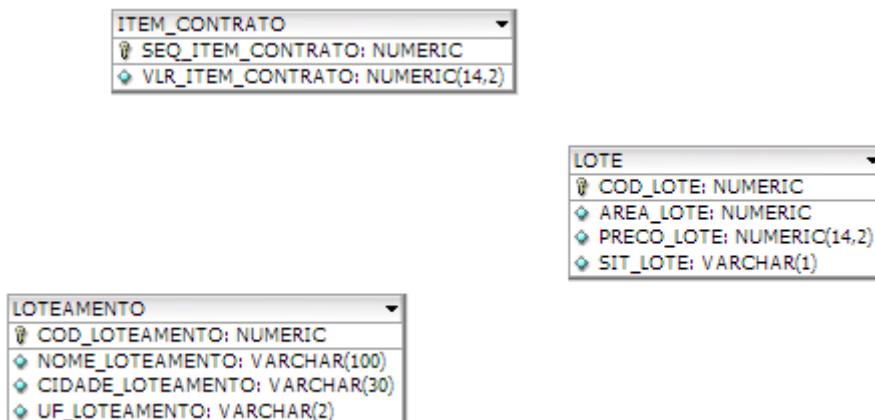


Figura 80. Tabelas Item Contrato, Lote e Loteamento
Fonte: Autor.

Para configurar o relacionamento “Contem” entre CONTRATO e ITEM CONTRATO, clicamos no ícone "New 1:N Relation", em seguida clicamos na tabela CONTRATO e logo após, na tabela ITEM CONTRATO. A chave primária da tabela CONTRATO é adicionada como chave estrangeira na tabela ITEM CONTRATO, conforme mostrado na figura 81.

Observe que agora a tabela ITEM CONTRATO possui uma chave composta por Num_Contrato e Seq_Item_Contrato.

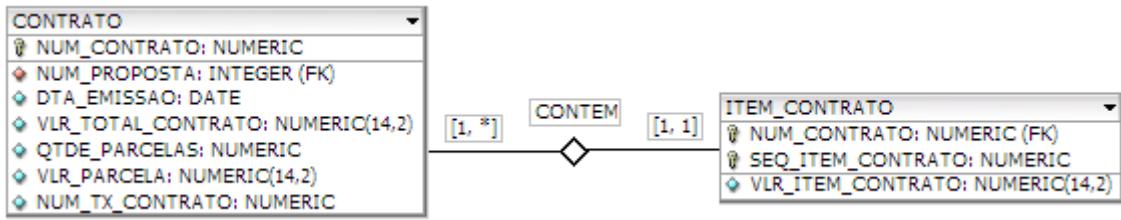


Figura 81. Relacionamento 1:N entre tabelas Contrato e Item Contrato
Fonte: Autor.

Continuando a criação do modelo lógico, relacionamos agora LOTEAMENTO e LOTE, para isso clicamos novamente no ícone "New 1:N Relation", em seguida clicamos na tabela LOTEAMENTO e logo após, na tabela LOTE. A chave primária da tabela LOTEAMENTO é adicionada como chave estrangeira na tabela LOTE, conforme mostrado na figura 82.

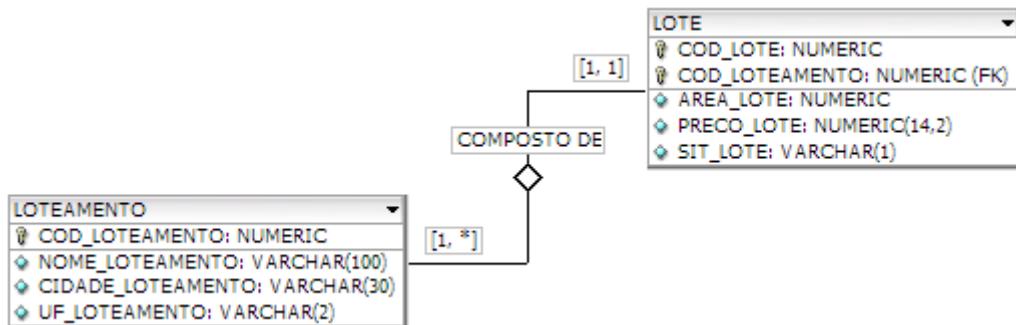


Figura 82. Relacionamento 1:N entre tabelas loteamento e lote
Fonte: Autor

Para relacionar as tabelas ITEM CONTRATO e LOTE, na criação deste relacionamento utilizaremos novamente o tipo "New 1:1 Non-Identifying-Relation". Clicamos no ícone do relacionamento, clicamos na tabela LOTE e em seguida clicamos na tabela ITEM CONTRATO.

Os atributos Cod_lote e Cod_Loteamento da tabela LOTE serão adicionados como chaves estrangeiras na tabela ITEM CONTRATO, conforme mostrado na figura 83.

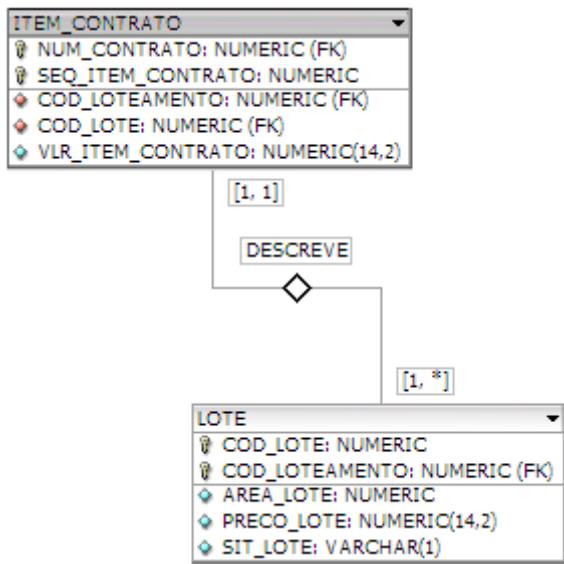


Figura 83. Relacionamento entre Item Contrato e Lote

Fonte: Autor.

A figura 84 mostra o modelo lógico inicial do projeto.

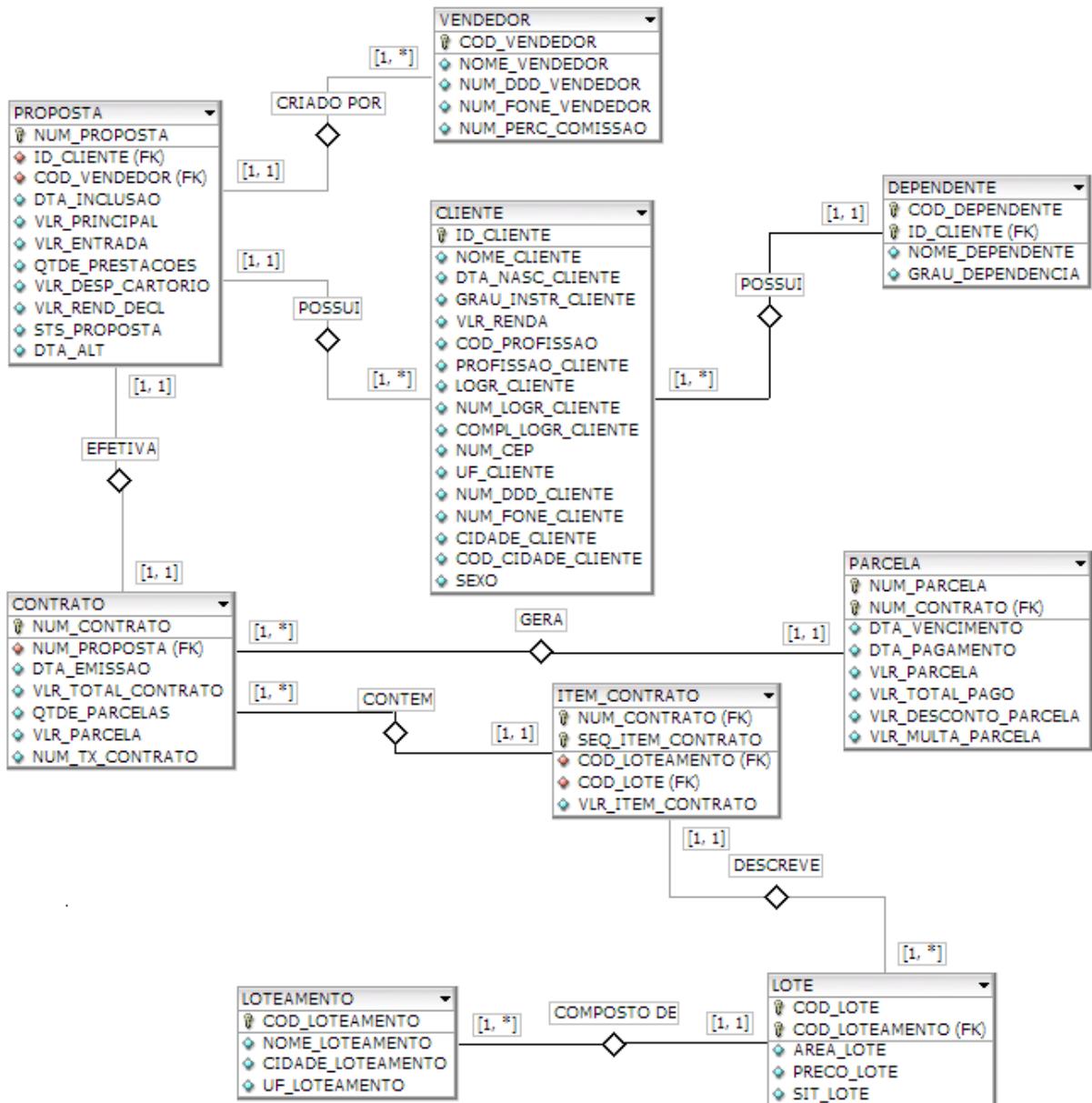


Figura 84. Modelo Lógico Inicial

Fonte: Autor.

Observando a figura 84, nota-se que nem todos os atributos descritos para cada entidade no modelo conceitual estão representados no modelo lógico, isso se deve ao fato de que no modelo lógico a preocupação em criar um banco de dados íntegro e normalizado nos leva a eliminar redundâncias. Podemos observar que nas tabelas modeladas cada atributo possui um único tipo de dado e um único valor, ou seja, não temos grupos repetidos de itens de dados, por este motivo o modelo está normalizado na 1FN. Para testar a 2FN analisamos as tabelas DEPENDENTE, PARCELA, ITEM CONTRATO e LOTE, que são as tabelas que possuem chave primária composta. Nestas tabelas, como precisamos de todos os campos da chave

primária composta para determinarmos os valores dos campos não chave, podemos certificar que estão na 2FN. A próxima e última etapa é validar a 3FN. A tabela CLIENTE possui dois campos diferentes que determinam o valor do atributo profissão, o mesmo ocorre com a cidade do cliente, neste caso há dependências transitivas. Para que o modelo esteja normalizado na 3FN precisamos ajustar a tabela CLIENTE.

Observe o modelo lógico normalizado na 3FN demonstrado na figura 85.

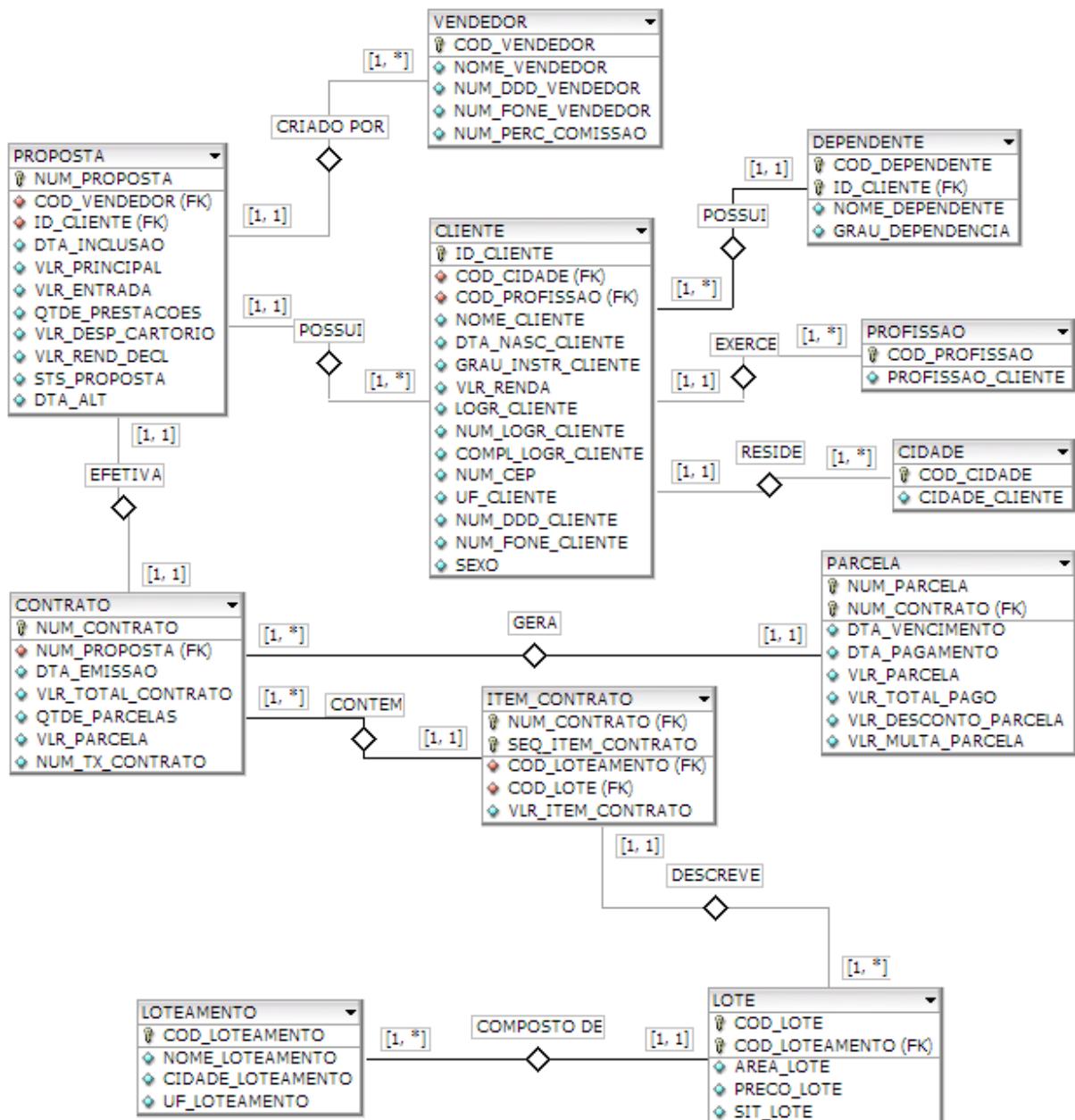


Figura 85. Modelo Lógico atual normalizado na 3FN
Fonte: Autor.

3.6 Modelo físico

Chegamos ao modelo físico, ou modelo de dados de baixo nível. Conforme visto até aqui, criamos o modelo conceitual, que serviu de base para criação do modelo lógico, em seguida configuramos o modelo lógico e ajustamos aspectos inerentes ao negócio, mas voltados à integridade dos dados na aplicação do banco de dados relacional. Nesta etapa faremos o mapeamento do modelo lógico em um esquema físico de acordo com o SGBD específico, este é o nível dos códigos, da criação dos objetos no banco de dados.

3.6.1 SGBD Oracle Express Edition

O sistema gerenciador de banco de dados será do tipo relacional, para este trabalho utilizaremos um SGBD da Oracle. Embora informações relevantes sejam registradas a seguir, não descreveremos a instalação em si do SGBD.

O Oracle Express Edition é um SGBD que pode ser baixado gratuitamente, é uma versão gratuita do banco de dados relacional, seu arquivo executável é relativamente pequeno (300 Mb), é fácil de instalar, gerenciar e desenvolver. O Software oferece uma interface baseada em navegador para administrar o banco de dados, criar tabelas, executar consultas, scripts SQL, exibições e outros objetos de banco de dados.

A versão Express Edition, no entanto, dá direito a apenas um banco de dados, que é instalado juntamente com o software. O nome desse banco de dados é XE, é o nome de serviço padrão, tem limite máximo de 11Gb na versão 11g e embora você possa criar mais de um esquema, não é possível criar outro banco de dados. Um esquema é o conjunto de todos os objetos pertencentes à uma conta de usuário criada no banco de dados. O Oracle Express Edition 11g release 2 é a versão mais recente, está disponível para sistemas Windows x64, Windows x32 e Linux x64.

Para baixar o Sistema gerenciador de banco de dados da Oracle basta acessar 'www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html'. Conforme mostra a figura 86, a página de download será aberta.

Para download do software é preciso aceitar o acordo de licença e conforme o sistema operacional, baixar uma das três versões disponíveis. Será solicitado login na conta gratuita Oracle, caso não possua, basta clicar em “Criar Conta”, conforme mostra a figura 87.

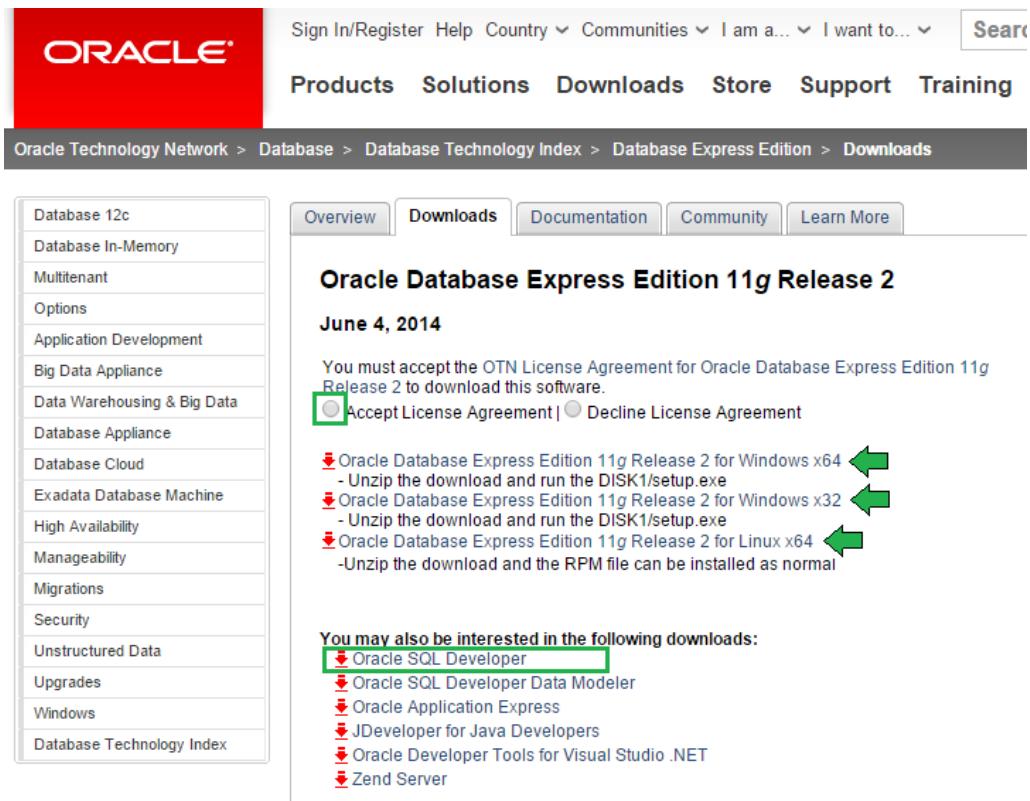


Figura 86. Página da Oracle para baixar o SGBD Express Edition
Fonte: Autor.



Figura 87. Conta Gratuita Oracle para baixar aplicações
Fonte: Autor.

3.6.2 Oracle SQL Developer

A Oracle SQL Developer é uma ferramenta gráfica gratuita que aumenta a produtividade e simplifica as tarefas de desenvolvimento de banco de dados.

Com o SQL Developer, é possível procurar objetos no banco de dados, executar instruções e scripts SQL, editar e depurar declarações PL/SQL além de manipular e exportar dados. A ferramenta se conecta a bancos de dados Oracle e a bancos de dados de terceiros (Não Oracle), permitindo visualizar dados e Metadados, possibilitando inclusive migrar esses bancos de dados para Oracle.

Para fazer o download do Oracle SQL Developer basta clicar no link destacado na figura 86 e a interface da aplicação é exibida na figura 88.

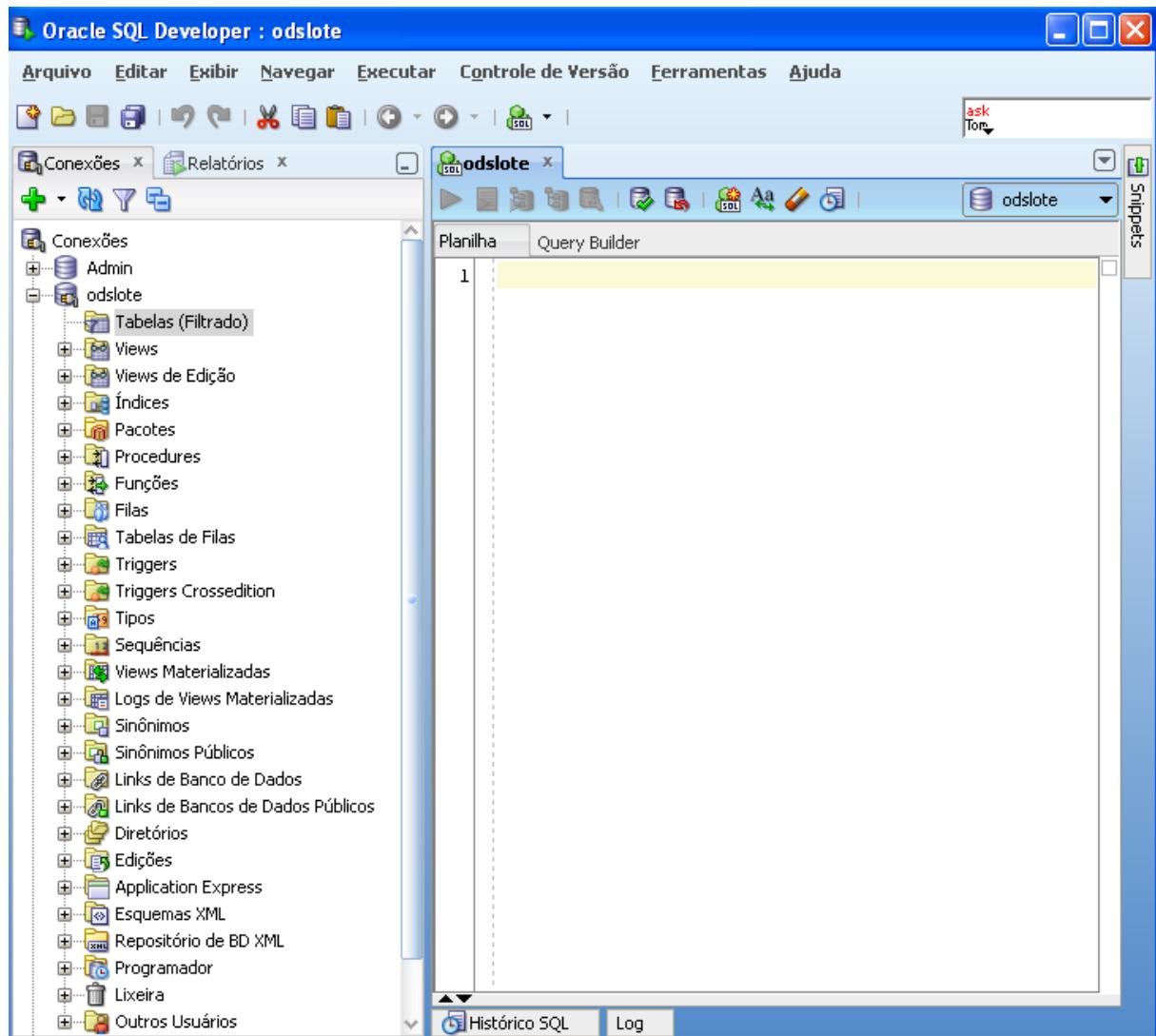


Figura 88. Área de trabalho do Oracle SQL Developer
Fonte: Autor.

