

UNIVERSIDADE DO VALE DO RIO DOS SINOS — UNISINOS
UNIDADE ACADÊMICA GRADUAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

EDUARDO FERRARI COPAT

**COLOPH: UMA FERRAMENTA CASE PARA ENSINO DE BANCO DE DADOS
RELACIONAL**

São Leopoldo
2013

Eduardo Ferrari Copat

**COLOPH: UMA FERRAMENTA CASE PARA ENSINO DE BANCO DE DADOS
RELACIONAL**

Trabalho de Conclusão de Curso apresentado
como requisito parcial para a obtenção do
título de Bacharel em Ciência da Computação
pela Universidade do Vale do Rio dos Sinos —
UNISINOS

Orientador:
Prof. Msc. Gilberto Irajá Müller

São Leopoldo
2013

AGRADECIMENTOS

Gostaria de agradecer, primeiramente, ao apoio essencial dos meus pais. Jamais teria chego aonde estou sem o seu suporte, o qual muitas vezes é invisível aos filhos.

Ao meu irmão, que no futuro irá fazer um trabalho de conclusão melhor que o meu.

Agradeço também a minha namorada Larissa, pela compreensividade do momento do trabalho de conclusão, a qual estamos compartilhando a mesma experiência, e também pelas palavras motivadoras.

Não vou agradecer aos meus amigos, porque eles são ciência, mas ao contrário. Eles são fonte de constante distração e desvirtuação, e me ajudaram a não focar no trabalho de conclusão.

E por fim, mas não menos importante, o meu orientador Gilberto, pessoa que admiro muito, e que sempre esteve disponível para me ajudar de bom grado, com suas extensas e detalhadas revisões.

RESUMO

CONTEXTO: O conhecimento de banco de dados é importante para a atuação de um profissional da tecnologia da informação, tal educação pode ser obtida a partir da disciplina de banco de dados, obrigatória na maioria dos cursos de computação.

PROBLEMA: Não existe uma ferramenta CASE baseada em web para uso acadêmico a qual permite a modelagem de banco de dados com uma ferramenta de diagramação, e que também contemple consultas em álgebra relacional e SQL.

SOLUÇÃO: É proposto o desenvolvimento de um protótipo de uma ferramenta CASE acadêmica, que será disponibilizada via web, e que permita a modelagem conceitual, lógica e física de banco de dados, assim como transformação entre modelos e consulta em SQL e em álgebra relacional.

MÉTODO PROPOSTO: Neste trabalho, inicialmente será realizado uma consulta ao referencial teórico de banco de dados, com foco na modelagem de dados e em consultas SQL e álgebra relacional, assim como pontos relacionados à utilização de ferramentas CASE e educação a distância. Posteriormente serão aplicados estudos de caso com o protótipo e avaliações qualitativas e quantitativas, para assim, analisar os benefícios ao ser utilizado o protótipo, tanto para os alunos, como para professores. Além disso, será realizado um estudo comparativo com ferramentas correlatas ao propósito deste estudo.

Palavras-chave: Banco de dados. Educação à distância. Ferramentas de aprendizado.

ABSTRACT

CONTEXT: Database knowledge is a significant aspect for technology information professionals, this education may be obtained from the database course, which is obligatory in most computing related university courses.

PROBLEM: There is no web-based CASE tool with academic purposes which allows database modeling with a diagramming tool, and also contemplates relational algebra and SQL queries.

SOLUTION: It is proposed an academic web-based CASE tool prototype development, which allows conceptual, logic and physical database modeling, along with model transformation and relational algebra and SQL queries.

PROPOSED METHOD: In this work, a theoretical database study is performed, focused in data modeling and relational algebra and SQL queries, and points regarding CASE tools and distance education. After this, study cases will be applied with the prototype, along with qualitative and quantitative analysis. In this way, it is pretended to evaluate the prototype benefits, for both teachers and students. Besides, it will be performed a comparative study with similar correlated tools.

Keywords: Database. Distance education. Learning tools.