3.6.3 DBDesigner e Scripts do modelo físico

Após a instalação e configuração do SGBD foi criado um esquema/usuário que nomeamos como odslove. Neste usuário criaremos as tabelas representadas no modelo lógico.

Após a criação do modelo lógico, o DBDesigner Fork permite exportar o script para criação do modelo físico, para isso, com o modelo aberto, clicamos em “file”, “Export”, “SQL Create Script ...”, conforme mostrado na figura 89.

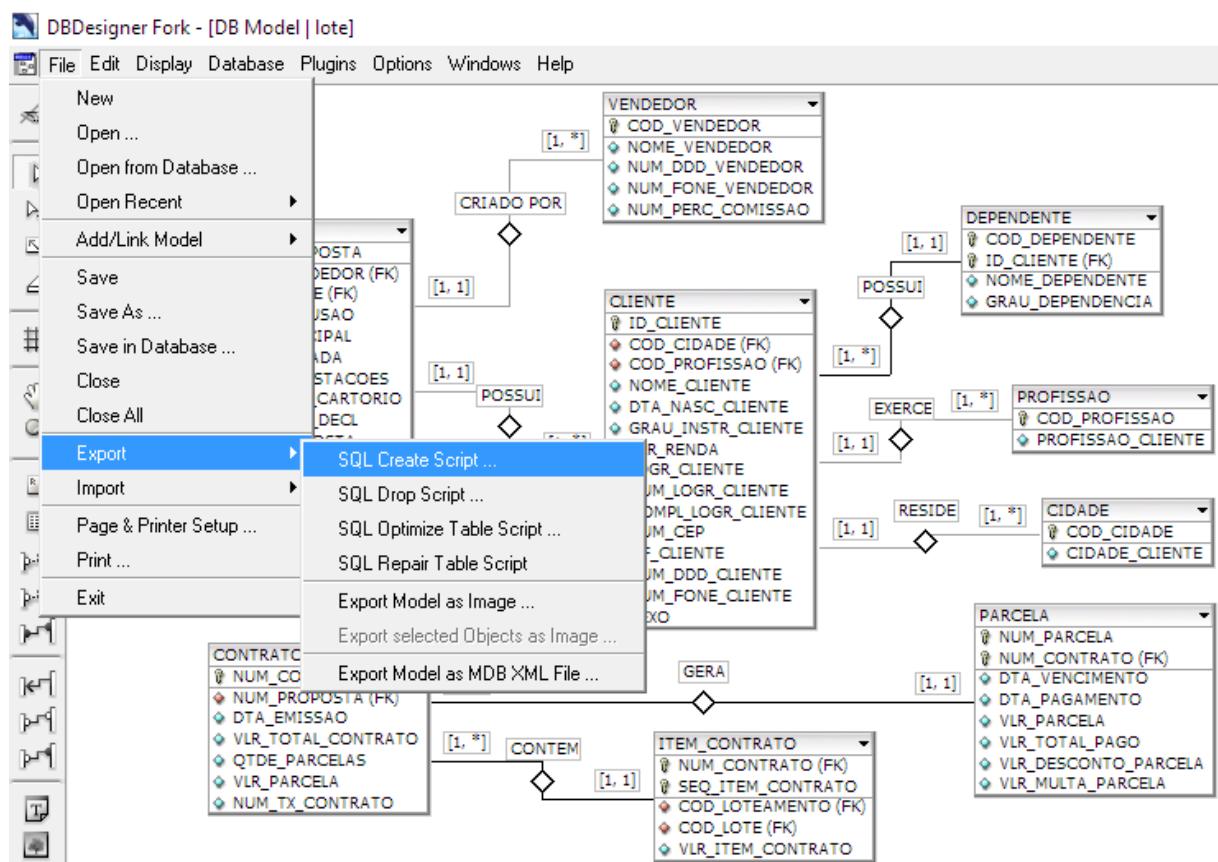


Figura 89. Exportando script para criação dos objetos
Fonte: Autor.

Na janela que abre, clicamos na caixa de seleção “Target Data Base”, nesta seleção escolhemos o banco de dados para o qual desejamos gerar o script, as opções disponíveis são Firebird, SQL Server, MySQL, Oracle, SQLite e PostgreSQL. Como o nosso banco de dados é Oracle, escolhemos esta opção e em seguida desmarcamos a opção “Generate Indexes for FKs”, conforme a figura 90.

Clicamos em “Save Script to File” e escolhemos algum local em disco para armazenar o arquivo com o script de criação dos objetos.

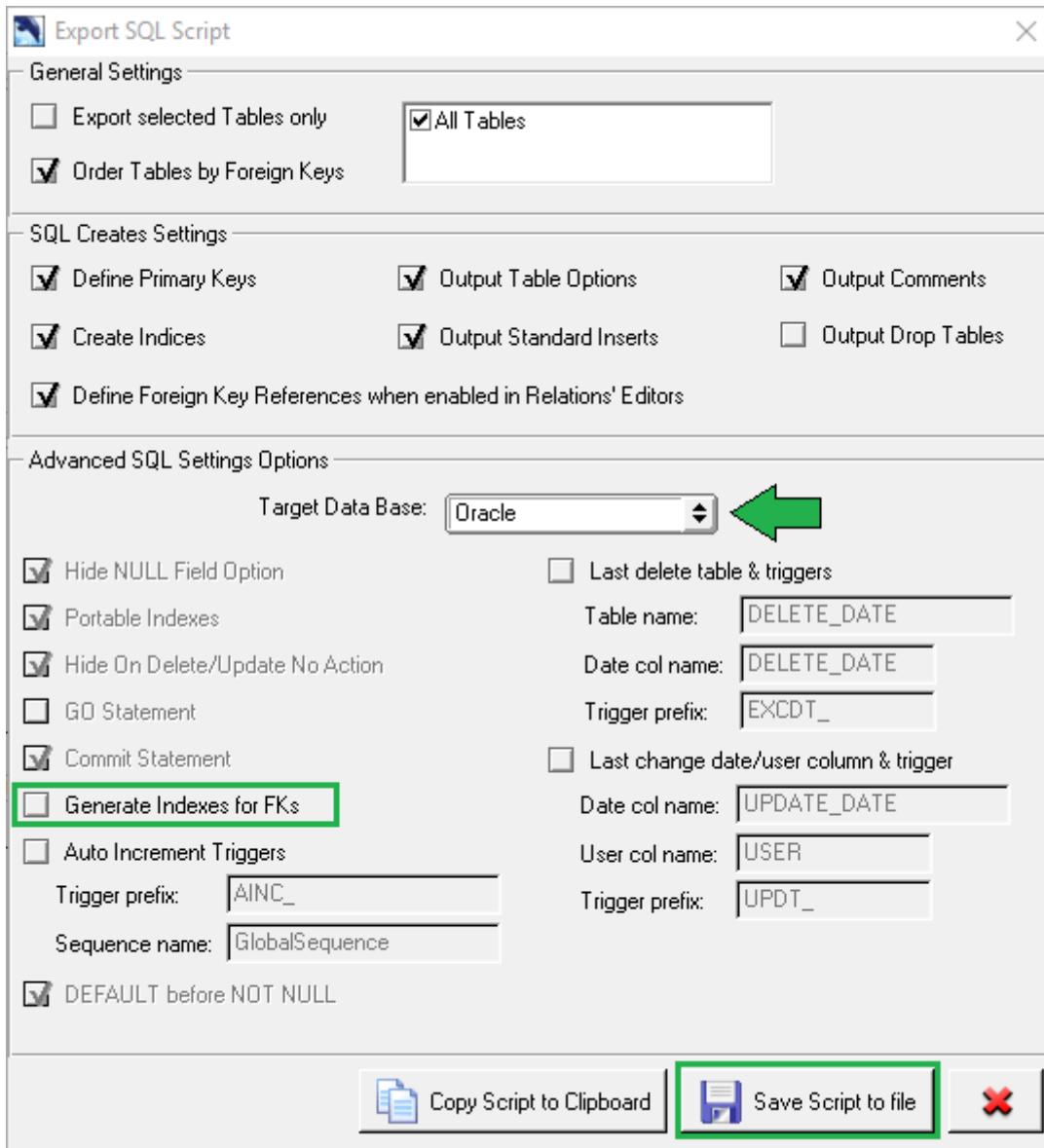
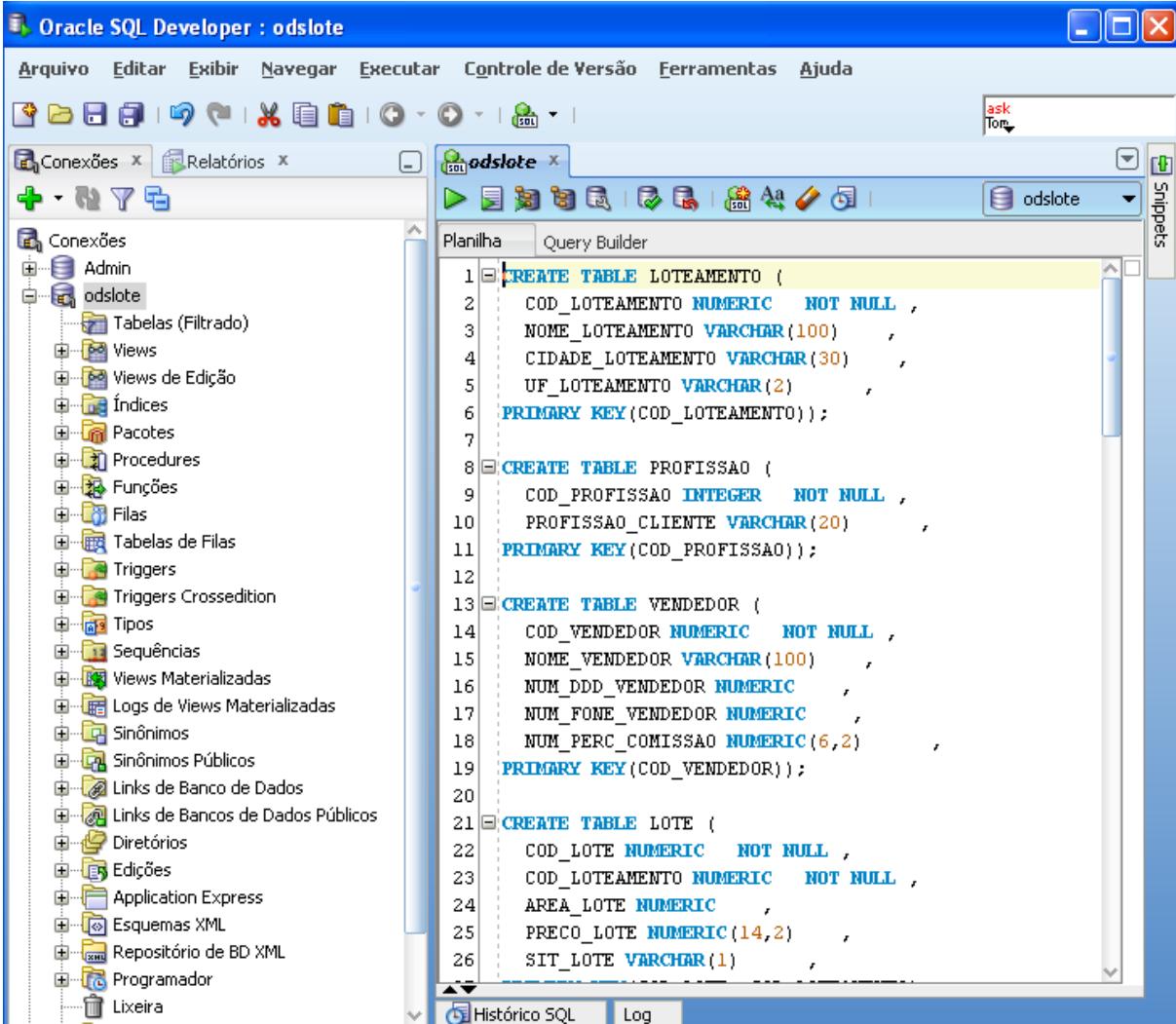


Figura 90. Configurando opções para exportar script de criação dos objetos
Fonte: Autor.

O script é gerado, conforme mostrado no apêndice deste trabalho, observe que o DBDesigner Fork organiza o script na ordem correta para criação dos objetos, primeiro ele cria as tabelas que possuem apenas chave primária, depois ele cria as tabelas que referenciam estas chaves primárias na chave estrangeira. Por exemplo, se você tentar criar primeiro a tabela CLIENTE, haverá retorno do banco de dados com a mensagem de erro na linha de comando, a mensagem de erro informará que a tabela não existe, isso porque a tabela CLIENTE possui como Foreign Key o campo Cod_Profissao, esta chave estrangeira faz referência à tabela PROFISSAO, que obrigatoriamente deve ser criada primeiro. A ordem correta para criação das

tabelas é Loteamento, Profissão, Vendedor, Lote, Cidade, Cliente, Dependente, Proposta, Contrato, Item contrato e Parcela.

Para criar as tabelas no banco de dados, copiamos o script gerado e colamos na planilha SQL do Oracle SQL Developer, conforme a figura 91. Em seguida com o botão direito do mouse clicamos em “Selecionar tudo” e clicamos no ícone para executar a instrução, conforme mostra a figura 92.



```

CREATE TABLE LOTEAMENTO (
    COD_LOTEAMENTO NUMERIC NOT NULL ,
    NOME_LOTEAMENTO VARCHAR(100) ,
    CIDADE_LOTEAMENTO VARCHAR(30) ,
    UF_LOTEAMENTO VARCHAR(2) ,
    PRIMARY KEY(COD_LOTEAMENTO));
;

CREATE TABLE PROFISSAO (
    COD_PROFISSAO INTEGER NOT NULL ,
    PROFISSAO_CLIENTE VARCHAR(20) ,
    PRIMARY KEY(COD_PROFISSAO));
;

CREATE TABLE VENDEDOR (
    COD_VENDEDOR NUMERIC NOT NULL ,
    NOME_VENDEDOR VARCHAR(100) ,
    NUM_DDD_VENDEDOR NUMERIC ,
    NUM_FONE_VENDEDOR NUMERIC ,
    NUM_PERC_COMISSAO NUMERIC(6,2) ,
    PRIMARY KEY(COD_VENDEDOR));
;

CREATE TABLE LOTE (
    COD_LOTE NUMERIC NOT NULL ,
    COD_LOTEAMENTO NUMERIC NOT NULL ,
    AREA_LOTE NUMERIC ,
    PRECO_LOTE NUMERIC(14,2) ,
    SIT_LOTE VARCHAR(1) ,
    );
;
```

Figura 91. Copiando Script para o SQL Developer
Fonte: Autor.

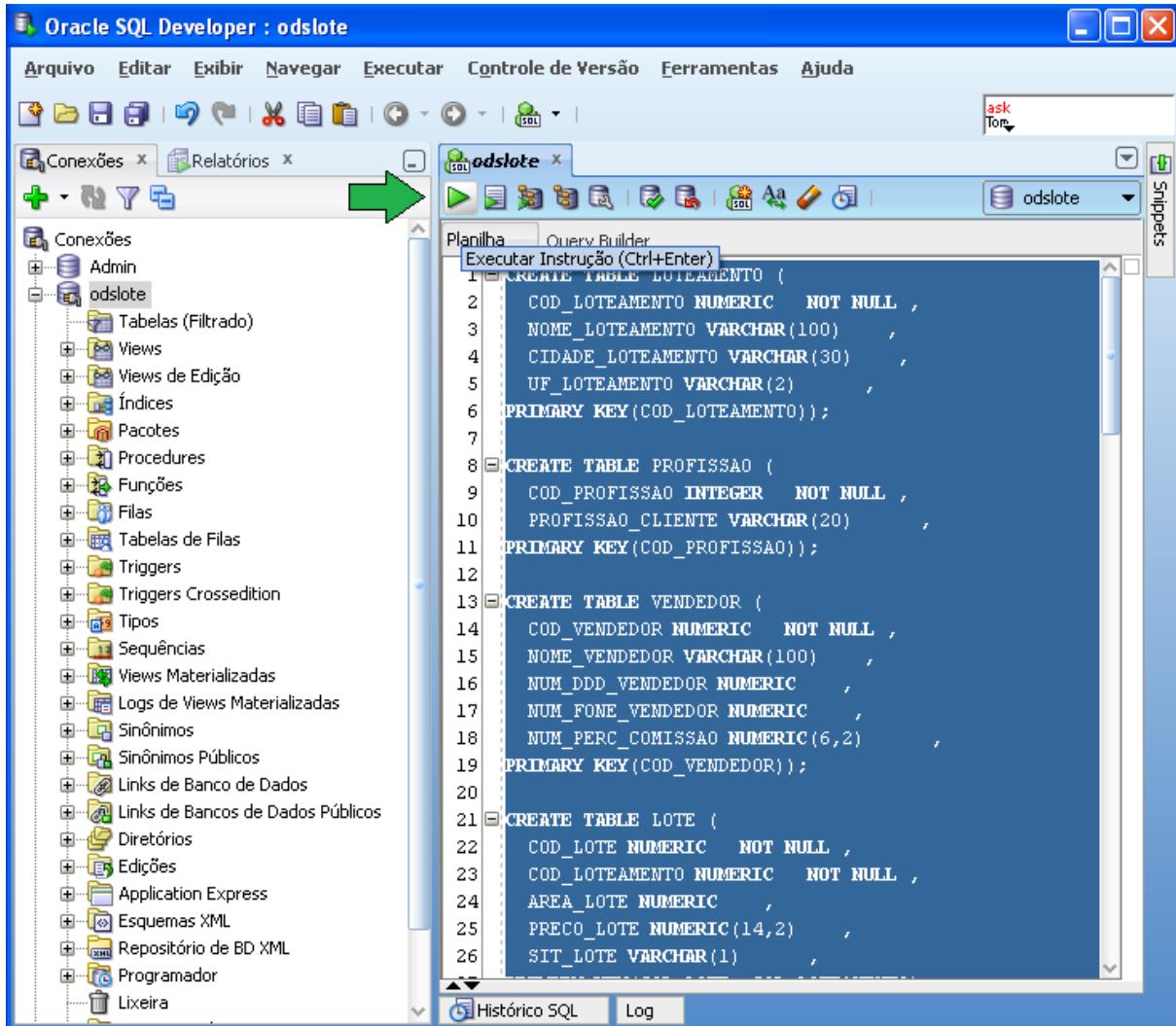


Figura 92. Executando Script para criar objetos
Fonte: Autor.

Após a execução do script gerado pelo DBDesigner Fork todas as tabelas, chaves primárias e chaves estrangeiras são criadas no esquema odslote do banco de dados, conforme mostrado na figura 93.

O próximo passo é popular as tabelas, ou seja, adicionar registros de dados provenientes dos quatro arquivos em formato txt gerados do sistema transacional.

Neste processo extrairemos os dados dos arquivos, transformaremos esses dados conforme as regras de negócio mapeadas no modelo lógico e por fim, faremos a carga dos dados em suas respectivas tabelas, a este processo dá se o nome de ETL, do inglês Extract Transform Load. No processo de ETL várias fontes de dados podem ser utilizadas, no nosso caso utilizaremos apenas quatro, este processo não é um recurso exclusivo para construção de data warehouses e/ou data marts, conforme descrito até aqui estamos criando um banco de dados operacional.