LISTA DE FIGURAS

Figura 1 – Etapas de um projeto de banco de dados	13
Figura 2 – Visão geral de um sistema de banco de dados	19
Figura 3 – A arquitetura de três níveis ANSI/SPARC	20
Figura 4 – Representação de entidades em um DER	22
Figura 5 – Representação de atributos em um DER	23
Figura 6 – Representação de atributos compostos em um DER	23
Figura 7 – Representação de atributos multivalorados em um DER	24
Figura 8 – Representação de atributos derivados em um DER	24
Figura 9 – Representação de relacionamento em um DER	25
Figura 10 – Representação de um auto-relacionamento em um DER	25
Figura 11 – Representação de um relacionamento ternário em um DER	26
Figura 12 – Diagrama de ocorrências no relacionamento Lotação	27
Figura 13 – Representação das cardinalidades de um relacionamento em um DER	28
Figura 14 – Representação de uma entidade fraca e um relacionamento identificador em um DER	29
Figura 15 – Representação de relacionamento em um DER	30
Figura 16 – Tabela Estudante	31
Figura 17 – Relação Estudante	31
Figura 18 – Relação Estudante com chave primária	32
Figura 19 – Relação Estudante e Curso	33
Figura 20 – A chave alternativa CPF na relação Empregado	34
Figura 21 – Relações na notação IE	36
Figura 22 – Relacionamento entre as relações Pessoa e Carteira de habilitação	36
Figura 23 – Relacionamento entre as relações Sala de Reuniões e Empregado	37
Figura 24 – Possíveis combinações de relacionamento na notação IE	38
Figura 25 – Possíveis notações de subtipos na notação IE	39
Figura 26 – A chave estrangeira na notação IE no ERWin	39

Figura 27 – Regras para a transformação de relacionamentos binários do modelo ER para o modelo relacional	42
Figura 28 – Comportamento da operação de seleção	44
Figura 29 – Resultado da operação de seleção	45
Figura 30 – Funcionamento da operação de projeção	46
Figura 31 – Resultado da operação de projeção	46
Figura 32 – Funcionamento da operação de união	47
Figura 33 – Resultado da operação de união	48
Figura 34 – Comportamento da operação de intersecção	48
Figura 35 – Funcionamento da operação de diferença	49
Figura 36 – Resultado da operação de diferença	50
Figura 37 – Comportamento da operação de produto cartesiano	50
Figura 38 – Resultado da operação de produto cartesiano	51
Figura 39 – Funcionamento da operação de renomeação	52
Figura 40 – Relações Empregado e Empregado Tempo Integral	54
Figura 41 – Relações Empregado e Empregado Tempo Integral	54
Figura 42 – Junção externa à esquerda	54
Figura 43 – Junção externa à direita	55
Figura 44 – Junção externa total	55
Figura 45 – Comportamento da operação de divisão	56
Figura 46 – Ferramenta CASE ERWin	74
Figura 47 – Interface para a solução de problemas envolvendo a modelagem ER na ferramenta ACME-DB	77
Figura 48 – Modelagem conceitual na ferramenta BrModelo	78
Figura 49 – Ferramenta elaborada por Batmaz e Hinde (2007)	79
Figura 50 – Ferramenta elaborada por Kung e Tung (2006)	80
Figura 51 – O KERMIT	81
Figura 52 – O protótipo de ambiente de apoio ao processo de ensino aprendizagem de banco de dados: módulo SQL	82
Figura 53 – Ferramenta de correção automática de consultas em álgebra relacional	83

Figura 54 – Fluxo do método de pesquisa	86
Figura 55 – Arquitetura do protótipo	88
Figura 56 – Cronograma	90

LISTA DE TABELAS

LISTA DE SINTAXES

Sintaxe 1 – Convenções para nomear relações - exemplo Estudante	35
Sintaxe 2 – Convenções para nomear relações - exemplo Curso	35
Sintaxe 3 – Operação de seleção	44
Sintaxe 4 – Operação de seleção com comparações	45
Sintaxe 5 – Operação de projeção	46
Sintaxe 6 – Operação relacional composta	47
Sintaxe 7 – Operação de união	48
Sintaxe 8 – Operação relacional de intersecção	49
Sintaxe 9 – Operação de diferença	49
Sintaxe 10 – Operação relacional de produto cartesiano	51
Sintaxe 11 – Renomeação	52
Sintaxe 12 – Designação	52
Sintaxe 13 – Operação com produto cartesiano para exemplificar junção natural	53
Sintaxe 14 – Junção natural	53
Sintaxe 15 – Obtendo agências do Brooklyn	55
Sintaxe 16 – Obtendo clientes	56
Sintaxe 17 – Operação de divisão	56
Sintaxe 18 – Operação de projeção generalizada	57
Sintaxe 19 – Funções agregadas	58
Sintaxe 20 – Operação de inclusão	58
Sintaxe 21 – Operador de atualização	58
Sintaxe 22 – Operação de atualização	59
Sintaxe 23 – Operação de exclusão	59
Sintaxe 24 – Definição de um esquema em SQL	60
Sintaxe 25 – Criação de uma tabela em SQL	61
Sintaxe 26 – Criação de uma tabela em SQL denotando um esquema	61
Sintaxe 27 – Criação de um domínio em SQL	62
Sintaxe 28 – Declaração de chave primária em SQL	63
Sintaxe 29 – Declaração de chave alternativa em SQL	63
Sintaxe 30 – Declaração de chave estrangeira em SQL	64
Sintaxe 31 – Inserção em SQL	65
Sintaxe 32 – Inserção alternativa em SQL	65
Sintaxe 33 – Modificação em SQL	65
Sintaxe 34 – Modificação alternativa em SQL	66
Sintaxe 35 – Remoção em SQL	66
Sintaxe 36 – Bloco SELECT-FROM-WHERE	66
Sintaxe 37 – Consulta básica em SQL	67
Sintaxe 38 – Consulta básica com conectivos em SQL	67
Sintaxe 39 – Consulta com junção em SQL	68
Sintaxe 40 – Consulta em SQL	69
Sintaxe 41 – Consulta em SQL - Exemplo 2	69
Sintaxe 42 – Operador de união	70
Sintaxe 43 – Operador de intersecção	70
Sintaxe 44 – Operador except	71
Sintaxe 45 – Funções agregadas em SQL	71
Sintaxe 46 – Funções agregadas em SQL - exemplo 2	72