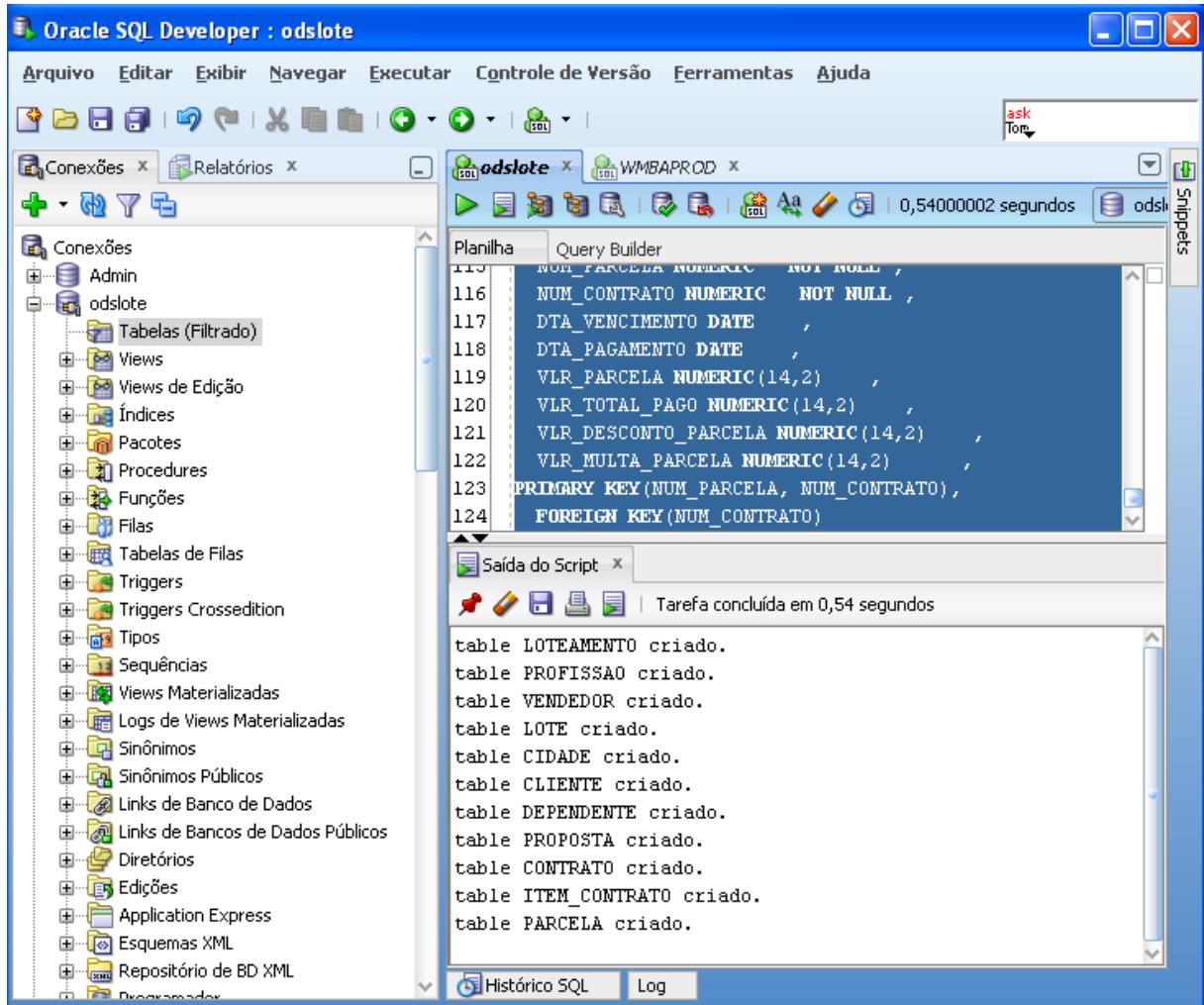


Figura 93. Tabelas criadas pelo Oracle SQL Developer
Fonte: Autor.

3.6.4 ETL com Pentaho Data Integration

Pentaho Data Integration ou PDI, também chamado Kettle, é um componente da Pentaho responsável pela extração, transformação e processos de carga (ETL).

Embora ferramentas de ETL sejam mais frequentemente utilizadas em ambientes de armazéns de dados, PDI pode também ser usado para outros fins, tais como, migrar dados entre aplicações ou bases de dados, exportar dados de bancos de dados para arquivos simples, carregar dados maciçamente em bancos de dados, limpeza de dados e integração de aplicações.

Como uma ferramenta de ETL, é a ferramenta de código aberto mais popular disponível. PDI suporta uma vasta gama de formatos de entrada e saída, incluindo arquivos de texto, planilhas de dados e bancos de dados comerciais e livres. Além

disso, a capacidade de transformação do PDI permite manipular os dados com muito poucas limitações.

PDI é fácil de usar, cada processo é criado com uma ferramenta gráfica onde você especifica o que fazer sem necessidade de escrever código para indicar como fazê-lo, isso porque possui uma rica biblioteca de componentes construídos para acessar e transformar dados de inúmeras fontes de dados. Permite criar transformações dinâmicas, utilizando variáveis para determinar mapeamentos de campo, validação e regras de execução. Possui depurador integrado para testes e ajustes na execução do trabalho, além de permitir agendamento para execução automática. Pode ser usado como uma aplicação autónoma, ou pode ser utilizado como parte da suíte completa Pentaho (Roldán María C).

3.6.5 Carregando dados dos arquivos para as Tabelas

Ao determinarmos os tipos de relacionamentos no modelo lógico, garantimos a integridade das informações fazendo o uso de Foreign Keys, ou chaves estrangeiras, isso porque os valores presentes nas colunas definidas como chave estrangeira devem ter um correspondente nas tabelas que possuem estas colunas como chave primária, caso contrário, durante o processo de inserção ou atualização, o banco de dados retorna uma mensagem de erro, desta forma as restrições de chave estrangeira identificam os relacionamentos entre as tabelas e asseguram que a integridade referencial seja mantida. Em outras palavras, não haverá registros sem correspondente ou duplicados no banco de dados.

Para que o processo de ETL seja realizado com sucesso, a ordem correta de processamento deve ser executada, ou seja, não podemos popular a tabela CLIENTE sem antes popular a tabela PROFISSAO e CIDADE, isso porque acabaria gerando um erro por violação de integridade referencial e o processo seria abortado.

Não será abordado neste trabalho detalhes de utilização da ferramenta PDI para construção das transformações.

Para o processo de carga dos arquivos, as telas mostradas servirão de base para um breve entendimento das etapas de leitura, transformação e carga dos dados.

Como são arquivos provenientes de um sistema transacional, pouco tratamento se faz necessário, diferentemente de quando se carrega dados provenientes de arquivos e planilhas de input manual.

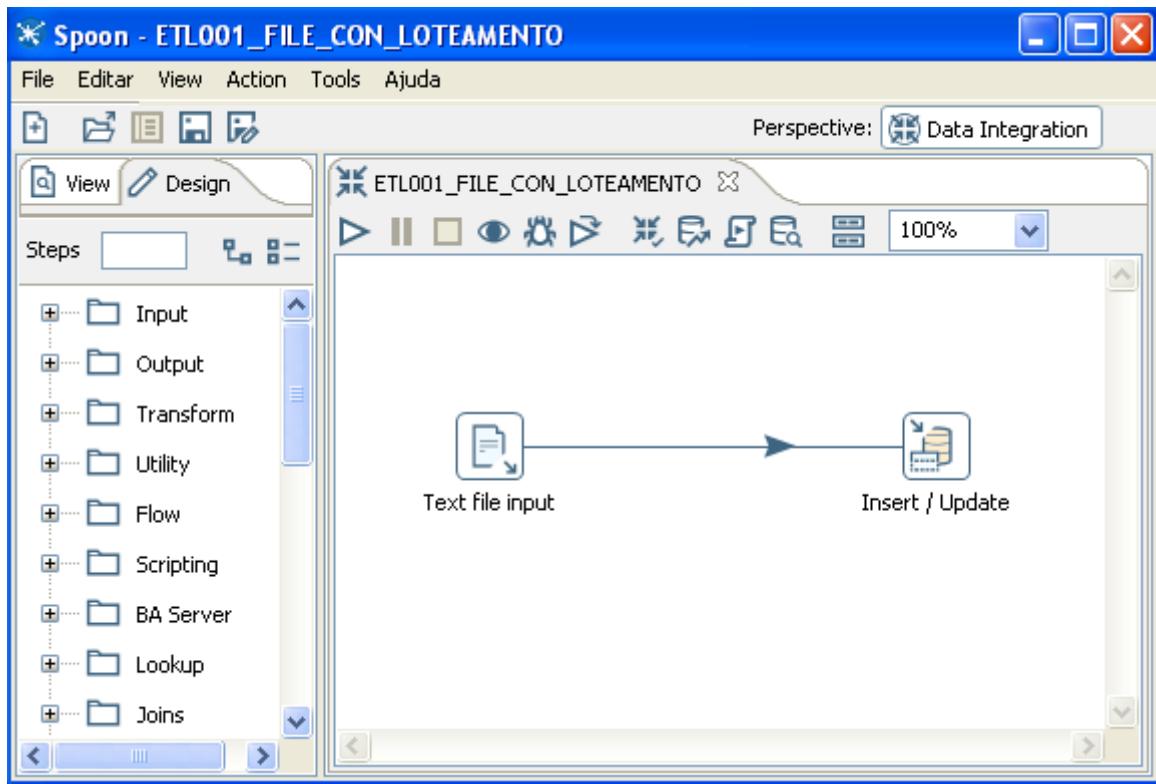


Figura 94. Carga de dados na Tabela Loteamento
Fonte: Autor.

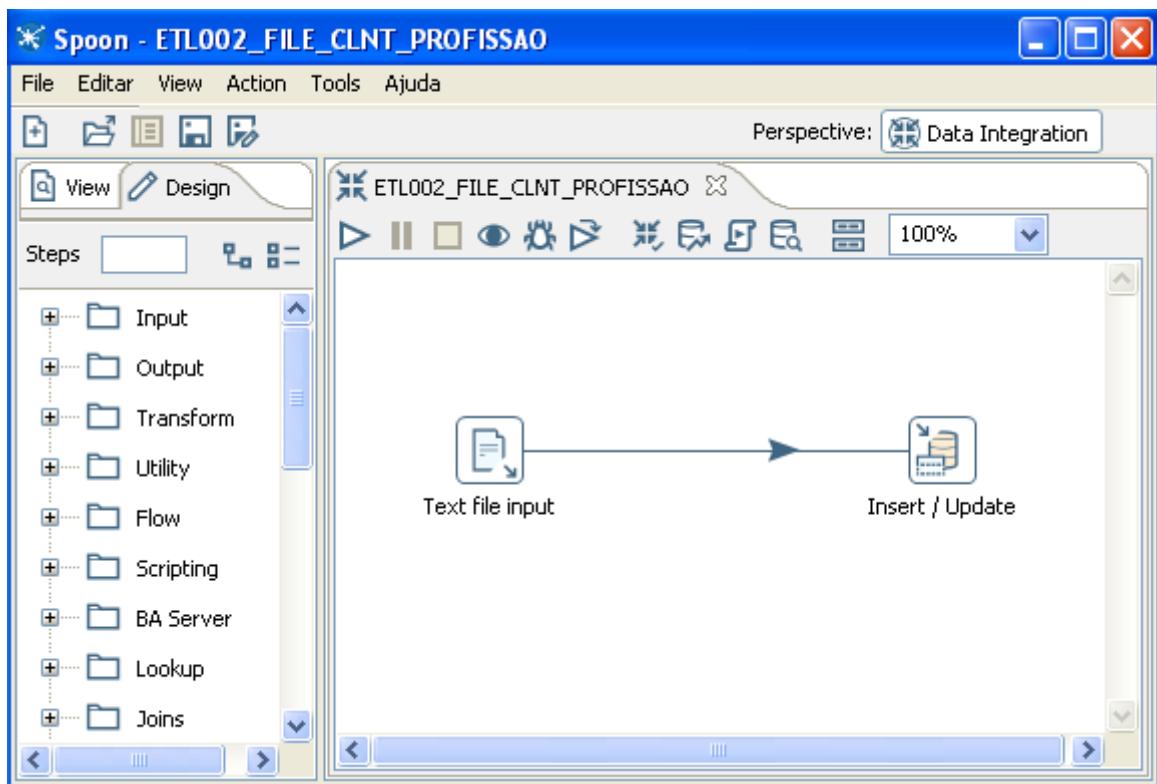


Figura 95. Carga de dados na Tabela Profissao
Fonte: Autor.

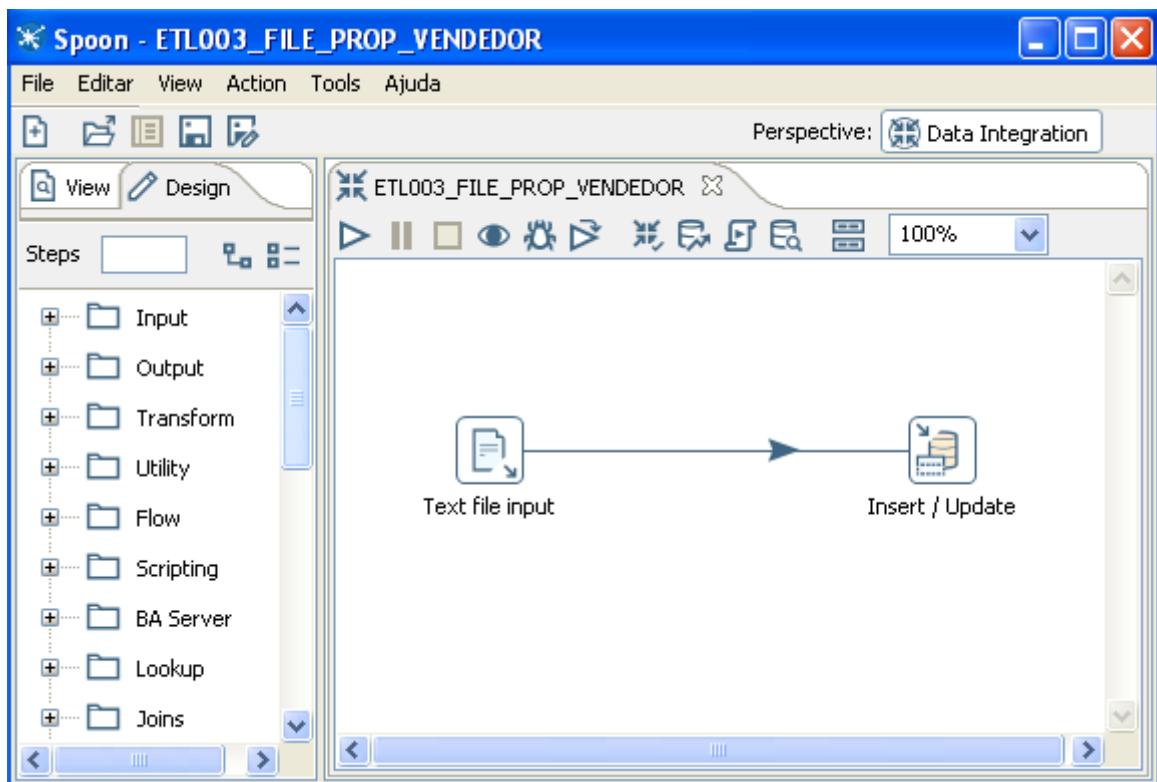


Figura 96. Carga de dados na Tabela Vendedor
Fonte: Autor.

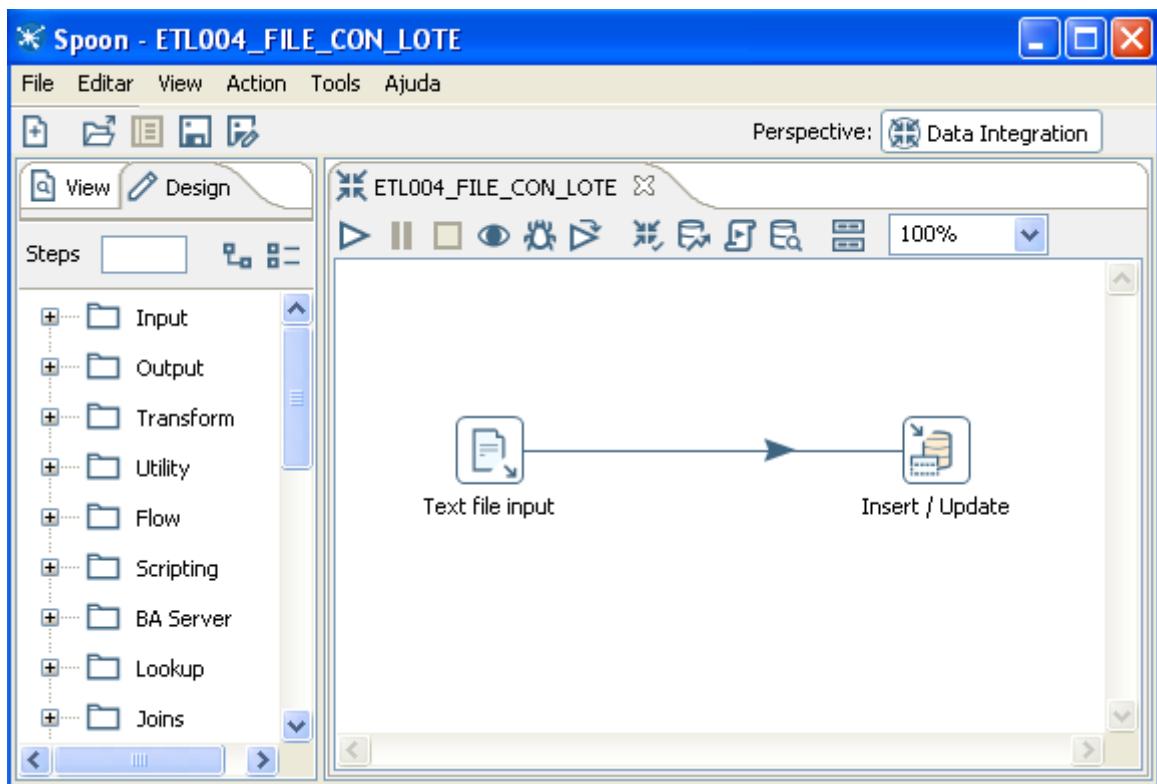


Figura 97. Carga de dados na Tabela Lote

Fonte: Autor.

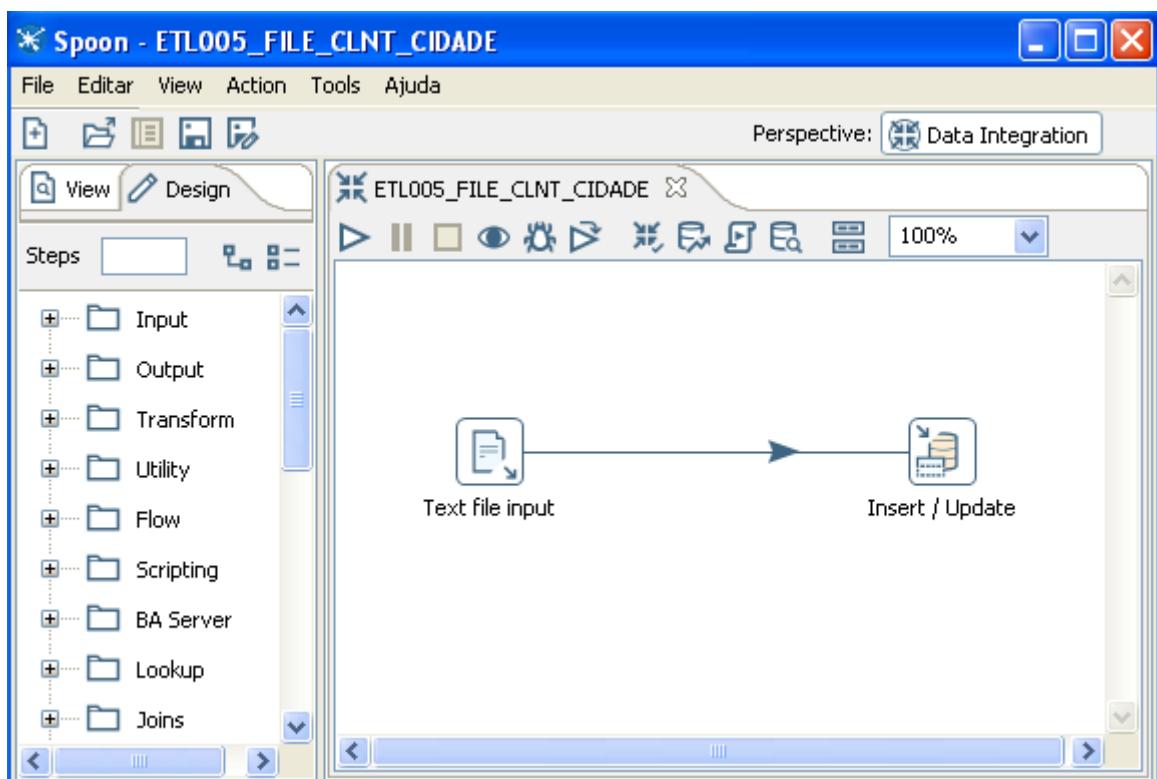


Figura 98. Carga de dados na Tabela Cidade

Fonte: Autor.

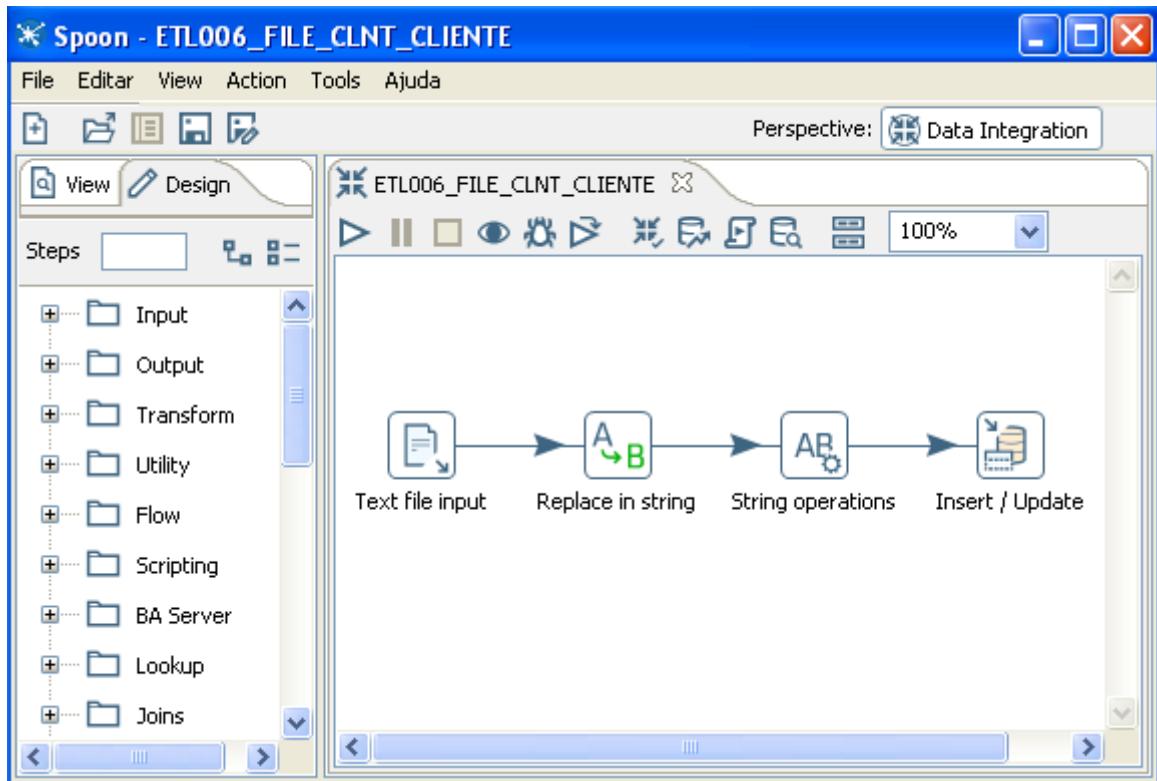


Figura 99. Carga de dados na Tabela Cliente

Fonte: Autor.