LISTA DE SIGLAS

SGBD	Sistema de Gerência de Banco de Dados
SQL	Structured Query Language
ER	Entidade-Relacionamento
DER	Diagrama Entidade-Relacionamento
CASE	Computer Aided Software Engineering
IE	Information Engineering
IDEF1X	Integration DEFinition for Information Modeling (IDEF1X)
DDL	Data Definition Language
DML	Data Manipulation Language
EAD	Educação A Distância
FK	Foreign Key

SUMÁRIO

1 INTRODUÇÃO	13
1.1 Ferramentas CASE	15
1.2 Disciplina de banco de dados	16
1.3 Motivação e justificativa	16
1.4 Objetivos	16
1.4.1 Objetivos específicos	17
1.5 Delimitação de estudo	17
1.6 Estrutura do trabalho	17
2 REFERENCIAL TEÓRICO	18
2.1 Banco de dados	18
2.1.1 Abstração de dados	20
2.1.2 Modelagem de dados	20
2.1.3 Projeto de banco de dados	21
2.2 O modelo conceitual	22
2.2.1 Modelagem entidade-relacionamento	22
2.2.2 Entidades	22
2.2.3 Atributos	22
2.2.4 Relacionamentos	24
2.2.5 Entidades fracas	28
2.2.6 Generalização/especialização	29
2.3 O modelo lógico	30
2.3.1 Modelo relacional	30
2.3.2 Chaves	31
2.3.3 Restrições de integridade	34
2.3.4 Especificação de um banco de dados relacional	34
2.3.5 Notações para a modelagem relacional	35
2.4 O modelo físico	39
2.5 Transformação entre modelos	40
2.6 Álgebra relacional	43
2.6.1 Seleção	44
2.6.2 Projeção	45
2.6.3 A operação relacional composta	47
2.6.4 União	47
2.6.5 Intersecção	48
2.6.6 Diferença	49
2.6.7 Produto cartesiano	50
2.6.8 Renomeação	51
2.6.9 Designação	52
2.6.10 Junção natural	53
2.6.11 Junção externa	53
2.6.12 Divisão	55
2.6.13 Projeção generalizada	57
2.6.14 Funções agregadas	57
2.6.15 Inclusão	58
2.6.16 Atualização	58

2.6.17 Exclusão	59
2.7 SQL	59
2.7.1 Esquemas	60
2.7.2 Data Definition Language	60
2.7.3 Data Manipulation Language	64
2.8 Ferramentas CASE	72
2.8.1 Ferramentas CASE para modelagem de banco de dados	72
2.9 Educação a distância	74
 3 TRABALHOS RELACIONADOS	 76
3.1 ACME-DB	76
3.2 BrModelo	77
3.3 A web-based semi-automatic assessment tool for conceptual database diagram	78
3.4 A web-based tool to enhance teaching/learning database normalization	79
3.5 KERMIT	80
3.6 Um protótipo de ambiente de apoio ao processo de ensino aprendizagem de banco de dados: módulo SQL	81
3.7 An automatic correction tool for relational algebra queries	82
 4 METODOLOGIA DE PESQUISA	 84
 5 PROTÓTIPO E TECNOLOGIAS UTILIZADAS	 87
 6 CONCLUSÃO	 89
6.1 Conclusão final	89
6.2 Trabalhos futuros	89
6.3 Cronograma	89
 REFERÊNCIAS	 91

1 INTRODUÇÃO

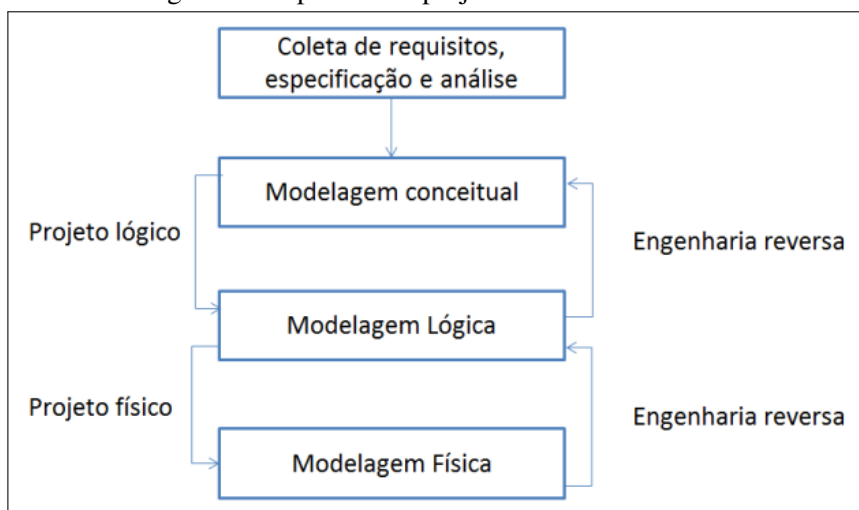
Um banco de dados é uma coleção de dados inter-relacionados; por dados, entendem-se fatos que possuem um significado e podem ser armazenados. Bancos de dados são amplamente utilizados em diversos segmentos da indústria e possuem o objetivo de representar características de um determinado cenário do mundo real, também conhecido como minimundo. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006; ELMASRI; NAVATHE, 2010).

Segundo Silberschatz, Korth e Sudarshan (2006, p. 4), uma das características principais da utilização de um banco de dados é prover alguma forma de abstração de dados. A abstração de dados, neste caso, supre informações relacionadas à organização e o seu armazenamento de dados. Para Elmasri e Navathe (2010, p. 30), pode-se obtê-la a partir de um modelo de dados, isto é, uma coleção de conceitos que são utilizados para a descrição da estrutura de um banco de dados; por estrutura, entende-se seus tipos de dados, relacionamentos e restrições.

O conhecimento de um modelo de dados, ou seja, conhecer técnicas de modelagem de dados, é importante para usuários de banco de dados como administradores e desenvolvedores de software, pois proveem aplicabilidade para diversos segmentos da indústria, assim como facilitam a comunicação com o usuário final. (TEOREY et al., 2010).

Conforme a Figura 1, o projeto de um banco de dados inicia com a fase de coleta de requisitos, especificação e análise; estes processos são detalhados e traduzidos em um modelo conceitual. (RAMAKRISHNAN; GEHRKE, 2003, p. 26).

Figura 1: Etapas de um projeto de banco de dados



Fonte: Adaptado de Elmasri e Navathe (2010, p. 201)

Heuser (2001, p. 5) destaca que "Um modelo conceitual é uma descrição do banco de dados de forma independente de implementação em um SGBD¹ [...] A técnica mais difundida de modelagem conceitual é a abordagem entidade-relacionamento (ER).".

¹"Um Sistema de Gerência de Banco de Dados é constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados". (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 1).

A modelagem conceitual é tida como de alto nível e de grande capacidade semântica, pois consegue expressar, com suas representações, os requisitos elaborados com facilidade, logo, pode ser utilizada como forma de comunicação com usuários finais de um sistema.

Conforme Silberschatz, Korth e Sudarshan (2006), o modelo ER desenvolvido por Chen (1976), baseia-se em premissas do mundo real, onde conjuntos de objetos do minimundo os quais deseja-se armazenar informações são considerados entidades, que são unicamente identificados em relação aos outros objetos. As associações entre as entidades são denominadas relacionamentos como, por exemplo, definir o relacionamento denominado locação, onde se estabelece que um departamento possui várias pessoas, e uma pessoa pode estar alocada em um departamento.

A próxima etapa de um projeto de banco de dados é a criação do modelo lógico, que descreve a estrutura de um banco de dados. Neste passo, mapeia-se o modelo de alto nível previamente elaborado pela modelagem conceitual para o modelo do SGBD escolhido. A abordagem mais utilizada é o modelo relacional proposto por Codd (1970) e sua popularidade se deve à facilidade de modificação de sua estrutura, que é simples e versátil, e pela produtividade de programadores que o utilizam devido a sua simples estrutura tabular e formas de consulta aos dados. (GARCIA-MOLINA; ULLMAN; WIDOM, 2008; RAMAKRISHNAN; GEHRKE, 2003).