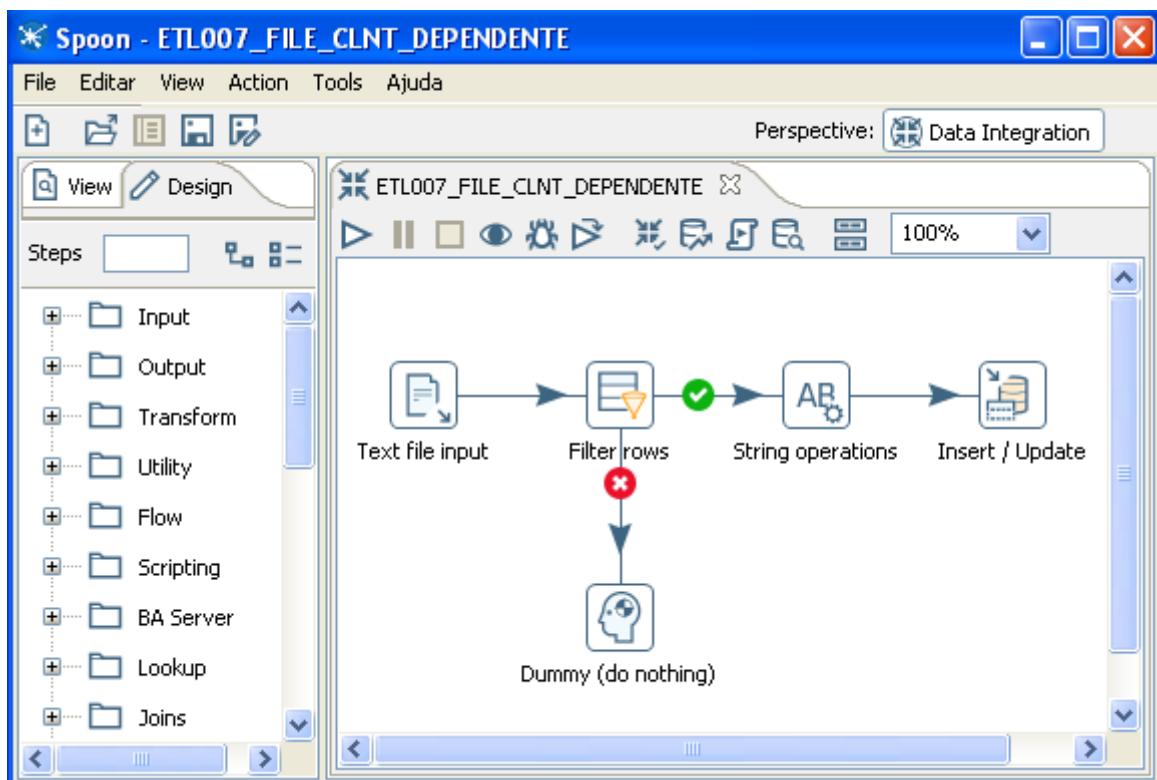


Figura 100. Carga de dados na Tabela Dependente

Fonte: Autor.

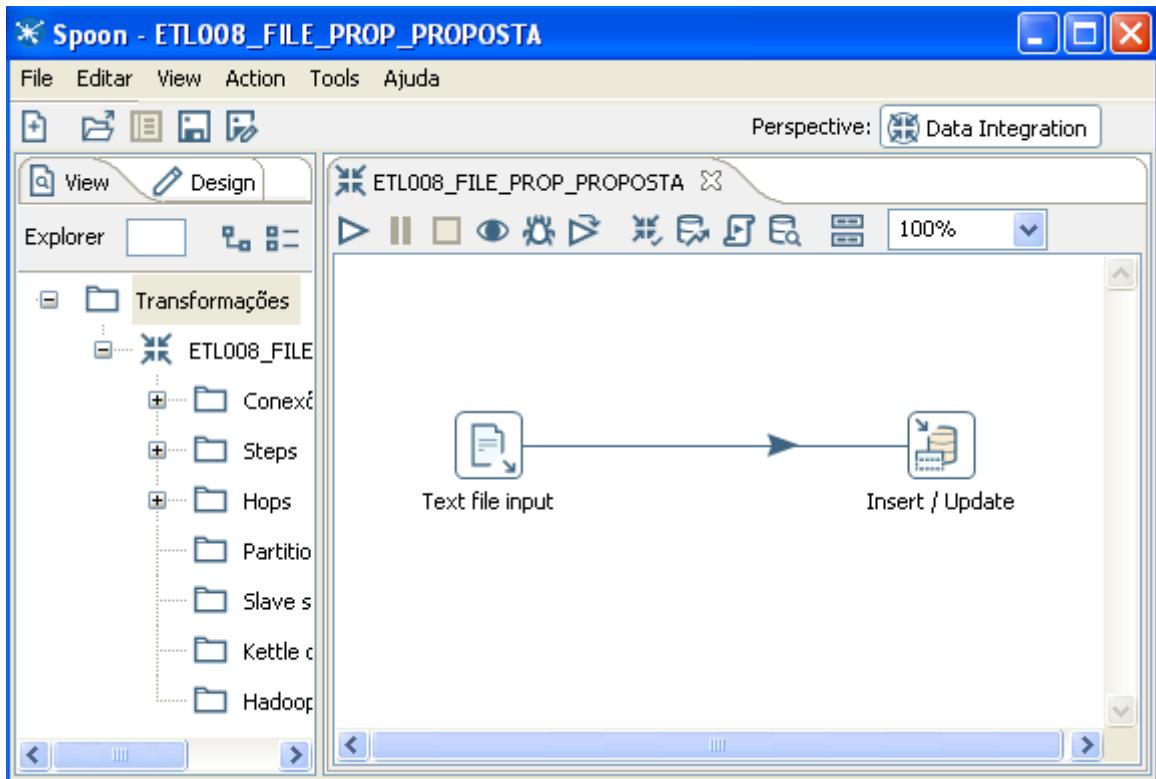


Figura 101. Carga de dados na Tabela Proposta
Fonte: Autor.

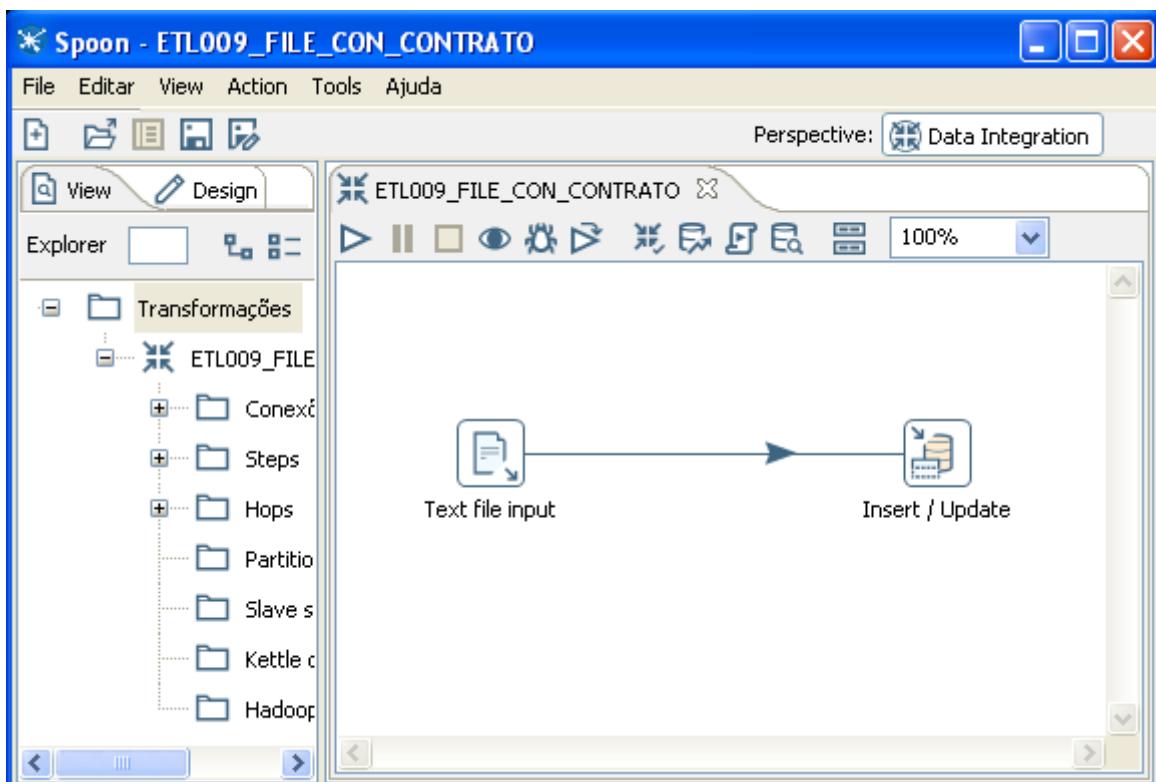


Figura 102. Carga de dados na Tabela Contrato
Fonte: Autor.

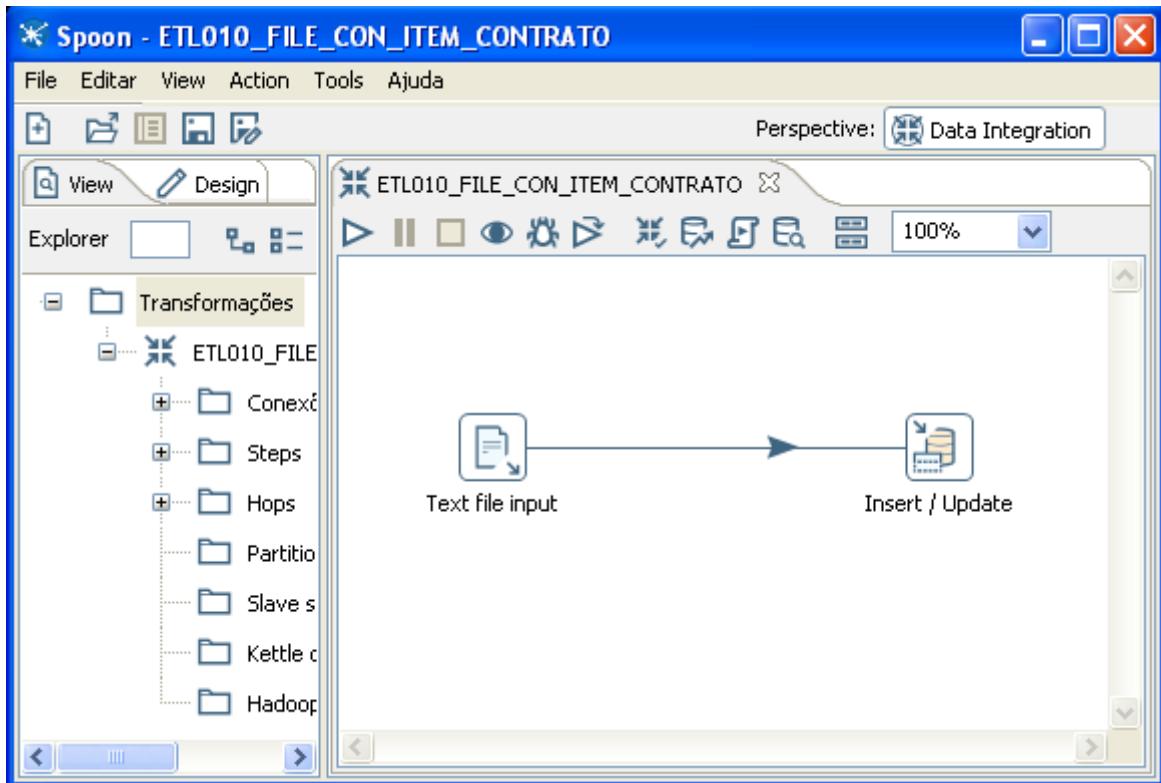


Figura 103. Carga de dados na Tabela Item Contrato

Fonte: Autor.

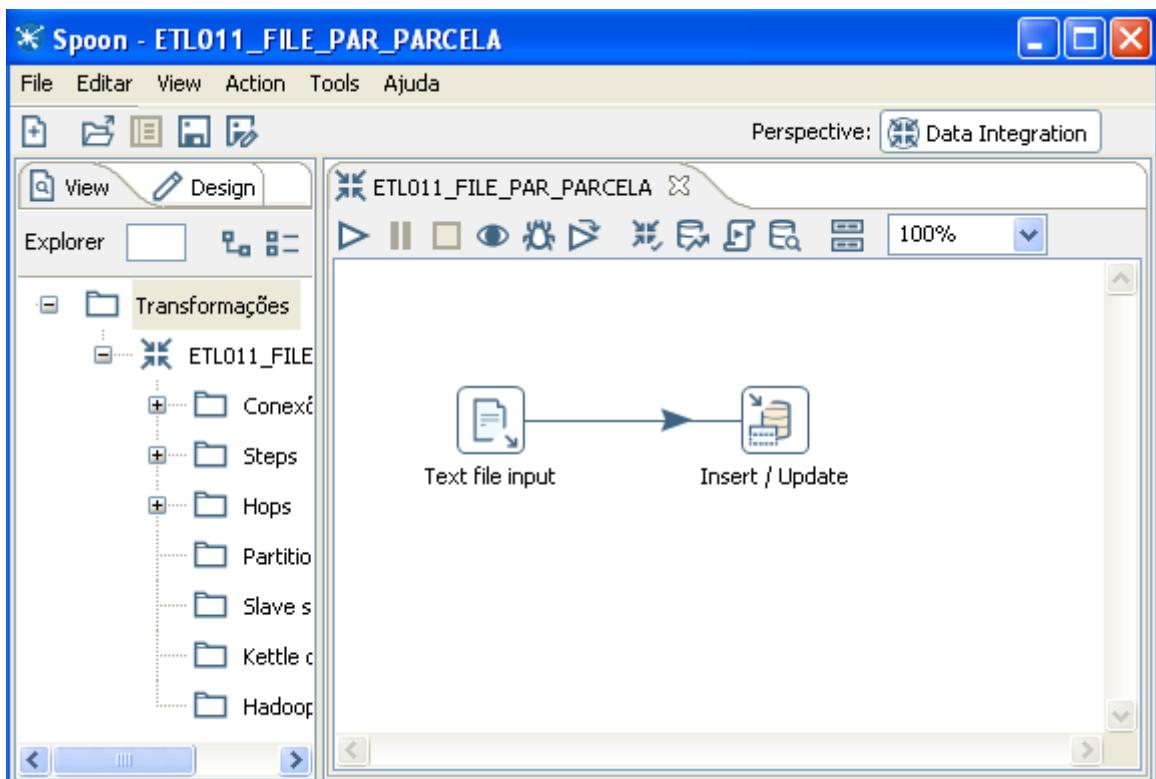


Figura 104. Carga de dados na Tabela Parcela

Fonte: Autor.

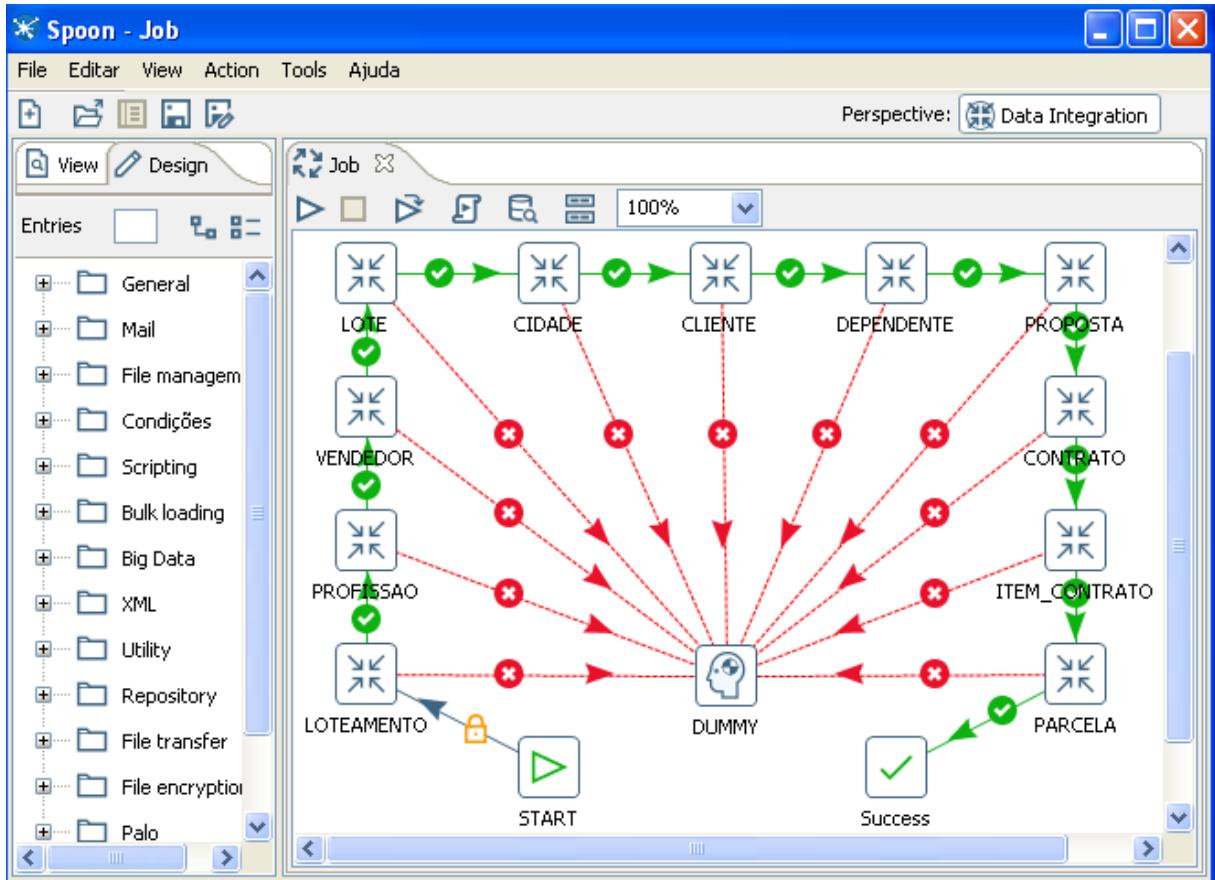


Figura 105. Job para executar todas as transformações de carga
Fonte: Autor.

3.6.6 Conclusão da criação do banco de dados operacional

Com o banco de dados operacional criado, uma gama de possibilidades se abre, a partir desta fase já se pode gerar relatórios para a organização sem impactar o ambiente transacional. Outras fontes de dados podem ser carregadas, ou seja, dados de outros sistemas e aplicações podem ser integrados. Este esforço de integração entre os sistemas transacionais da empresa simplifica o processo de carga de um futuro data warehouse.

Como os bancos de dados relacionais não são projetados para que o usuário simplesmente veja, entenda e analise, remodelaremos as tabelas do ambiente operacional e criaremos um ambiente analítico, nesta etapa, criaremos um data mart para análises gerenciais dos dados operacionais, para isso criaremos um modelo multidimensional.

Mais detalhes sobre data warehouses, data marts e modelagem multidimensional estão descritas na parte teórica deste trabalho.

3.7 Modelagem Multimimensional

O primeiro passo para iniciar a modelagem dimensional é definir a granularidade. Como os dados de menor nível granular são fundamentais para responder perguntas mais específicas, estudamos os dados disponibilizados no ambiente operacional e de acordo com as expectativas do cliente definimos os níveis mais baixos a serem abordados.

3.7.1 Dimensão Tempo (Quando)

A dimensão tempo é fundamental em um banco de dados dimensional. Embora possa conter níveis de detalhes em horas, minutos e até mesmo segundos, geralmente contém dias como nível granular mais baixo. Com a granularidade diária podemos organizar os dados por dias, semanas, quinzenas, meses, bimestres, trimestres, semestres, ano, etc. Diferente das outras dimensões, a dimensão tempo pode ser criada e carregada antecipadamente, uma única vez. É a dimensão que sofre menos impacto com alterações e mudanças do negócio.

Criamos a tabela T001_DIM_TEMPO, conforme o script mostrado na tabela 38, em seguida adicionamos os valores dos campos gerados a partir de uma planilha externa, um exemplo dos dados constantes na dimensão tempo é mostrado na tabela 39.

```

CREATE TABLE T001_DIM_TEMPO
(
    ID_TEMPO NUMBER NOT NULL
    , DATA DATE
    , DIA NUMBER
    , NUM_DIA_SEMANA NUMBER
    , DSC_DIA_SEMANA VARCHAR2(15 BYTE)
    , DSC_DIA_SEMANA_ABREV VARCHAR2(15 BYTE)
    , COD_DIA_UTIL NUMBER
    , DSC_DIA_UTIL VARCHAR2(5 BYTE)
    , DSC_FERIADO VARCHAR2(50 BYTE)
    , COD QUIZENA NUMBER
    , DSC QUIZENA VARCHAR2(20 BYTE)
    , MES NUMBER
    , DSC_MES VARCHAR2(10 BYTE)
    , COD_ANO_MES NUMBER
    , DSC_MES_ANO VARCHAR2(8 BYTE)
    , COD_BIMESTRE NUMBER
    , DSC_BIMESTRE VARCHAR2(15 BYTE)
    , COD_TRIMESTRE NUMBER
    , DSC_TRIMESTRE VARCHAR2(15 BYTE)
    , COD_ESTACAO NUMBER
    , DSC_ESTACAO VARCHAR2(10 BYTE)
    , COD_SEMESTRE NUMBER
    , DSC_SEMESTRE VARCHAR2(15 BYTE)
    , ANO NUMBER
    , CONSTRAINT T001_DIM_TEMPO_PK PRIMARY KEY
        ( ID_TEMPO ) ENABLE );

```

Tabela 38. Script de criação da Dimensão Tempo
Fonte: Autor.

T001_DIM_TEMPO	
ID_TEMPO	20160101
DATA	01/01/2016
DIA	1
NUM_DIA_SEMANA	6
DSC_DIA_SEMANA	SEXTA-FEIRA
DSC_DIA_SEMANA_ABREV	SEX
COD_DIA_UTIL	0
DSC_DIA_UTIL	FE
DSC_FERIADO	Confraternização Universal
COD QUIZENA	1
DSC QUIZENA	1ª Quinzena
MES	1
DSC_MES	JANEIRO
COD_ANO_MES	201601
DSC_MES_ANO	Jan/16
COD_BIMESTRE	1
DSC_BIMESTRE	1º Bimestre
COD_TRIMESTRE	1
DSC_TRIMESTRE	1º Trimestre
COD_ESTACAO	1
DSC_ESTACAO	Verão
COD_SEMESTRE	1
DSC_SEMESTRE	1º Semestre
ANO	2016

Tabela 39. Exemplo de dados constantes na Dimensão Tempo
Fonte: Autor.