A abordagem relacional consiste em um conjunto de relações também conhecidas como tabelas, onde dados correlacionados são armazenados em tuplas (linhas); uma tabela possui atributos (colunas), os quais auxiliam na interpretação dos valores em cada tupla. De forma similar ao modelo conceitual, relacionamentos também ocorrem no modelo relacional. (ELMASRI; NAVATHE, 2010; HEUSER, 2001).

Após a modelagem lógica, pode-se gerar o modelo físico, onde são descritos detalhes técnicos de armazenamento interno das estruturas e o uso de índices, exclusivo para cada SGBD. Neste ponto, também é definido o tipo de dado do atributo, ou seja, o seu domínio. (BURLESON; KELLY, 2004).

Seguindo as etapas do projeto do banco de dados, e após um possível refinamento do modelo de dados, implementa-se o projeto em um SGBD e, para isto, utiliza-se a linguagem SQL (Structured Query Language), que é a linguagem para a criação, manipulação e consulta de dados em um SGBD relacional (RAMAKRISHNAN; GEHRKE, 2003, p. 100).

O SQL pode ser considerado o responsável pela popularidade dos SGBDs relacionais; uma de suas vantagens é quando se decide trocar de SGBD relacional, pois o custo da troca é baixo devido à padronização da linguagem, com isso, pode-se utilizar as mesmas instruções SQL. (ELMASRI; NAVATHE, 2010, p. 87).

Garcia-Molina, Ullman e Widom (2008) afirmam que o cerne do SQL utiliza conceitos da álgebra relacional. Segundo Silberschatz, Korth e Sudarshan (2006, p. 69), "A álgebra relacional é uma linguagem de consultas procedural. Consiste em um conjunto de operações tendo como entrada uma ou duas relações e produzindo, como resultado, uma nova relação". Mui-

tas instruções SQL são baseadas neste conjunto de expressões de álgebra relacional que foram modificadas para uma forma mais fácil de entendimento.

1.1 Ferramentas CASE

Para facilitar o projeto de um banco de dados, ferramentas CASE² propiciam um ambiente que utiliza metodologias estruturadas e interfaces gráficas que auxiliam na automação de tarefas do ciclo de um projeto, assim como diminuem custos de desenvolvimento e tornam a manutenção mais prática. Nestas ferramentas, é possível modelar o banco de dados utilizando a abordagem entidade-relacionamento e relacional. (ROB; CORONEL, 2009, p. 632).

As ferramentas comerciais mais populares, no que se concede a modelagem, em sua grande parte oferecem apenas a modelagem lógica e física; são softwares que devem ser instalados no computador do usuário e têm um custo de licença (POOLET, 2010), porém, existem versões gratuitas para a comunidade, mas com limitações, tais como o CA ERwin Data Modeler Community Edition (SYBASE, 2013), Toad Data Modeler Community (WORLD, 2013) e o Oracle SQL Developer Data Modeler (ORACLE, 2013). Teorey et al. (2010) destacam que softwares comerciais apresentam competência na modelagem lógica e na sua transcrição para os banco de dados físicos.

Todavia, existem diversas ferramentas acadêmicas que auxiliam a resolução dos exercícios em bancos de dados, porém, Pereira (2011) analisa este conjunto de ferramentas, e conclui que apenas uma minoria agrega as técnicas de projeto de banco de dados em um único software. Também analisa que poucas ferramentas são baseadas em web, caminho oposto se comparado ao crescimento do uso de ferramentas online para o ensino, o qual fornece novas formas de realização de exercícios e práticas. (BARROS, 2010; PEREIRA et al., 2012).

Entre as ferramentas acadêmicas mais promissoras, pode-se destacar o BrModelo (CÂNDIDO, 2008), que é uma ferramenta de modelagem conceitual e lógica de banco de dados e que realiza transformação entre modelos. Outra ferramenta promissora é o KERMIT (SURAWERA; MITROVIC, 2002), que é um sistema de tutoria inteligente para modelagem conceitual utilizando o modelo ER. Ambos necessitando de instalação local.

Destaca-se também o ACME-DB, o qual fornece um ambiente web completo para o ensino de banco de dados, com tópicos de modelagem conceitual e lógica, diagramas de classe, normalização, álgebra relacional e SQL, porém, não fornece uma ferramenta CASE para a modelagem de banco de dados. (MASÓ; GESA; MATEMÁTICA APLICADA, 2010).

² *Computer Aided Software Engineering*, que significa, em uma tradução literal, engenharia de software suportada por computador (SILVA; VIDEIRA, 2001)

1.2 Disciplina de banco de dados

A disciplina de banco de dados consta como básica para todos os cursos de computação segundo as Diretrizes Curriculares Nacionais estabelecidas pelo MEC (2003, p. 20), assim como internacionalmente, no currículo de ciência da computação elaborado pela ACM³ e IEEE⁴. (CASSEL et al., 2008, p. 86).

Dentre as atividades elaboradas na disciplina, inclui-se o aprendizado dos conceitos de modelagem e projeto de banco de dados (SBC, 2005, p. 11), conhecimento o qual todo profissional da área da tecnologia da informação deve possuir (HEUSER, 2001), além de consultas em SQL e álgebra relacional.

1.3 Motivação e justificativa

Em aulas de banco de dados, os estudantes necessitam realizar atividades que envolvem a modelagem entidade-relacionamento e relacional de um determinado problema, o que os leva a utilização de uma ferramenta CASE; os alunos, em sua maioria, fazem uso de ferramentas comerciais para solucionar as questões propostas pelos professores. (DIETRICH; URBAN, 1996; REGUERAS et al., 2007).

Bogdanović et al. (2008) apontam que os alunos recorrem ao uso de ferramentas comerciais devido a sua popularidade e aceitação no mercado, porém, seu uso apresenta dificuldades para usuários inexperientes e podem desviar o foco da aprendizagem de modelagem de banco de dados.

Pereira e Resende (2012) relatam que as inovações tecnológicas recentes exigem novos estímulos aos alunos e, neste âmbito, faz-se necessário uma ferramenta CASE baseada em web para uso acadêmico que contemple aspectos para o ensino da modelagem de um banco de dados, tais como:

- Modelagem conceitual, lógica e física;
- Transformação entre modelos;
- Consulta em SQL;
- Consulta em álgebra relacional.

1.4 Objetivos

Desenvolver um protótipo de uma ferramenta CASE para o ensino de banco de dados disponível em ambiente web, tendo como foco a modelagem conceitual, lógica e física, transformação entre modelos e a consulta em SQL e álgebra relacional.

³Association for Computing Machinery - <http://www.acm.org/>

⁴IEEE Computer Society - <http://www.ieee.org>

1.4.1 Objetivos específicos

1. Analisar os benefícios de uma ferramenta de ensino de banco de dados em sala de aula;
2. Analisar os benefícios de uma ferramenta de ensino com professores de banco de dados;
3. Promover a integração entre aluno e professor na disciplina de banco de dados.

1.5 Delimitação de estudo

Devido aos inúmeros modelos de dados que podem ser utilizados tanto para a modelagem conceitual como para a modelagem lógica (HEUSER, 2001, p. 5), este trabalho limitar-se-á ao uso do modelo de Chen (1976) para a modelagem conceitual, e o modelo relacional proposto por Codd (1970), com a notação Information Engineering (IE), proposta por Martin e Finkelstein (1981).

1.6 Estrutura do trabalho

Este trabalho está dividido da seguinte forma: o segundo capítulo aborda o referencial teórico necessário para este estudo, onde será apresentada uma visão geral sobre banco de dados e modelagem de dados, após, será abordado a modelagem conceitual, lógica e física de banco de dados; em seguida, será tratado os temas de consulta em SQL, álgebra relacional e ferramentas CASE; para o fechamento do capítulo será abordado o tema de educação a distância.

O terceiro capítulo elenca os trabalhos relacionados a este estudo proposto, isto é, estudos que propõem uma abordagem similar a este. O quarto capítulo irá tratar sobre a metodologia de pesquisa proposta neste estudo e, o quinto capítulo, abordará as tecnologias que serão utilizadas para o protótipo. O último capítulo apresentará as conclusões parciais deste trabalho, juntamente com a apresentação dos trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo tem como objetivo apresentar a base teórica que será utilizada para compreensão dos temas propostos por este trabalho. É apresentada uma visão geral sobre banco de dados e modelagem de dados, estudos no que envolve a modelagem conceitual, lógica e física de banco de dados; em seguida, trata-se sobre a consulta em SQL, consulta em álgebra relacional e ferramentas CASE; por fim, aborda-se o tema de educação a distância

2.1 Banco de dados

Um banco de dados é uma coleção de dados inter-relacionados; por dados, entendem-se fatos que possuem um significado e podem ser armazenados, como por exemplo, uma agenda de telefones ou os alunos de uma universidade. Esta coleção de dados deve ter um significado intrínseco e lógico, possuir aspectos e interações do mundo real como fonte de dados e ser de interesse à um determinado grupo de usuários. Alterações que ocorram nestes aspectos devem ser refletidos no banco de dados. (ELMASRI; NAVATHE, 2010, p. 4).