Criamos a tabela T001_DIM_TEMPO. É opcional a padronização de nomenclatura, atribuímos “T” ao tipo do objeto, (T=Tabela, V= View e MV= Materialized view), “001” é o número do objeto, “DIM” é a sua função e por último o nome do objeto criado.

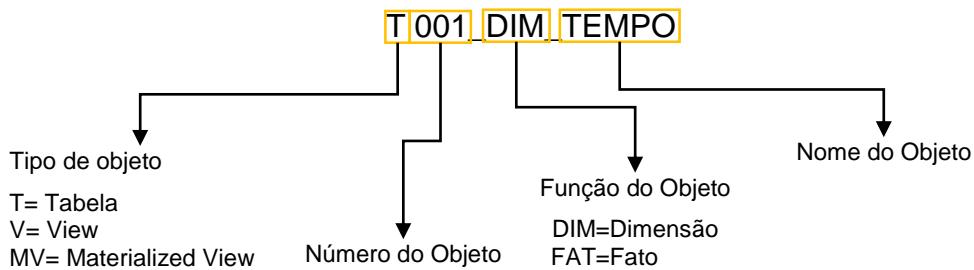


Figura 106. Padronização opcional de nomenclatura

Fonte: Autor.

Conforme mostrado na tabela 39, foi criado na dimensão tempo o atributo Id_tempo, é um atributo numérico, composto pela data em formato yyyyymmdd. A Dimensão permitirá descrições e agregações úteis como nome do dia da semana, descrição da quinzena, agregação por trimestres, semestres, descrição de feriados, entre outros.

3.7.2 Dimensão Produto (O quê)

No modelo de negócios mapeado, o produto vendido em um contrato é o lote, o lote é componente de um loteamento, deste modo, para gerar a dimensão produto utilizaremos uma Materialized view criada com a junção das tabelas Lote e Loteamento.

Visões materializadas são objetos do banco de dados Oracle que contém dados locais armazenados na forma de tabelas. Geralmente são utilizadas para criar agregações de tabelas em determinado intervalo de tempo, que pode ser definido no próprio banco de dados ou executado manualmente (ON DEMAND).

As Materialized views utilizam dados de uma ou mais tabelas originais, solicitadas através de um Select e armazenadas em um único objeto destinatário sob a forma de tabela, ao atualizar a visão materializada, este objeto é atualizado com a última visão disponível dos dados existentes nas tabelas originais.

```

CREATE MATERIALIZED VIEW MV002_DIM_PRODUTO
USING INDEX
REFRESH ON DEMAND COMPLETE
WITH PRIMARY KEY
DISABLE QUERY REWRITE AS
SELECT
    LOTE.COD_LOTE,
    LOTE.COD_LOTEAMENTO,
    LOTE.AREA_LOTE,
    LOTE.PRECO_LOTE,
    LOTE.SIT_LOTE,
    LOTEAMENTO.NOME_LOTEAMENTO,
    LOTEAMENTO.CIDADE_LOTEAMENTO,
    LOTEAMENTO.UF_LOTEAMENTO
FROM
    LOTE
LEFT JOIN LOTEAMENTO ON (LOTE.COD_LOTEAMENTO = LOTEAMENTO.COD_LOTEAMENTO);

COMMENT ON MATERIALIZED VIEW MV002_DIM_PRODUTO IS 'Membro do modelo Dimensional Contrato
ODSLOTE.MV002_DIM_PRODUTO';

```

Tabela 40. Script de Criação da Dimensão Produto

Fonte: Autor.

A Dimensão MV002_DIM_PRODUTO foi criada, o modo de atualização é “On Demand”, isso porque queremos que sua atualização ocorra em um processo separado após a finalização da carga dos dados operacionais. Observe que a chave primária é composta, respeitando a necessidade do negócio, isso gera linhas repetidas na tabela e vai justamente ao contrário do que foi feito na modelagem do banco de dados operacional, na modelagem dimensional a desnormalização é uma característica comum.

	COD_LOTE	COD_LOTEAMENTO	AREA_LOTE	PRECO_LOTE	SIT_LOTE	NOME_LOTEAMENTO
	1	1	100	40000 A		JARDIM DAS CAMELIAS
	2	1	100	40000 A		JARDIM DAS CAMELIAS
	3	1	100	40000 A		JARDIM DAS CAMELIAS
	6	1	100	40000 A		JARDIM DAS CAMELIAS
	7	1	100	40000 A		JARDIM DAS CAMELIAS
	8	1	100	40000 A		JARDIM DAS CAMELIAS
	1	2	100	40000 A		JARDIM DOS IPES
	2	2	400	40000 A		JARDIM DOS IPES

Figura 107. Dimensão Produto desnortmalizada

Fonte: Autor.

3.7.3 Dimensão Vendedor (como)

Um contrato é a efetivação de uma proposta prospectada e efetivada por um vendedor, embora o aceite seja por parte do cliente, é por meio dos esforços do vendedor que o acordo de compra é assinado. Portanto a dimensão Vendedor oferecerá para o analista uma importante visão a respeito do desempenho dos vendedores no resultado das vendas.

A tabela vendedor oferece todos os campos que farão parte da dimensão, no entanto, para evitar problemas na estrutura dimensional, não utilizaremos a tabela original, ao invés disso utilizaremos novamente uma Materialized view, isso porque caso a tabela original receba novos atributos, isso não impactará no modelo dimensional, uma vez que a view materializada continuará inalterada, criamos então a dimensão MV003_DIM_VENDEDOR.

```

CREATE MATERIALIZED VIEW MV003_DIM_VENDEDOR
USING INDEX
REFRESH ON DEMAND COMPLETE
WITH PRIMARY KEY
DISABLE QUERY REWRITE AS
SELECT
  COD_VENDEDOR,
  NOME_VENDEDOR,
  NUM_DDD_VENDEDOR,
  NUM_FONE_VENDEDOR,
  NUM_PERC_COMISSAO
FROM VENDEDOR;

COMMENT ON MATERIALIZED VIEW MV003_DIM_VENDEDOR IS 'Membro do modelo Dimensional Contrato
ODSLOTE.MV003_DIM_VENDEDOR';

```

Tabela 41. Script de Criação da Dimensão Vendedor
Fonte: Autor.

3.7.4 Dimensão Cliente (Quem)

Ao criarmos o modelo dimensional, estamos desnormalizando a informação, obviamente que mantemos um atributo chave para eliminar linhas duplicadas, este atributo chave será o único relacionamento com a tabela fato e é este o motivo da existência do modelo dimensional, embora ele até gere alguma redundância, ele simplifica a abordagem sobre os dados para o analista de negócios.

Na dimensão MV004_DIM_CLIENTE Id_cliente é o atributo chave, adicionamos a profissão, a idade e desnormalizamos a informação de dependente.

```

CREATE MATERIALIZED VIEW MV004_DIM_CLIENTE
USING INDEX
REFRESH ON DEMAND COMPLETE
WITH PRIMARY KEY
DISABLE QUERY REWRITE AS
SELECT
    CLIENTE.ID_CLIENTE,
    NOME_CLIENTE,
    DTA_NASC_CLIENTE,
    TRUNC(MONTHS_BETWEEN(SYSDATE,DTA_NASC_CLIENTE)/12) IDADE,
    SEXO,
    PROFISSAO_CLIENTE,
    NVL(GRAU_INSTR_CLIENTE,'NI') GRAU_INSTR_CLIENTE,
    VLR_RENDER,
    A.DEPENDENTE1, A.GRAU_DEPNCIA1,
    A.DEPENDENTE2, A.GRAU_DEPNCIA2,
    A.DEPENDENTE3, A.GRAU_DEPNCIA3,
    NUM_DDD_CLIENTE,
    NUM_FONE_CLIENTE
FROM CLIENTE
INNER JOIN PROFISSAO ON (CLIENTE.COD_PROFISSAO = PROFISSAO.COD_PROFISSAO)
LEFT JOIN (
    SELECT
        ID_CLIENTE,
        MAX(DECODE(COD_DEPENDENTE,1,NOME_DEPENDENTE))DEPENDENTE1,
        MAX(DECODE(COD_DEPENDENTE,1,GRAU_DEPENDENCIA))GRAU_DEPNCIA1,
        MAX(DECODE(COD_DEPENDENTE,2,NOME_DEPENDENTE))DEPENDENTE2,
        MAX(DECODE(COD_DEPENDENTE,2,GRAU_DEPENDENCIA))GRAU_DEPNCIA2,
        MAX(DECODE(COD_DEPENDENTE,3,NOME_DEPENDENTE))DEPENDENTE3,
        MAX(DECODE(COD_DEPENDENTE,3,GRAU_DEPENDENCIA))GRAU_DEPNCIA3
    FROM DEPENDENTE
    GROUP BY ID_CLIENTE
)A ON (CLIENTE.ID_CLIENTE =A.ID_CLIENTE);

COMMENT ON MATERIALIZED VIEW MV004_DIM_CLIENTE IS 'Membro do modelo Dimensional Contrato
ODSLOTE.MV004_DIM_CLIENTE';

```

Tabela 42. Script de Criação da Dimensão Cliente

Fonte: Autor.

3.7.5 Dimensão Localidade (Onde)

A dimensão MV005_DIM_LOCALIDADE foi criada. As informações de localidade são atributos da tabela Cliente, logo, poderíamos adicionar os campos na própria dimensão Cliente, no entanto optamos por criar uma dimensão separada, isso porque desta forma prevemos a necessidade de efetuar mudanças em atributos da localidade sem alterar a dimensão Cliente, imagine que futuramente o usuário tenha interesse em criar agrupamentos, separações por região comercial ou adição de novos atributos geográficos.

```

CREATE MATERIALIZED VIEW MV005_DIM_LOCALIDADE
USING INDEX
REFRESH ON DEMAND COMPLETE
WITH PRIMARY KEY
DISABLE QUERY REWRITE AS
SELECT
ID_CLIENTE,
REPLACE(LOGR_CLIENTE,'!','") LOGRADOURO,
NUM_LOGR_CLIENTE,
NUM_CEP,
CIDADE_CLIENTE,
UF_CLIENTE
FROM CLIENTE
INNER JOIN CIDADE ON (CIDADE.COD_CIDADE = CLIENTE.COD_CIDADE);

COMMENT ON MATERIALIZED VIEW MV005_DIM_LOCALIDADE IS 'Membro do modelo Dimensional Contrato
ODSLOTE.MV005_DIM_LOCALIDADE';

```

Tabela 43. Script de Criação da Dimensão Localidade

Fonte: Autor.

3.7.6 Fato Indicadores (Métricas)

A tabela Fato é a tabela que contém métricas a serem agregadas ou contadas. Existem diferentes tipos de medidas que podem ser armazenadas na tabela Fato, as medidas aditivas são as mais comuns, são aquelas que podem ser somadas ou contadas em qualquer uma das visões e o resultado será novas medidas, por exemplo, no modelo em questão, o valor total do contrato, saldo devedor, quantidade de parcelas, quantidade de propostas, etc.

As medidas semi-aditivas são aquelas que podem ser somadas apenas por algumas visões específicas, por exemplo, o tamanho da área de um lote.

Os fatos não-aditivos são aqueles que não podem ser somados em nenhuma das visões, por exemplo, campos que compõe percentuais fixos de índices, porcentagens, proporções, etc. Para estes campos, o numerador e o denominador devem ser armazenados na tabela fato.

Ao modelar a tabela fato, os atributos chave das dimensões devem estar em conformidade com o nível atômico desta tabela, definir um nível de detalhe que garanta a maior quantidade de respostas possíveis sem impactar na performance de processamento é o recurso chave que garante o sucesso do modelo. Não é um processo baseado em um relatório específico, e nem deve ser, por este motivo, várias perguntas devem ser feitas ao modelo para compor a tabela de métricas,

desde a visão macro até a visão micro, obviamente que no âmbito das expectativas de respostas esperadas pelo gestor ou analista.

Pergunta	O modelo responde?
Quantos clientes há na base de dados?	Sim
Quantas propostas foram prospectadas por cliente?	Sim
Sabemos a data da última proposta prospectada?	Sim
Sabemos os valores da última proposta negociada?	Sim
Sabemos quais propostas foram efetivadas?	Sim
Sabemos onde mais vendemos?	Sim
Se a proposta gerou contrato, sabemos quais são os valores do contrato?	Sim
Conseguimos saber quais são os produtos do contrato?	Sim
Conseguimos informações sobre as parcelas do contrato?	Sim
É possível saber se o contrato possui parcelas em atraso?	Sim
Conseguimos a data da última parcela paga?	Sim
Pode informar o saldo devedor por contrato?	Sim

Tabela 44. Questões preliminares que o modelo deve responder
Fonte: Autor.

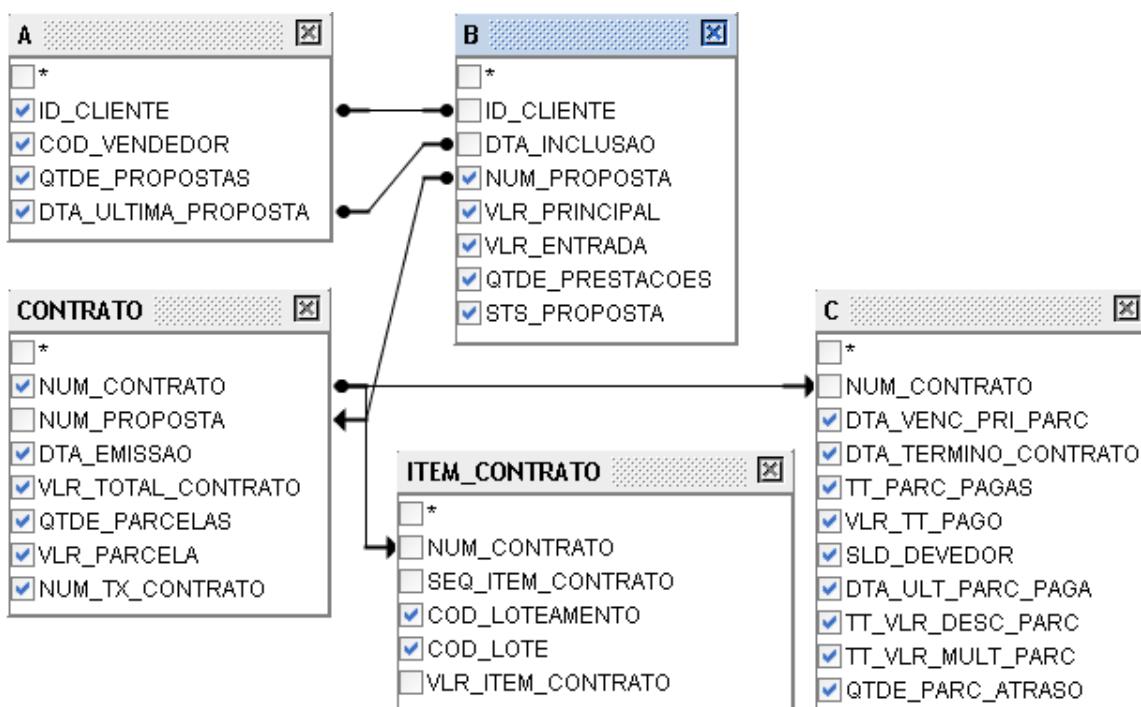


Figura 108. Esquema relacional da tabela Fato
Fonte: Autor.

No processo de criação do modelo multidimensional primeiro definimos a granularidade, em seguida definimos as dimensões e por último criamos a Fato. A

tabela 44 mostra o script de criação da MV006_FAT_INDICADORES. Se possível todas as chaves devem ser numéricas.

```

CREATE MATERIALIZED VIEW MV006_FAT_INDICADORES
USING INDEX
REFRESH ON DEMAND COMPLETE
WITH PRIMARY KEY
DISABLE QUERY REWRITE AS
SELECT
    A.ID_CLIENTE, A.COD_VENDEDOR,
    ITEM_CONTRATO.COD_LOTEAMENTO, ITEM_CONTRATO.COD_LOTE,
    TO_NUMBER(TO_CHAR(A.DTA_ULTIMA_PROPOSTA,'YYYYMMDD'))ID_TEMPO,
    A.QTDE_PROPOSTAS, A.DTA_ULTIMA_PROPOSTA,
    B.NUM_PROPOSTA, B.VLR_PRINCIPAL, B.VLR_ENTRADA, B.QTDE_PRESTACOES, B.STS_PROPOSTA,
    CONTRATO.NUM_CONTRATO, CONTRATO.DTA_EMISSAO,
    CONTRATO.VLR_TOTAL_CONTRATO, CONTRATO.QTDE_PARCELAS,
    CONTRATO.VLR_PARCELA, CONTRATO.NUM_TX_CONTRATO,
    C.DTA_VENC_PRI_PARC, C.DTA_TERMINO_CONTRATO,
    C.TT_PARC_PAGAS, C.VLR_TT_PAGO,
    C.SLD_DEVEDOR, C.QTDE_PARC_ATRASO,
    C.DTA_ULT_PARC_PAGA, C.TT_VLR_DESC_PARC,
    C.TT_VLR_MULT_PARC
FROM
(
    SELECT
        CLIENTE.ID_CLIENTE,
        PROPOSTA.COD_VENDEDOR,
        COUNT(*)QTDE_PROPOSTAS,
        MAX(DTA_INCLUSAO)DTA_ULTIMA_PROPOSTA
    FROM CLIENTE
    INNER JOIN PROPOSTA ON (CLIENTE.ID_CLIENTE = PROPOSTA.ID_CLIENTE)
    GROUP BY CLIENTE.ID_CLIENTE, PROPOSTA.COD_VENDEDOR
)A
INNER JOIN
(
    SELECT
        ID_CLIENTE, DTA_INCLUSAO, NUM_PROPOSTA, VLR_PRINCIPAL,
        VLR_ENTRADA, QTDE_PRESTACOES, STS_PROPOSTA
    FROM PROPOSTA
)B ON (B.ID_CLIENTE = A.ID_CLIENTE AND B.DTA_INCLUSAO = A.DTA_ULTIMA_PROPOSTA)
LEFT JOIN CONTRATO ON (B.NUM_PROPOSTA = CONTRATO.NUM_PROPOSTA)
LEFT JOIN ITEM_CONTRATO ON (CONTRATO.NUM_CONTRATO = ITEM_CONTRATO.NUM_CONTRATO)
LEFT JOIN (
    SELECT
        NUM_CONTRATO,
        MIN(DTA_VENCIMENTO) DTA_VENC_PRI_PARC,
        MAX(DTA_VENCIMENTO) DTA_TERMINO_CONTRATO,
        COUNT(DTA_PAGAMENTO) TT_PARC_PAGAS,
        SUM(NVL(VLR_TOTAL_PAGO,0))VLR_TT_PAGO,
        SUM(VLR_PARCELA)-SUM(NVL(VLR_TOTAL_PAGO,0)) SLD_DEVEDOR,
        MAX(DTA_PAGAMENTO) DTA_ULT_PARC_PAGA,
        SUM(VLR_DESCONTO_PARCELA) TT_VLR_DESC_PARC,
        SUM(VLR_MULTA_PARCELA) TT_VLR_MULT_PARC,
        SUM(CASE
            WHEN (TRUNC(DTA_VENCIMENTO) < TRUNC(SYSDATE) AND DTA_PAGAMENTO IS NULL)
            THEN 1 ELSE 0
        END) QTDE_PARC_ATRASO
    FROM PARCELA
    GROUP BY NUM_CONTRATO
)C ON (CONTRATO.NUM_CONTRATO = C.NUM_CONTRATO);

COMMENT ON MATERIALIZED VIEW MV006_FAT_INDICADORES IS 'Membro do modelo Dimensional Contrato
ODSLOTE.MV006_FAT_INDICADORES';

```

Tabela 45. Script de Criação da tabela Fato
Fonte: Autor.

Observe na figura 109 a cardinalidade do modelo e na figura 110 a estrutura lógica do esquema estrela ODSLOTE.

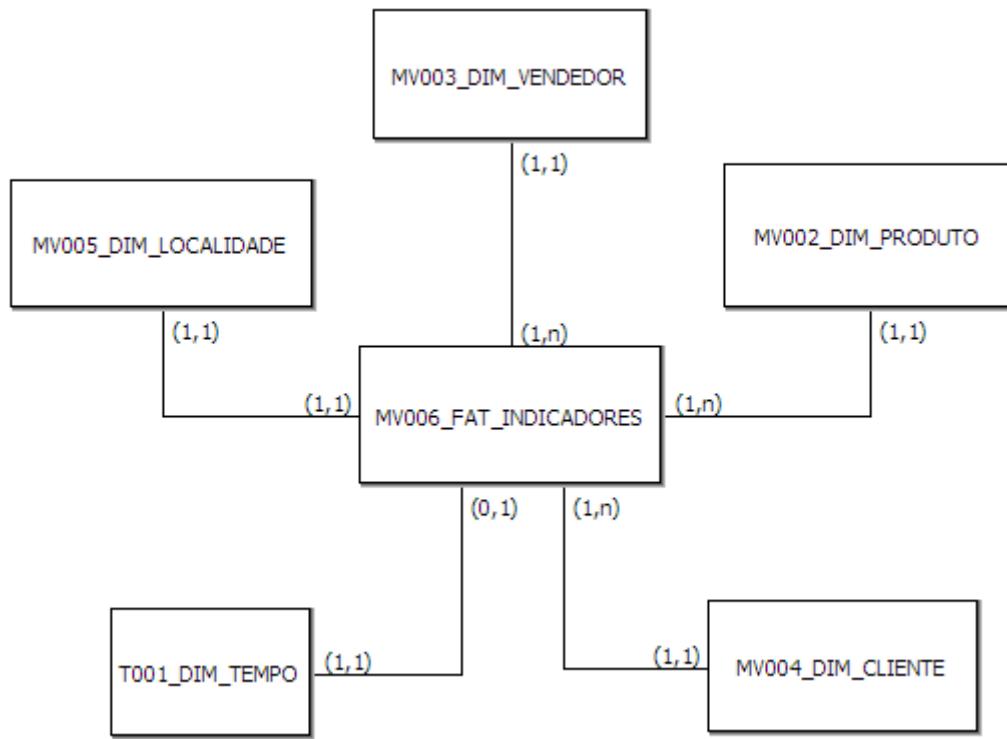


Figura 109. Cardinalidade do modelo dimensional
Fonte: Autor.