Silberschatz, Korth e Sudarshan (2006, p. 1) define que "um Sistema de Gerência de Banco de Dados (SGBD) é constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados". O SGBD facilita as tarefas de manutenção de um banco de dados, como a gerência de grande volume de informações, armazenamento de metadados, segurança dos dados armazenados e compartilhamento de dados. (TEOREY et al., 2010, p. 2).

De acordo com Silberschatz, Korth e Sudarshan (2006, p. 2), as informações poderiam ser armazenadas em um sistema de processamento de arquivos, comum em sistemas operacionais, porém esta escolha apresenta diversas desvantagens, tais como:

- **redundância de dados:** informações podem estar representadas em mais de um lugar no sistema, aumentando custos de armazenamento e acesso;
- **inconsistência de dados:** cópias dos dados poderão diferenciar-se do original ao longo do tempo;
- **isolamento de dados:** com os dados armazenados em diversos arquivos, os arquivos podem apresentar formatos diferentes, dificultando a recuperação dos dados;
- **problemas de integridade:** os valores dos dados em um banco de dados devem ser consistentes, ou seja, aplicar restrições aos dados em diversos arquivos pode ser um problema;
- **problemas de atomicidade:** sistemas computacionais são sujeitos a falhas, as operações envolvendo os dados devem ser atômicas, isto é, devem ser efetuadas completamente ou não. Esta propriedade é de difícil obtenção em um sistema de processamento de arquivos;

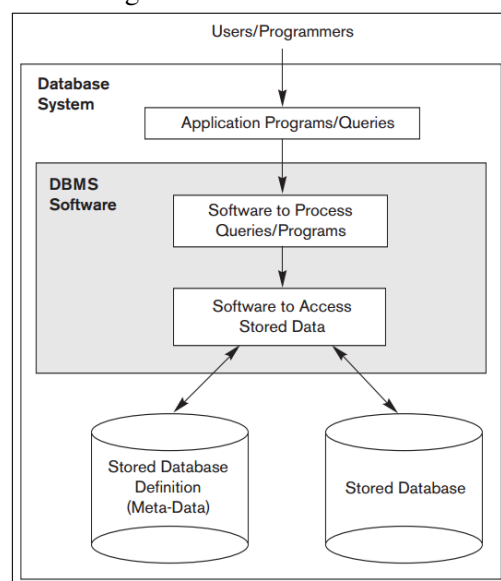
- **problemas de concorrência:** deve existir um rígido controle de acesso e atualizações simultâneas nos dados, sem um mecanismo bem definido para isso, a concorrência torna-se problemática;
- **problemas de segurança:** em muitos casos, apenas certos usuários estão autorizados a acesso a determinados dados. Tais regras de segurança são difíceis de serem definidas em um sistema de processamento de arquivos.

Nestes e outros cenários de dificuldades, a necessidade de um SGBD foi criada. Para Garcia-Molina, Ullman e Widom (2008, p. 1), é esperado que um SGBD:

- permita que usuários criem banco de dados baseados em esquemas, isto é, a estrutura lógica dos dados;
- permita que usuários consultem e modifiquem os dados;
- garanta o armazenamento de vastas quantidades de dados em um longo período de tempo;
- permita durabilidade, isto é, garantir que os dados sejam recuperados caso ocorram erros;
- controle do acesso aos dados, mesmo com interações e acessos simultâneos dos usuários.

A união entre o banco de dados e o SGBD forma um sistema de banco de dados. (EL-MASRI; NAVATHE, 2010, p. 7). A Figura 2 apresenta uma visão geral de um sistema de banco de dados.

Figura 2: Visão geral de um sistema de banco de dados

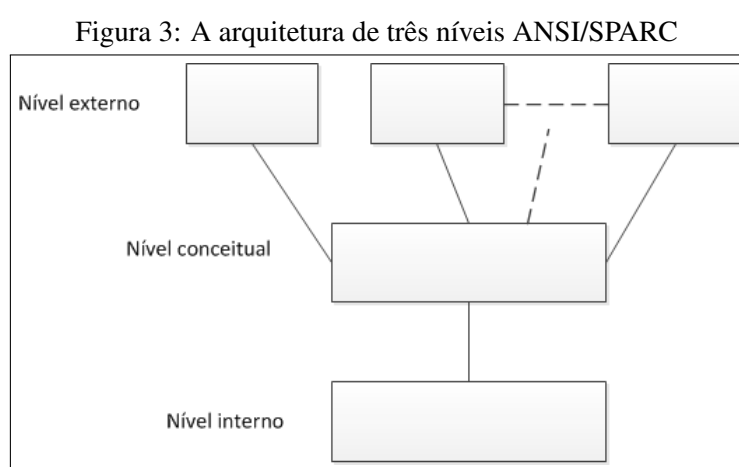


Fonte: Elmasri e Navathe (2010, p. 7)

2.1.1 Abstração de dados

Segundo Silberschatz, Korth e Sudarshan (2006, p. 4), uma das características principais da utilização de um banco de dados é prover alguma forma de abstração de dados. A abstração de dados, neste caso, supre informações relacionadas à organização e o seu armazenamento de dados. Tendo em vista que diversos usuários de banco de dados não tem conhecimento em computação, é necessário omitir certa complexidade em camadas de abstração.