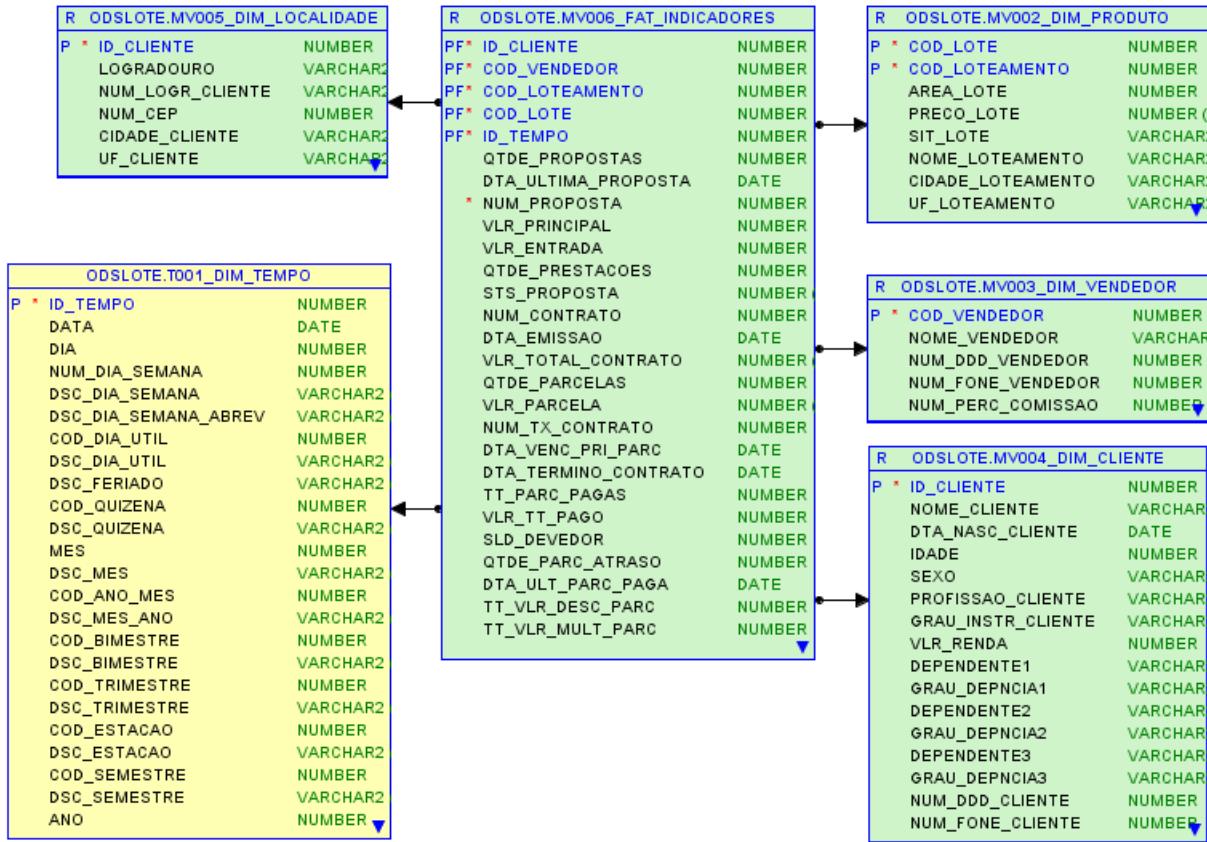


Figura 110. Star Schema ODSLote

Fonte: Autor.

Temos o nosso banco de dados multidimensional criado. Após o processo de carga do banco de dados operacional (normalizado), executaremos uma procedure que atualizará o modelo multidimensional (desnormalizado). Obviamente que toda uma estrutura de rastreio e validação de erros deverá ser criada, com parâmetros de processamento, quantidade de registros processados, datas, envio de e-mails, etc.

A sintaxe para executar a atualização das dimensões e da Fato é a descrita na tabela 46.

```

BEGIN
  dbms_mview.refresh ('MV002_DIM_PRODUTO');
  dbms_mview.refresh ('MV003_DIM_VENDEDOR');
  dbms_mview.refresh ('MV004_DIM_CLIENTE');
  dbms_mview.refresh ('MV005_DIM_LOCALIDADE');
  dbms_mview.refresh ('MV006_FAT_INDICADORES');
END;
  
```

Tabela 46. Script para atualizar modelo Dimensional ODSLote

Fonte: Autor.

3.8 Utilizando o modelo multidimensional

Vamos mostrar em uma breve demonstração, as possibilidades de trabalho com uma ferramenta analítica e um modelo multidimensional. O Intuito aqui não é evidenciar a ferramenta, mas sim as facilidades de uso dos recursos que se tornam possíveis ao utilizar um banco de dados dimensional.

3.8.1 O poder da do modelo multidimensional

As imagens a seguir mostram a quantidade de propostas efetivadas emitidas por cidade, agrupadas por período da data de emissão. Nestes exemplos, inicialmente veremos os valores agrupados por ano e cidade, (dimensões de Localidade, Tempo e métricas da tabela Fato), em seguida faremos uso de recursos como drill-down, roll-up e fatiamentos na dimensão tempo.

A descrição estará na legenda das figuras e todas as figuras a seguir são de autoria do autor deste trabalho.

Propostas Efetivadas por Cidade

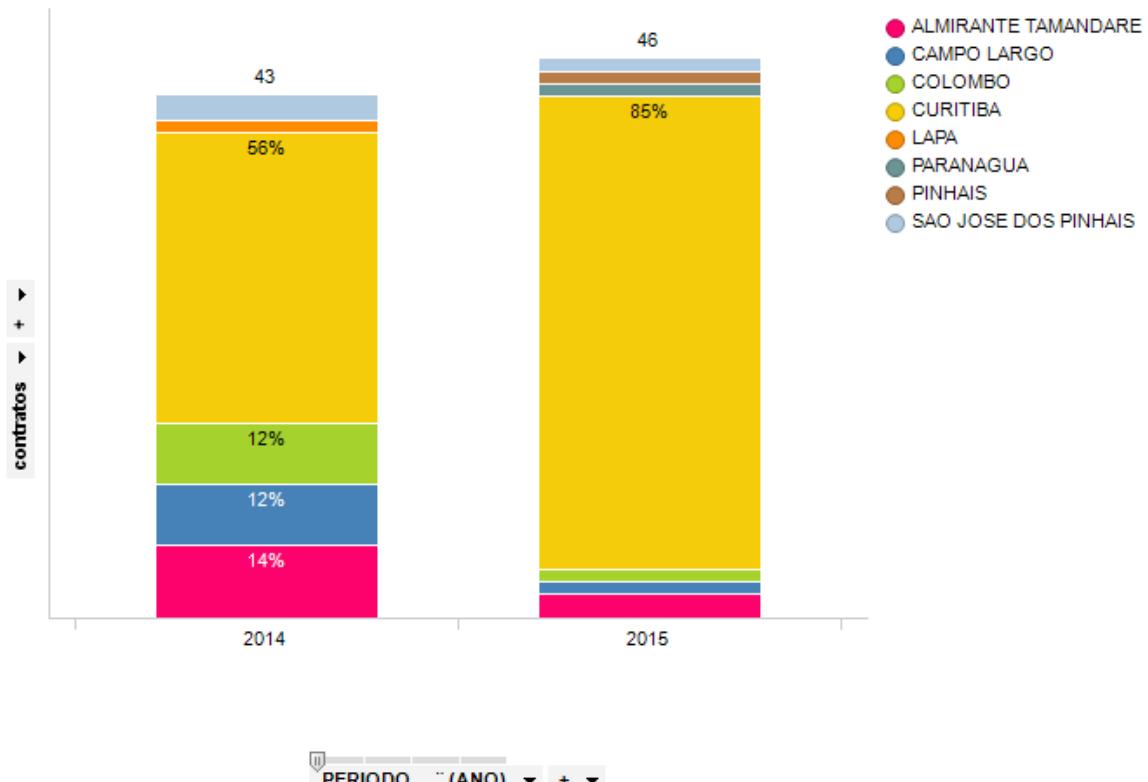


Figura 111. Indicadores no maior nível granular da dimensão Tempo.

Propostas Efetivadas por Cidade

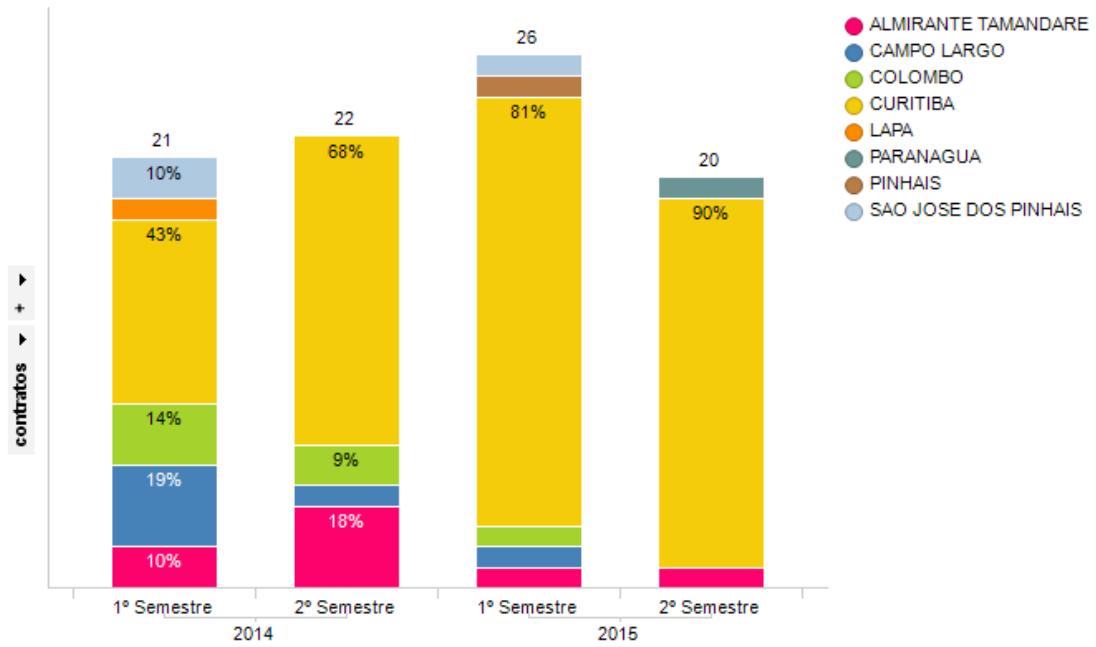


Figura 112. Aplicação de Drill-down I

Propostas Efetivadas por Cidade

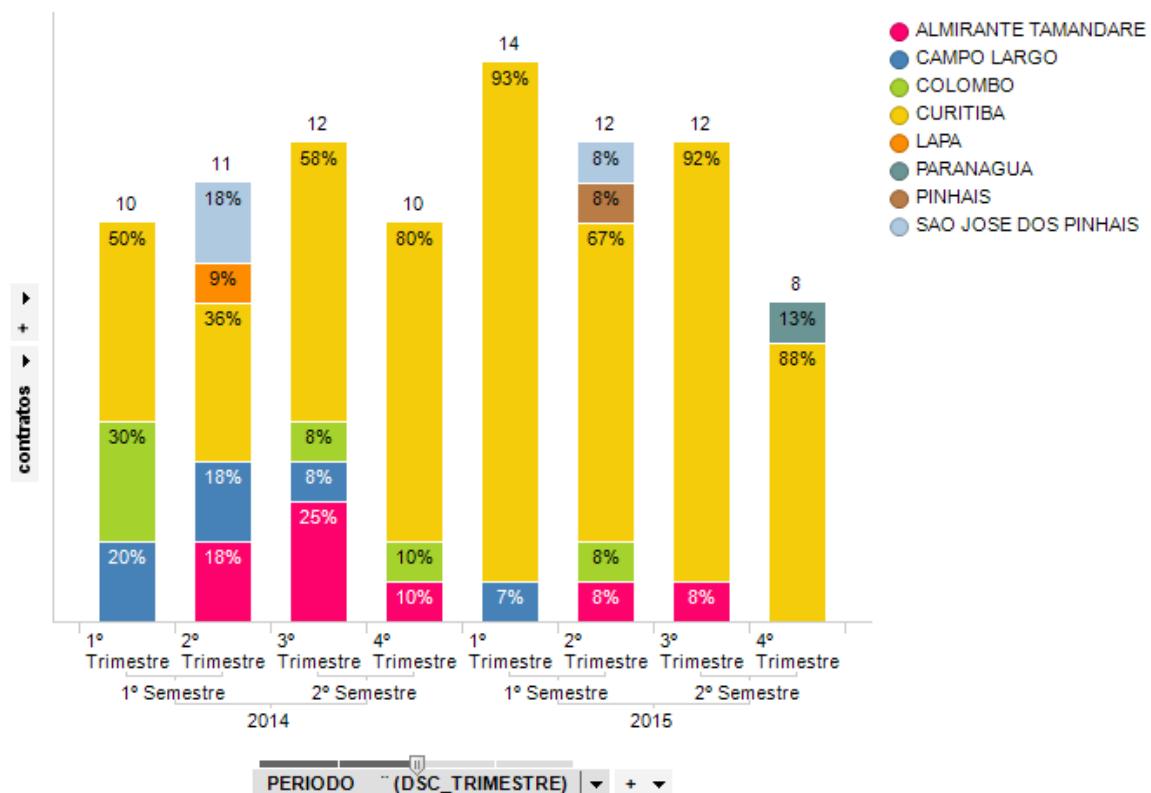


Figura 113. Aplicação de Drill-down II

Propostas Efetivadas por Cidade

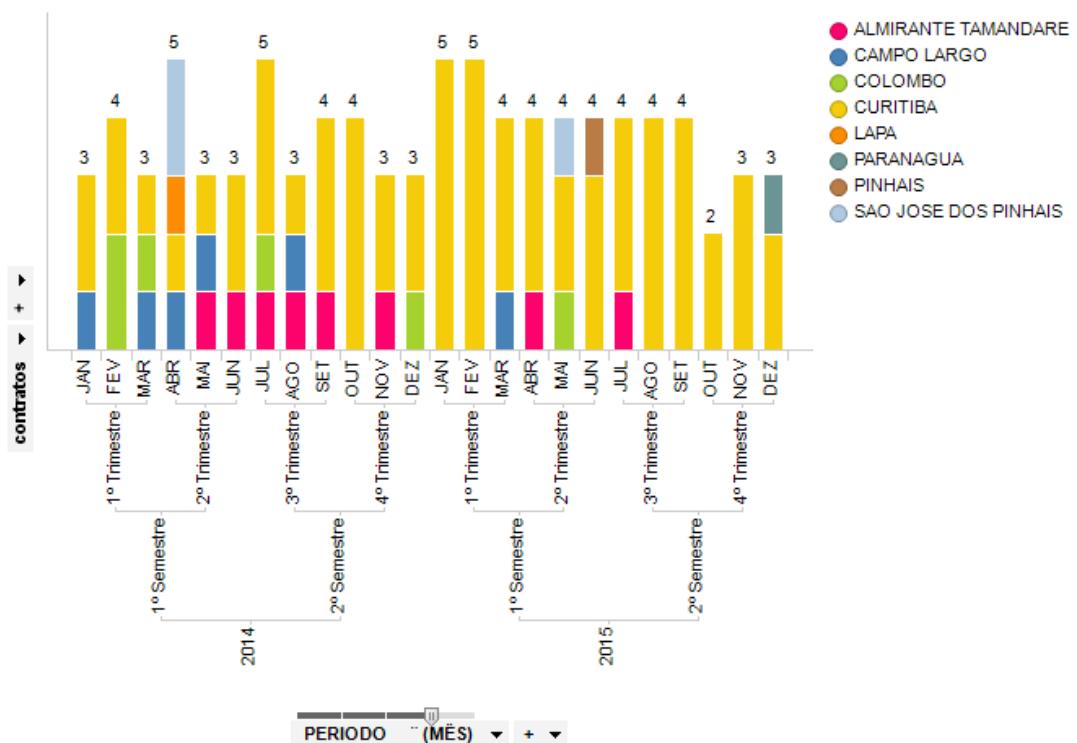


Figura 114. Aplicação de Drill-down III

Fonte: Autor.

Propostas Efetivadas por Cidade

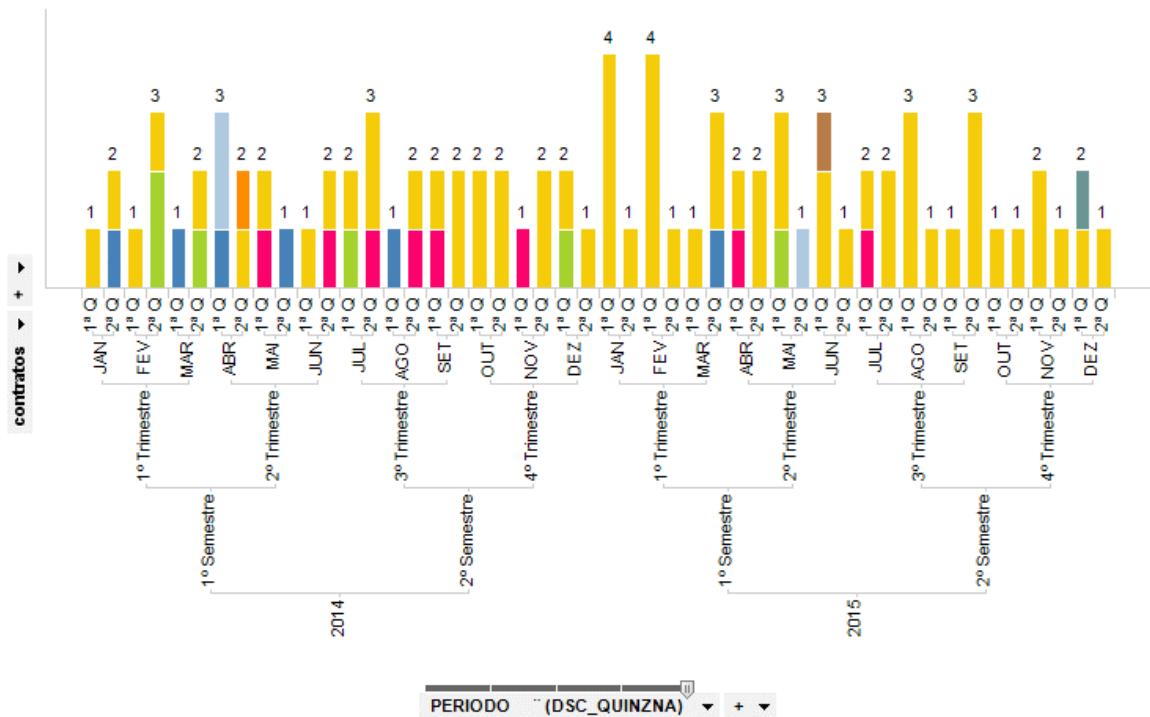


Figura 115. Aplicação de Drill-down IV

Fonte: Autor.

Propostas Efetivadas por Cidade

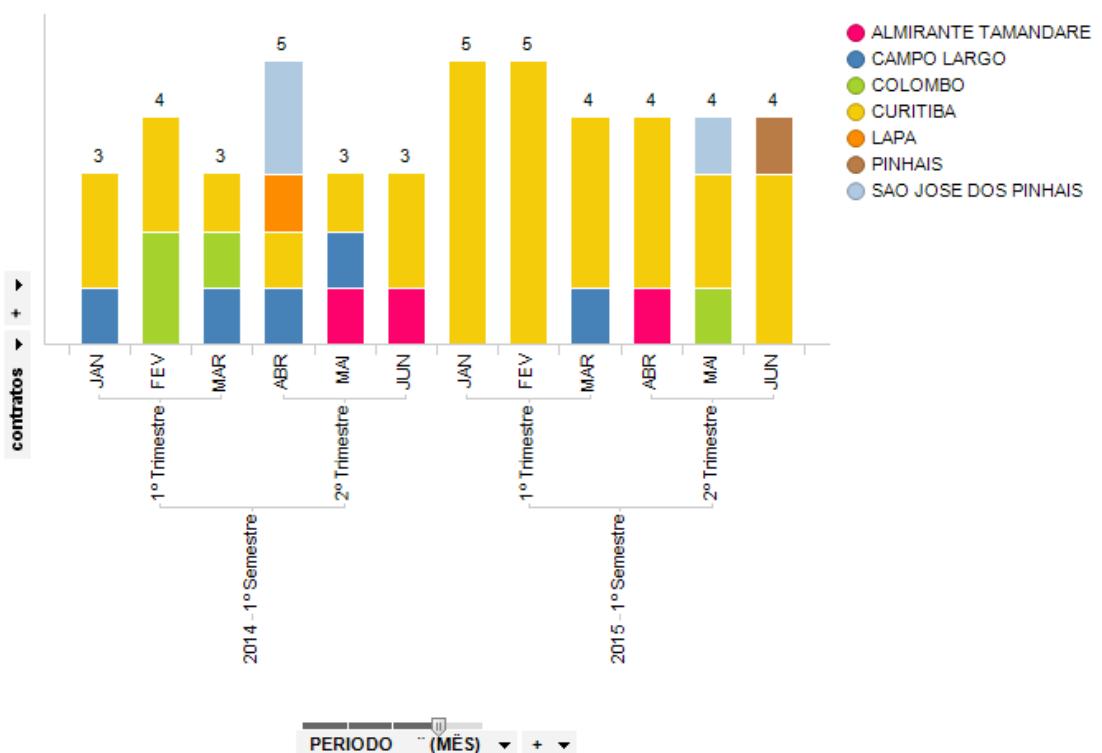


Figura 116. Comparando o resultado mensal do 1º semestre de cada ano

Fonte: Autor.

Propostas Efetivadas por Cidade

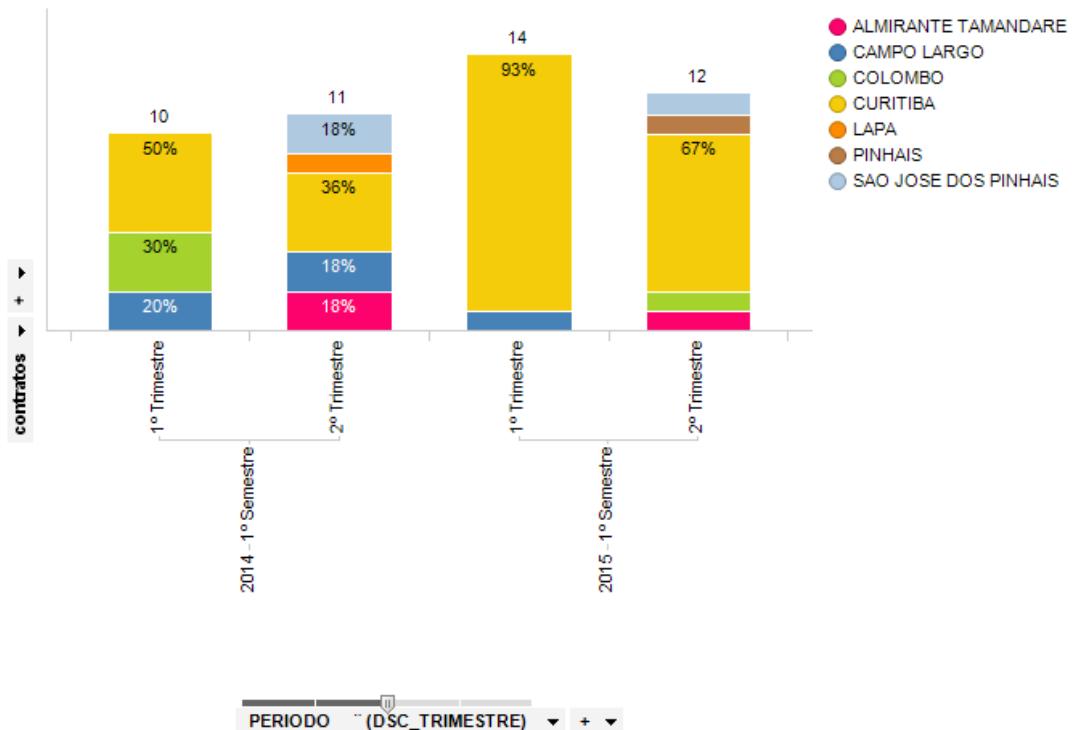


Figura 117. Aplicando Roll-up no comparativo semestral de cada ano
Fonte: Autor.

Propostas Efetivadas por Cidade

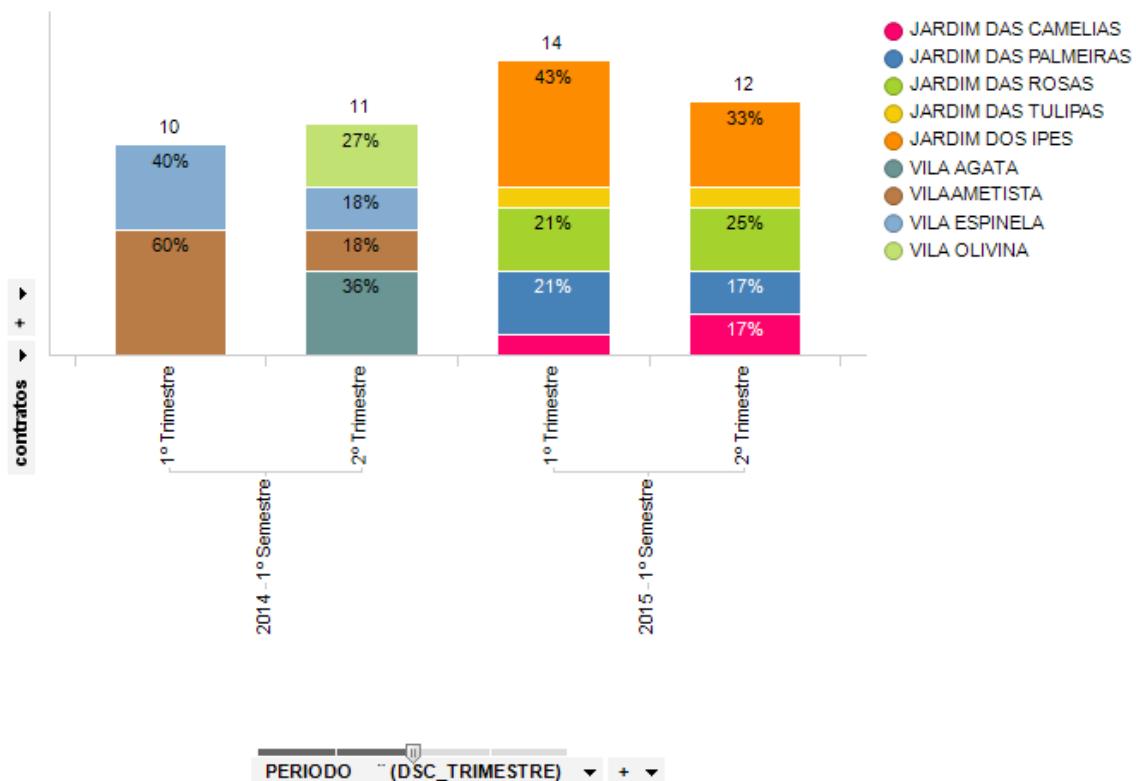


Figura 118. Trocando de Dimensão localidade pela Dimensão Produto
Fonte: Autor.

Período

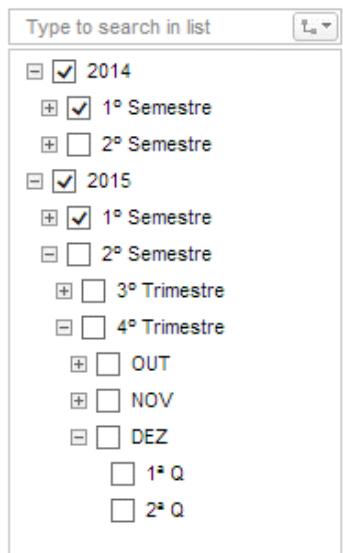


Figura 119. Hierarquia configurada da dimensão tempo
Fonte: Autor.

As figuras seguirão informações da quantidade de propostas trabalhadas por período, a separação de cores indica a quantidade de propostas que ficaram pendentes e a quantidade de propostas que foram efetivadas e geraram contrato, é um gráfico que indica o aproveitamento em relação à quantidade de propostas negociadas.

Por exemplo, a figura 111 informa que 43 propostas foram efetivadas no ano de 2014 e que em 2015 foram 46, este gráfico aponta uma singela melhora nos resultados, mas ainda não é suficiente para tirar alguma conclusão, como é um modelo dimensional, criamos um novo gráfico, onde além das propostas efetivadas, inserimos o a quantidade total de propostas negociadas, ou seja, a quantidade de propostas negociadas nos anos de 2014 e 2015 pode indicar algo sobre o número de contratos gerados?

Observe a figura 120, note que em 2015 o número de propostas negociadas, (indicada pela área verde), quase triplicou em relação ao ano anterior, no entanto estas propostas não foram convertidas em contrato (indicado pela azul), a pergunta que fica é, porquê? Houve mais trabalho em 2015, no entanto, o aproveitamento foi menor.

A dimensão tempo possui o campo numérico Cod_trimestre, no relatório, ao invés de usar a descrição “1º Trimestre”, utilizamos este atributo numérico para compor a descrição dos trimestres como “Q1”, “Q2”, “Q3” e “Q4”.

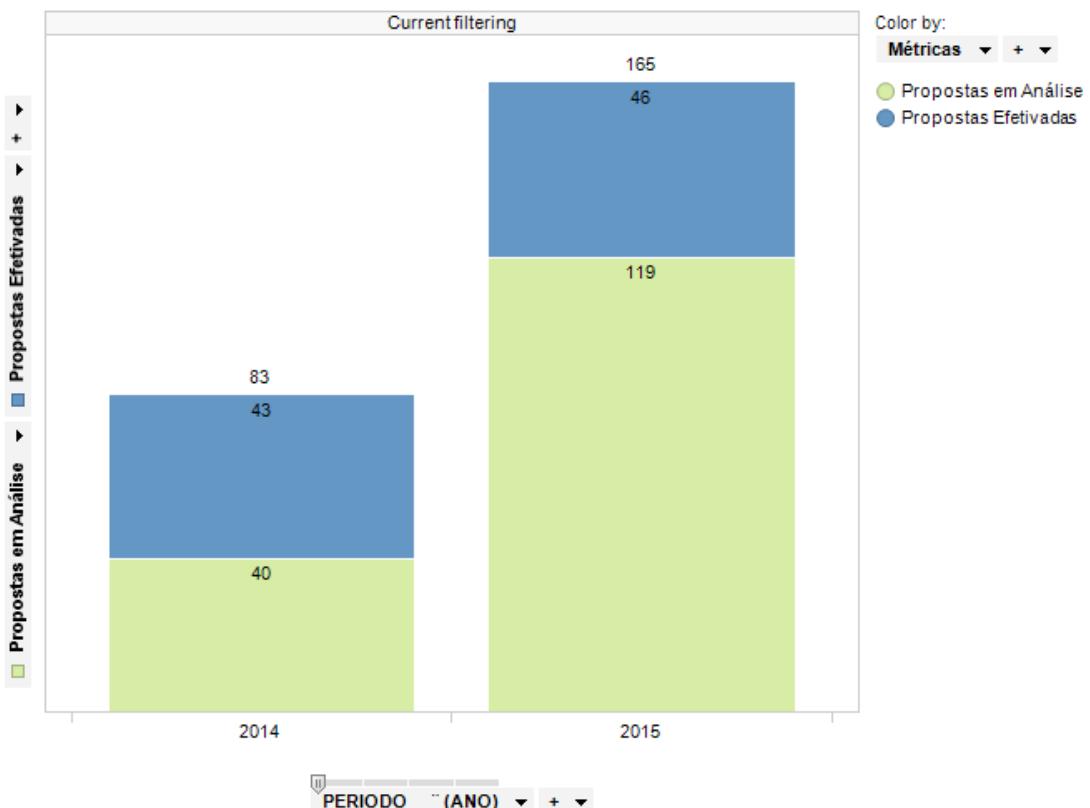


Figura 120. Aproveitamento de Contratos em relação a propostas negociadas.

Propostas x Contrato

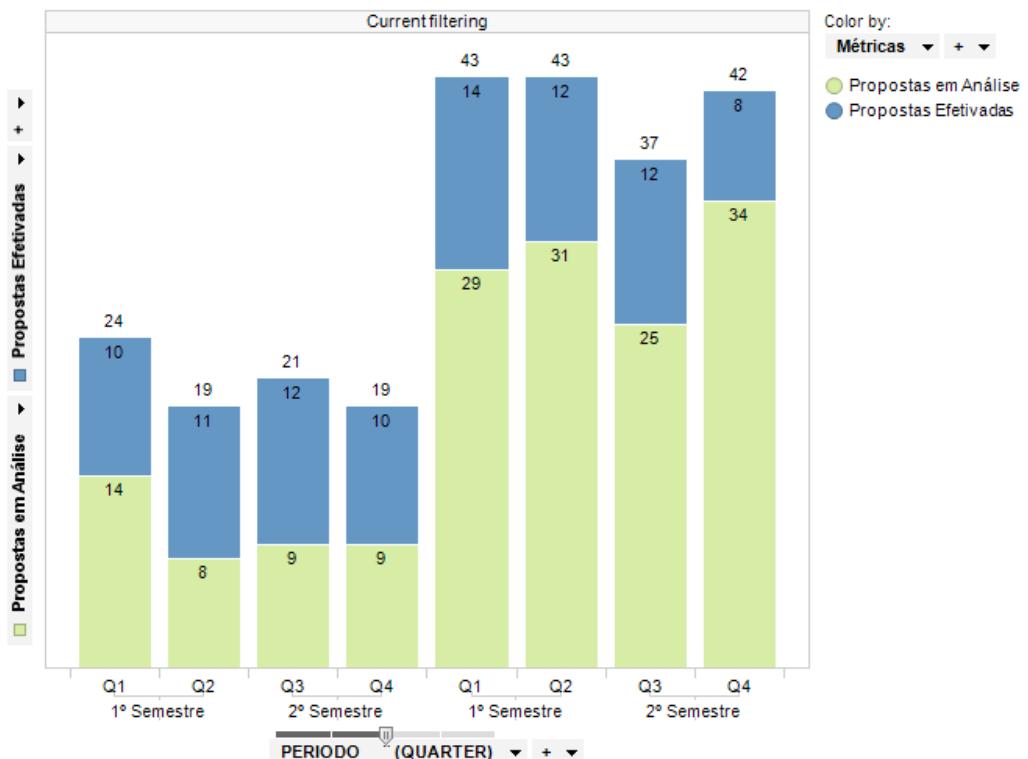


Figura 121. Drill-down da dimensão tempo para detalhar a análise

A figura 121 confirma, houve no ano inteiro de 2015 muito trabalho, no entanto muita negociação e pouca eficiência, afinal de contas propostas não geram receita. Apenas por curiosidade, teria como saber qual foi a participação de cada vendedor nestes números?

Como é um modelo dimensional e esta dimensão existe no modelo, sim, é possível, e bem simples, basta adicionar na análise o atributo Nome_vendedor da dimensão MV003_DIM_VENDEDOR.

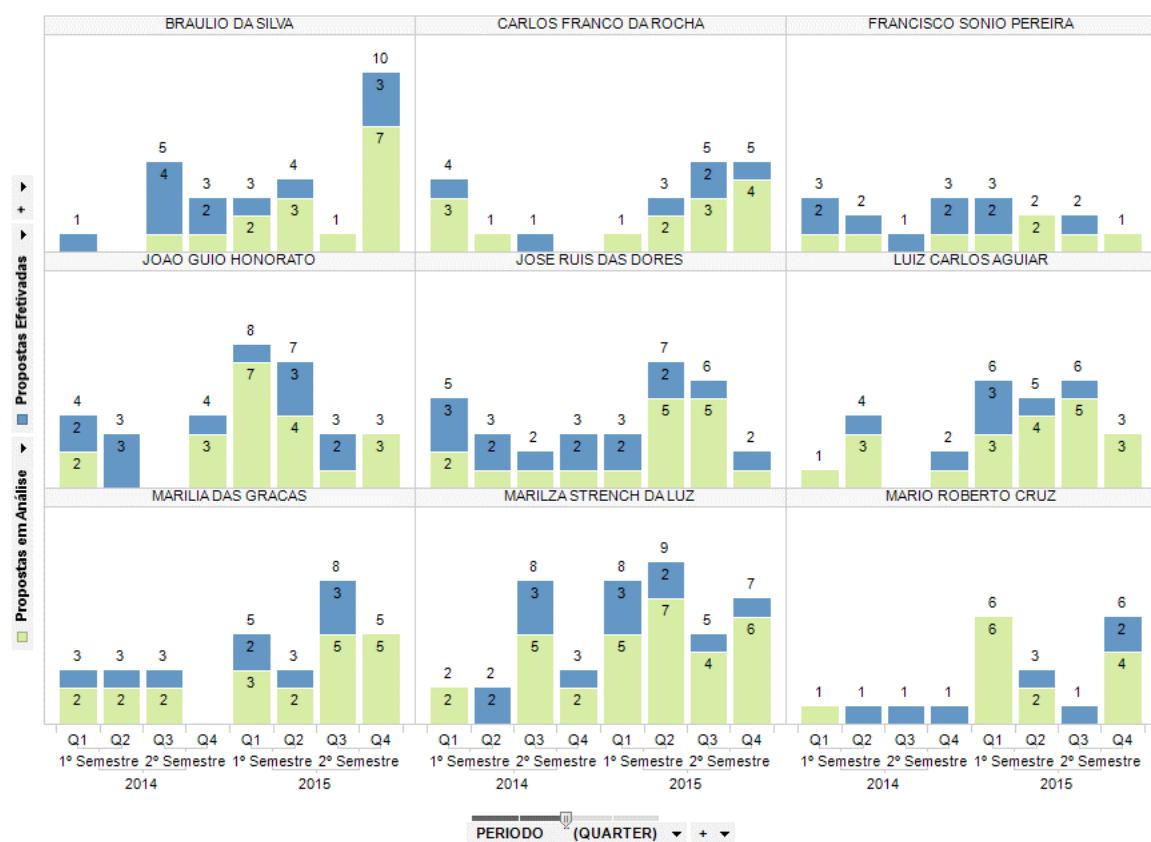


Figura 122. Análise com a adição do campo Nome_vendedor
Fonte: Autor.

Observe a figura 122, caso não tenha notado o nome do respectivo vendedor está logo acima dos gráficos, agora, o que há de errado com o Mario Roberto? A Marilza Strench está a todo vapor, mas espere, muito detalhe, é um modelo dimensional, vamos aplicar um roll-up sobre a hierarquia do período, queremos agora menos detalhe por período, conforme mostrado na figura 123.

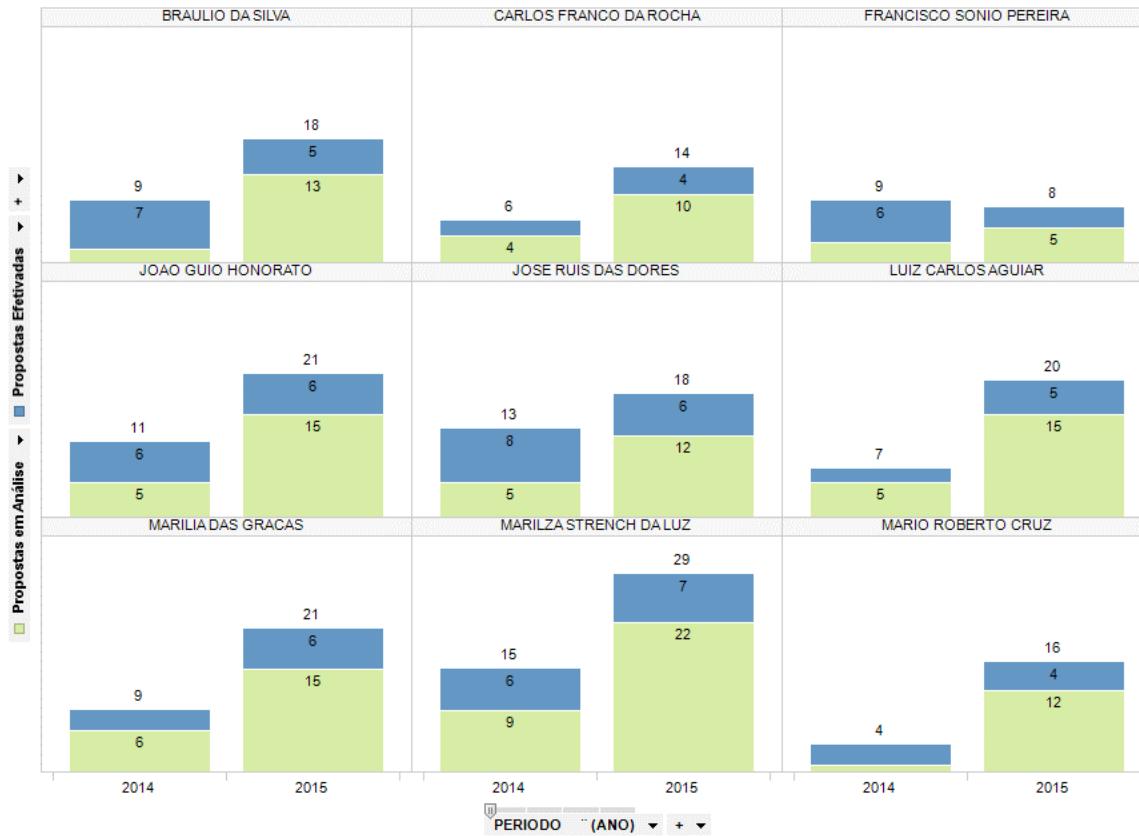


Figura 123. Análise após aplicação de roll-up em período
Fonte: Autor.

A figura 123 agora mostra com clareza o desempenho dos vendedores, inclusive quais são os vendedores com pior desempenho anual, mas ainda vemos apenas muitas propostas em análise e poucas propostas efetivadas, ou seja, continuamos a observar a baixa conversão em contratos.

Observe que inúmeras análises podem ser realizadas apenas incluindo ou alterando visões do modelo, métricas de fato, campos das dimensões, drill-downs, roll-ups, filtros em dimensões, etc.

Continuando a análise, a agora retirando a visão por vendedor, ou apenas deixando-a de lado, até porque ela não ajudou a responder nossa dúvida, porque tantas propostas e poucos contratos.

Podemos isolar o número de propostas efetivadas e utilizar a dimensão produto. É um modelo dimensional e é interessante sabermos quais produtos contribuíram para este resultado, por isso excluímos a métrica das propostas em análise, substituímos o campo da dimensão Vendedor pelo nome do Loteamento e ajustamos as cores conforme mostrado na figura 124.

I Propostas x Contrato

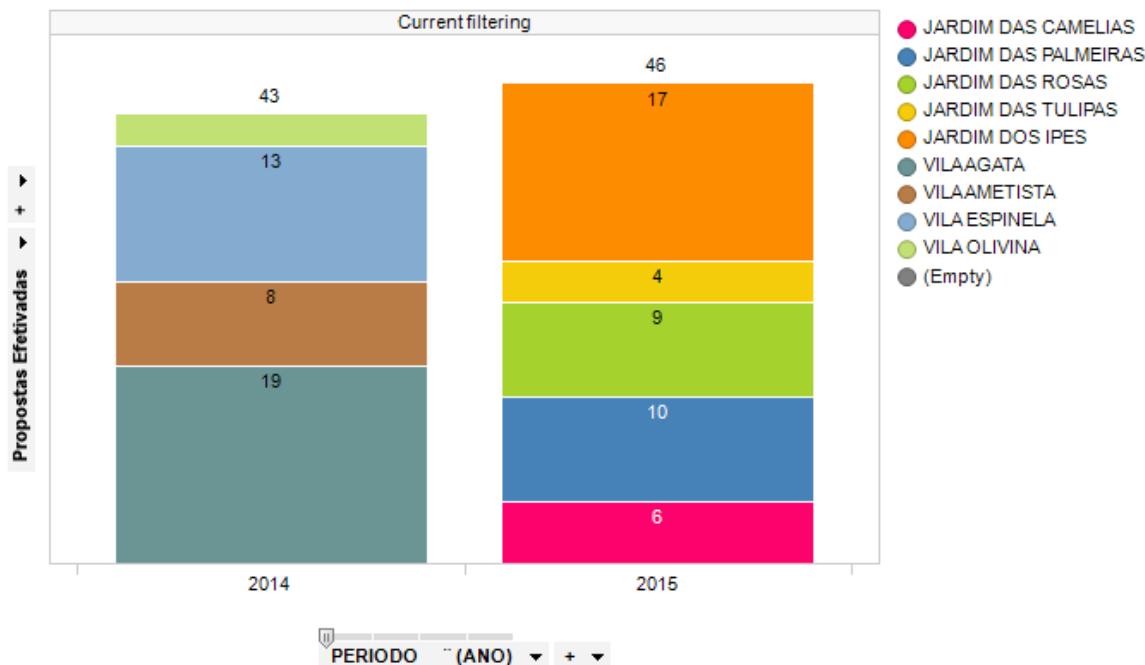


Figura 124. Alterando parâmetros da análise

Fonte: Autor.

I Propostas x Contrato

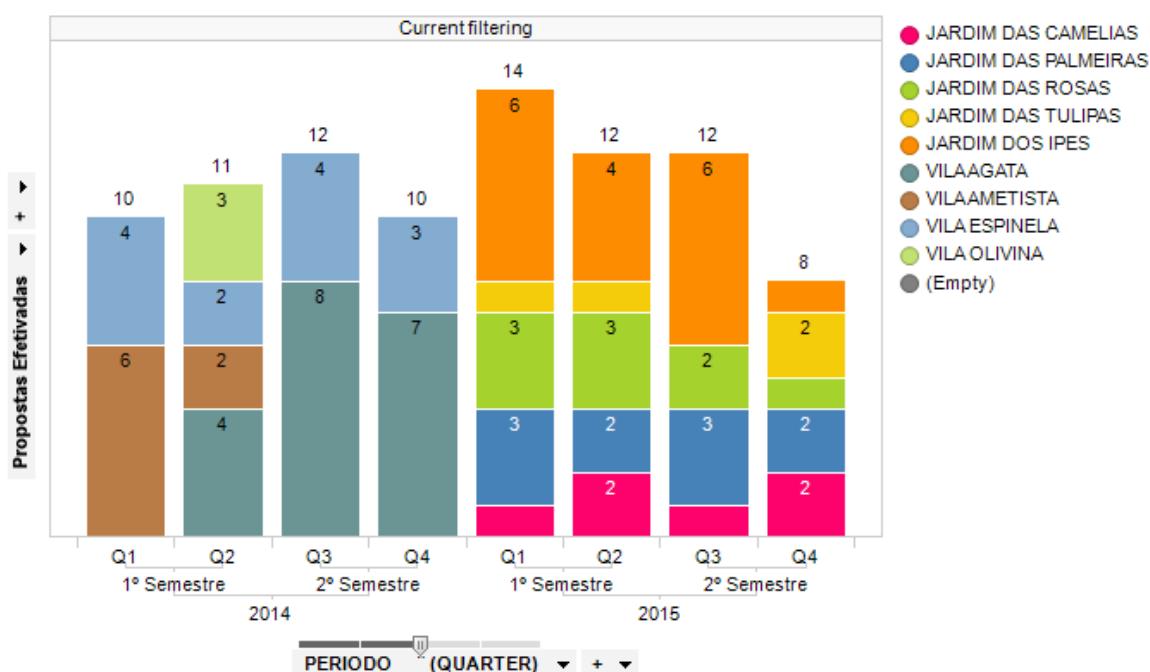


Figura 125. Aplicando drill-down em período

Fonte: Autor.

A figura 125 mostra que os loteamentos de 2014 não contribuíram para os resultados de 2015, por exemplo, o loteamento Vila ametista teve lotes vendidos apenas no primeiro semestre de 2014 e depois não mais, os loteamentos Vila Ágata, Vila Olivina e Vila Espinela também não tiveram contratos assinados a partir de 2014, isso indica que os contratos de 2015 sobretudo foram para novos loteamentos.

A partir deste ponto sabemos quais são os loteamentos que tiveram mais lotes vendidos em 2015, inclusive que foi no Jardim dos Ipês o maior volume de vendas. Esta análise nos leva a um questionamento, como está a situação destes loteamentos? Quantos lotes disponíveis há em cada um? O número de propostas pode ser justificado por uma demanda reprimida por falta de lotes no mesmo padrão dos já vendidos?

Vamos à dimensão produto, lá existe o campo SIT_LOTE, que indica a situação do lote, se já tem contrato assinado ou se está livre. Apenas contaremos a quantidade de lotes por este atributo.

I Situação Lote

● A
● L

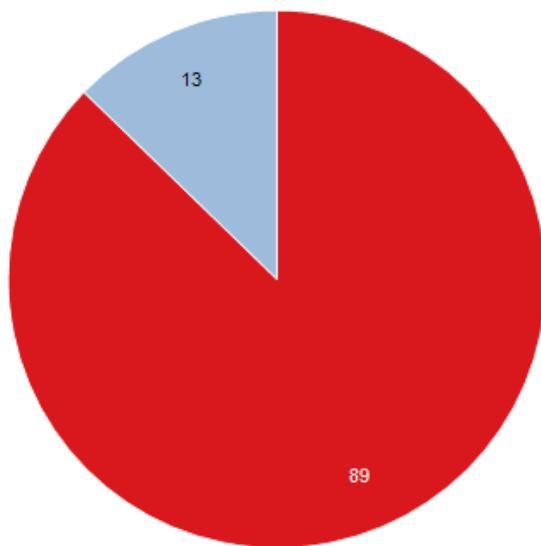


Figura 126. Análise sobre dimensão Produto
Fonte: Autor.

A figura 126 não mostra boa coisa, no modelo fictício ela indica que há poucos lotes disponíveis. Vamos aprofundar a análise apenas separando esses

números pelo seu respectivo loteamento, isso porque está com um nível granular alto. A figura 127 mostra a alteração com o nome do loteamento inserido na análise.

I Lotes Disponíveis



Figura 127. Adicionando detalhe sobre análise da dimensão Produto
Fonte: Autor.

A figura 127 é a última imagem sobre a exemplificação das possibilidades de análise sobre um modelo dimensional, observe que a quantidade de lotes disponíveis ajuda a justificar os motivos pelos quais há tantas propostas não efetivadas para o ano de 2015. Isso porque se um cliente desejar adquirir um lote nos padrões de preço e financiamento de um lote do Jardim Ipês por exemplo, ele

não assinará contrato até que um lote nestas características seja ofertado, e por não haver disponibilidade de produtos a ofertar, os números de venda não aumentam.

Vale lembrar que este é um modelo fictício e inúmeras outras variáveis podem ser consideradas, o cenário descrito é apenas para demonstrar como é uma breve utilização de um data mart.

A figura 128 mostra o que pode ser uma entre inúmeras visões pré configuradas para estudos e análises gerenciais, enfim a implementação de um sistema de informações.

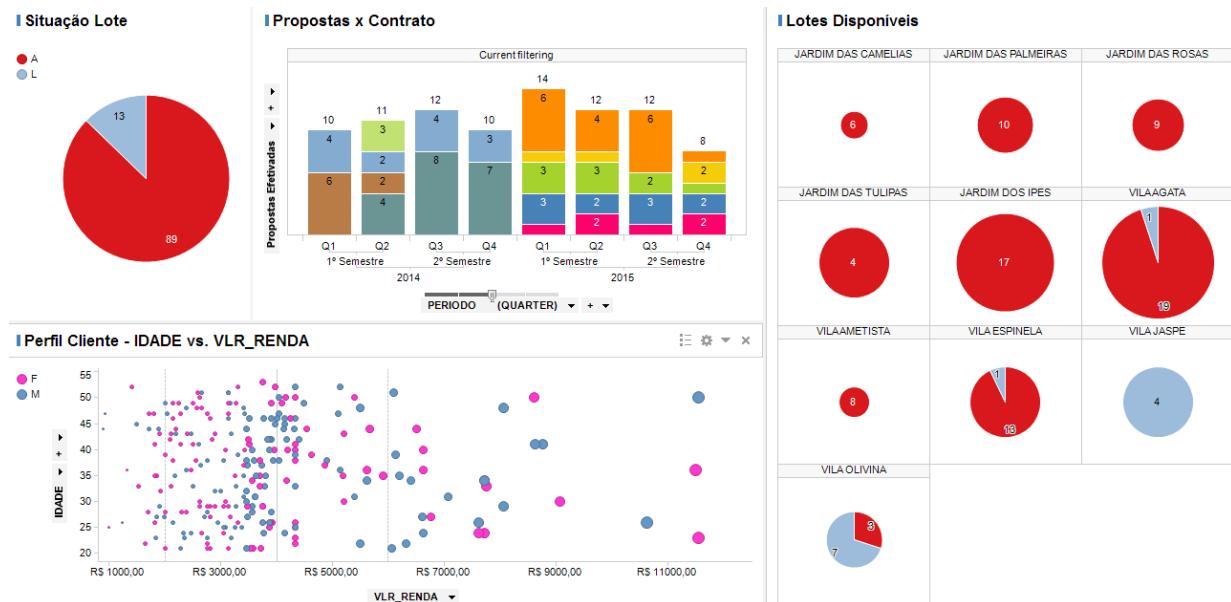


Figura 128. Painel com Informações sobre os dados.

Fonte: Autor.

4 CONCLUSÃO

Um modelo de dados é uma coleção de informações sobre os dados, que possui estreita relação com o negócio e serve como fonte de conhecimento sobre uma ou mais origem de dados. A modelagem lógica fornecerá a base sobre a qual se poderá construir um entendimento sobre os processos do negócio e alinhar a compreensão das perspectivas envolvidas em uma solução de Business Intelligence. Neste sentido, o presente trabalho apresentou a modelagem de dados e expôs os principais recursos para oferecer ao leitor o entendimento de quais conceitos são fundamentais para que haja uma perspectiva de implementação adequada às necessidades do negócio.

Por meio deste trabalho foi possível demonstrar que independente de qual tecnologia possa ser utilizada, o fato mais importante de um ambiente de business analytics, é o dado em si, para tal, apresentaram-se os entendimentos que abrangem como os dados devem ser estruturados e organizados para possibilitar a geração de relatórios e análises.

O estudo mostrou como o relacionamento e integração em um banco de dados pode transformar dados em informação e como uma aplicação de BI pode fornecer essa informação aos atores (gestores e dirigentes) do negócio, que tomarão por sua vez decisões com base nela. Esclareceu qual a finalidade de modelos dimensionais e as vantagens trazidas pelas aplicações OLAP, suas principais características, funcionais e analíticas.

Quanto aos objetivos, este trabalho mostrou os principais conceitos sobre modelagem de dados, desde a definição dos requisitos do negócio até a construção e implementação de uma aplicação de BI, para isso, um estudo de caso foi utilizado, onde reunidas as necessidades do negócio, preparou-se um modelo conceitual, projetou-se o modelo lógico, preparou-se a definição de integridade dos dados, projetou-se o modelo físico e implementação do banco de dados, foram descritos os processos de extração, transformação e carregamento (ETL), projetou-se o modelo multidimensional e foram criados relatórios, análises e painéis para uso dos usuários.

O Trabalho abordou a importância de toda a parte de infraestrutura não técnica, referindo metodologias, procedimentos e processos do negócio. Demonstrou a importância de um banco de dados normalizado na 3FN e evidenciou

como um modelo relacional bem construído, que gera resultados corretos e tem bom desempenho, é um sólido alicerce para o desenvolvimento de um modelo dimensional bem-sucedido.

Os exemplos utilizados neste trabalho, que representam características de pessoas ou operações do negócio estudado, são todos fictícios, qualquer semelhança com nomes, endereços, números de contato ou valores de transações reais é mera coincidência.

Enfatizando a modelagem de dados, este trabalho proporcionou mostrar as etapas para construção de um banco de dados operacional, umas das etapas na construção de um data warehouse, obviamente muito ainda existe a considerar para de fato implementar uma solução de BI utilizando um data warehouse como fonte de dados, no entanto, o trabalho apresenta como contribuições fornecidas pelo estudo, os motivos pelos quais o entendimento e documentação do dado não deve ser visto como um processo opcional.

Conforme descrito no trabalho, optamos em utilizar uma aplicação OLAP remodelando nosso banco de dados relacional no esquema em estrela. Para trabalhos futuros é possível aprofundar esta pesquisa realizando a modelagem em snowflake e comparando o desempenho e usabilidade entre os modelos. Não foram mencionadas as tecnologias multidimensionais ROLAP, MOLAP, HOLAP, neste caso, também fica em aberto para futuras pesquisas.

A definição de data warehouse descrita neste trabalho considera a arquitetura top-down, defendida por Bill Inmon, já o estudo de caso remete à criação de um data warehouse em arquitetura bottom-up, defendida por Ralph Kimball, que também afirma que um data warehouse deve ser modelado utilizando um modelo dimensional, em star schema. Inmon, é contrário a Kimball. Inmon afirma que um data warehouse não deve ser modelado em star schema, mas sim, usando um modelo relacional normalizado na 3FN. Deste modo, para trabalhos futuros é possível estudar a definição e diferenças entre as duas arquiteturas e métodos de modelagem para data warehouses, não mencionadas aqui por não ser o foco do trabalho.

REFERENCIAS

ABREU, M. P.; MACHADO, F. N. **Projeto de Banco de Dados - Uma Visão Prática.** 11^a. ed. São Paulo: Érica, 2004.

BECKER, F.; SCHULER, J. P. sourceforge.net. **DbDesignerFork**, 2013. Disponível em: <<http://sourceforge.net/projects/dbdesigner-fork/>>. Acesso em: Dezembro 2015.

CANDIDO, C. H. www.sis4.com. **brModelo 2.0**, 2007. Disponível em: <<http://www.sis4.com/brmodelo/Default.aspx>>. Acesso em: Dezembro 2015.

CHURCHER, C. **Beginning Database Design: From Novice to Professional.** 2nd. ed. New York: Apress, 2007.

DBDESIGNER4. fabforce.net. **DBDesigner4_manual_1.0.42.pdf**. Disponível em: <<http://www.fabforce.net/downloadfile.php?idownloadfile=5>>. Acesso em: Dezembro 2015.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de dados**. 6^a. ed. São Paulo: Pearson Education - Br, 2011.

HEUSER, C. A. **Projeto de Banco de Dados**. 4^a. ed. ,: Bookman, v. 4, 2008.

HOBERMAN, S. **Data Modeling for MongoDB**. 1st. ed. Basking Ridge: Technics Publications, 2014.

INMON, W. H. **Building the Data Warehouse**. 3rd. ed. New York: John Wiley & Sons, 2002.

INMON, W. H. **Building the Data Warehouse**. 4th. ed. New York: John Wiley & Sons, 2005.

INMON, W. H.; STRAUSS, D.; NEUSHLOSS, G. **DW 2.0: The Architecture for the Next Generation of Data Warehousing**. 1st. ed. ,: Morgan Kaufmann, 2008.

KIMBALL, R.; ROSS, M. **The Data Warehouse Toolkit**. 2nd. ed. New York: John Wiley & Sons, 2002.

POWELL, G. **Beginning Database Design**. Indianapolis: Wiley, 2006.

REDMOND, E.; WILSON, J. R. **Seven Databases in Seven Weeks**. [S.l.]: Pragmatic Bookshelf, 2012.

REEVES, L. L. **A Manager's Guide to Data Warehousing**. Indianapolis: Wiley, 2009.

ROLDÁN, M. C. [wiki.pentaho.com/](http://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration). **Pentaho Data Integration**, 2008. Disponível em: <[http://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+\(Kettle\)+Tutorial](http://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+(Kettle)+Tutorial)>. Acesso em: Dezembro 2015.

SAUTER, V. L. **Decision Support Systems for Business Intelligence**. 2nd. ed. St. Louis: John Wiley & Sons, 2010.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados**. Tradução da 5ª Edição. ed. Rio de Janeiro: Elsevier, 2006.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Database System Concepts**. 6th. ed. New York: McGraw-Hill Education, 2010.

STAIR, R. M.; REYNOLDS, G. W. **Principles of Information Systems: A Managerial Approach**. 9th. ed. Boston: Course Technology, 2009.

WITHEE, K. **Microsoft Business Intelligence For Dummies**. 1st. ed. Hoboken: Wiley, 2010.

APENDICE

Script gerado pelo DBDesigner Fork para criação dos objetos no banco de dados relacional.

```
CREATE TABLE LOTEAMENTO (
    COD_LOTEAMENTO NUMERIC NOT NULL,
    NOME_LOTEAMENTO VARCHAR(100),
    CIDADE_LOTEAMENTO VARCHAR(30),
    UF_LOTEAMENTO VARCHAR(2),
    PRIMARY KEY(COD_LOTEAMENTO));
```

```
CREATE TABLE PROFISSAO (
    COD_PROFISSAO INTEGER NOT NULL,
    PROFISSAO_CLIENTE VARCHAR(20),
    PRIMARY KEY(COD_PROFISSAO));
```

```
CREATE TABLE VENDEDOR (
    COD_VENDEDOR NUMERIC NOT NULL,
    NOME_VENDEDOR VARCHAR(100),
    NUM_DDD_VENDEDOR NUMERIC,
    NUM_FONE_VENDEDOR NUMERIC,
    NUM_PERC_COMISSAO NUMERIC(6,2),
    PRIMARY KEY(COD_VENDEDOR));
```

```
CREATE TABLE LOTE (
    COD_LOTE NUMERIC NOT NULL,
    COD_LOTEAMENTO NUMERIC NOT NULL,
    AREA_LOTE NUMERIC,
    PRECO_LOTE NUMERIC(14,2),
    SIT_LOTE VARCHAR(1),
    PRIMARY KEY(COD_LOTE, COD_LOTEAMENTO),
    FOREIGN KEY(COD_LOTEAMENTO)
        REFERENCES LOTEAMENTO(COD_LOTEAMENTO));
```

```
CREATE TABLE CIDADE (
    COD_CIDADE NUMERIC NOT NULL,
    CIDADE_CLIENTE VARCHAR(50),
    PRIMARY KEY(COD_CIDADE));
```

```

CREATE TABLE CLIENTE (
    ID_CLIENTE NUMERIC NOT NULL,
    COD_CIDADE NUMERIC NOT NULL,
    COD_PROFESSAO INTEGER NOT NULL,
    NOME_CLIENTE VARCHAR(255),
    DTA_NASC_CLIENTE DATE,
    GRAU_INSTR_CLIENTE VARCHAR(20),
    VLR_RENDA NUMERIC,
    LOGR_CLIENTE VARCHAR(255),
    NUM_LOGR_CLIENTE NUMERIC,
    COMPL_LOGR_CLIENTE VARCHAR(20),
    NUM_CEP NUMERIC,
    UF_CLIENTE VARCHAR(2),
    NUM_DDD_CLIENTE NUMERIC,
    NUM_FONE_CLIENTE NUMERIC,
    SEXO VARCHAR(2),
    PRIMARY KEY(ID_CLIENTE),
    FOREIGN KEY(COD_PROFESSAO)
        REFERENCES PROFESSAO(COD_PROFESSAO),
    FOREIGN KEY(COD_CIDADE)
        REFERENCES CIDADE(COD_CIDADE));

```

```

CREATE TABLE DEPENDENTE (
    COD_DEPENDENTE NUMERIC NOT NULL,
    ID_CLIENTE NUMERIC NOT NULL,
    NOME_DEPENDENTE VARCHAR(50),
    GRAU_DEPENDENCIA VARCHAR(20),
    PRIMARY KEY(COD_DEPENDENTE, ID_CLIENTE),
    FOREIGN KEY(ID_CLIENTE)
        REFERENCES CLIENTE(ID_CLIENTE));

```

```

CREATE TABLE PROPOSTA (
    NUM_PROPOSTA INTEGER NOT NULL,
    COD_VENDEDOR NUMERIC NOT NULL,
    ID_CLIENTE NUMERIC NOT NULL,
    DTA_INCLUSAO DATE,
    VLR_PRINCIPAL NUMERIC,
    VLR_ENTRADA NUMERIC,
    QTDE_PRESTACOES NUMERIC,

```

```

VLR_DESP_CARTORIO NUMERIC,
VLR_RENDER_DECL NUMERIC,
STS_PROPOSTA NUMERIC,
DTA_ALT DATE,
PRIMARY KEY(NUM_PROPOSTA),
FOREIGN KEY(ID_CLIENTE)
    REFERENCES CLIENTE(ID_CLIENTE),
FOREIGN KEY(COD_VENDEDOR)
    REFERENCES VENDEDOR(COD_VENDEDOR));

```

```

CREATE TABLE CONTRATO (
    NUM_CONTRATO NUMERIC NOT NULL,
    NUM_PROPOSTA INTEGER NOT NULL,
    DTA_EMISSAO DATE,
    VLR_TOTAL_CONTRATO NUMERIC(14,2),
    QTDE_PARCELAS NUMERIC,
    VLR_PARCELA NUMERIC(14,2),
    NUM_TX_CONTRATO NUMERIC,
    PRIMARY KEY(NUM_CONTRATO),
    FOREIGN KEY(NUM_PROPOSTA)
        REFERENCES PROPOSTA(NUM_PROPOSTA));

```

```

CREATE TABLE ITEM_CONTRATO (
    NUM_CONTRATO NUMERIC NOT NULL ,
    SEQ_ITEM_CONTRATO NUMERIC NOT NULL,
    COD_LOTEAMENTO NUMERIC NOT NULL,
    COD_LOTE NUMERIC NOT NULL,
    VLR_ITEM_CONTRATO NUMERIC(14,2),
    PRIMARY KEY(NUM_CONTRATO, SEQ_ITEM_CONTRATO),
    FOREIGN KEY(NUM_CONTRATO)
        REFERENCES CONTRATO(NUM_CONTRATO),
    FOREIGN KEY(COD_LOTE, COD_LOTEAMENTO)
        REFERENCES LOTE(COD_LOTE, COD_LOTEAMENTO));

```

```

CREATE TABLE PARCELA (
    NUM_PARCELA NUMERIC NOT NULL,
    NUM_CONTRATO NUMERIC NOT NULL,
    DTA_VENCIMENTO DATE,
    DTA_PAGAMENTO DATE,
    VLR_PARCELA NUMERIC(14,2),

```

```
VLR_TOTAL_PAGO NUMERIC(14,2),  
VLR_DESCONTO_PARCELA NUMERIC(14,2),  
VLR_MULTA_PARCELA NUMERIC(14,2),  
PRIMARY KEY(NUM_PARCELA, NUM_CONTRATO),  
FOREIGN KEY(NUM_CONTRATO)  
REFERENCES CONTRATO(NUM_CONTRATO));
```