Conforme Date (2003, p. 34) a arquitetura ANSI/SPARC¹ classifica essa abstração, dividindo-a em três níveis, ilustrados pela Figura 3:



Fonte: Adaptado de Date (2003, p. 34)

Conforme a Figura 3, os níveis podem ser definidos como:

- **nível interno:** preocupa-se em como os dados estão armazenados no sistema;
- **nível conceitual:** ignora os detalhes de armazenamento dos dados, e tem como foco descrições em alto nível do banco de dados;
- **nível externo:** o mais próximo dos usuários, e preocupa-se com o modo como os dados são visualizados pelos mesmos.

2.1.2 Modelagem de dados

Para Elmasri e Navathe (2010, p. 30), a abstração pode ser obtida a partir de um modelo de dados, isto é, uma coleção de conceitos que são utilizados para a descrição da estrutura de um banco de dados; por estrutura, entende-se seus tipos de dados, relacionamentos e restrições.

Ramakrishnan e Gehrke (2003, p. 10, tradução nossa) consideram que um modelo de dados "é um conjunto de descrições em alto nível dos dados", e Heuser (2001, p. 5) complementa,

¹A arquitetura foi proposta por um grupo de estudos em banco de dados, e foi elaborada por Tsichritzis e Klug (1978).

definindo como uma "descrição formal da estrutura de um banco de dados". A criação de um modelo de dados, isto é, a modelagem de dados, se dá a partir dos aspectos que se desejam ser representados do mundo real, esses aspectos também são conhecidos como domínio da aplicação ou minimundo. (HALPIN; MORGAN, 2008, p. 6).

Os modelos de dados podem ser divididos em duas categorias: (i) modelos de alto nível ou modelos conceituais, os quais se aproximam com a realidade pela qual os usuários interpretam os dados, e (ii) modelos de baixo nível ou modelos de dados físicos, que possuem detalhes da implementação do armazenamento dos dados. (ELMASRI; NAVATHE, 2010, p. 30). Modelos de alto nível são descritos nas seções 2.2 e 2.3 e modelos de baixo nível são descritos na seção 2.4.

A modelagem de dados é um dos processos mais importantes de um projeto de banco de dados, tratando-se de uma atividade desafiante, pois envolve análise, conhecimento e experiência; sua qualidade impacta diretamente os programas que utilizam o banco de dados. (SIMSION; WITT, 2004).

2.1.3 Projeto de banco de dados

Conforme Ramakrishnan e Gehrke (2003, p. 26), o projeto de um banco de dados pode ser dividido em seis etapas:

1. **análise de requisitos:** primeiramente, deve-se entender quais dados serão armazenados no banco de dados, e quais sistemas serão construídos em cima dos mesmos. Nesta etapa, deve-se entender o que os usuários esperam que o banco de dados ofereça;
2. **modelagem conceitual:** a partir das informações coletadas da etapa anterior, é possível criar um modelo de alto nível que descreve o banco de dados. Neste modelo, deve ser possível que tanto usuários como desenvolvedores consigam entender os processos que envolvem os dados, deste modo, o modelo pode ser refinado até com o suporte de pessoas que não tem um conhecimento técnico;
3. **modelo lógico:** nesta etapa, mapeia-se o modelo conceitual previamente elaborado para o modelo do SGBD escolhido. A abordagem mais popular é o modelo relacional, descrito na seção 2.4;
4. **refinamento:** o refinamento, neste caso, não se refere à reanálise dos requisitos ou de um modelo de um alto nível, mas sim no uso de teorias como a normalização, que é uma técnica que visa eliminar a redundância nos dados. (HEUSER, 2001, p. 114);
5. **modelo físico:** esta etapa pode envolver a construção de índices em tabelas, assim como aplicação de técnicas para otimizar o desempenho do sistema;
6. **segurança:** certos dados só devem ser acessados por determinados usuários. É parte do projeto de banco de dados identificar e assegurar as regras de acesso.

2.2 O modelo conceitual

Heuser (2001, p. 6) define um modelo conceitual como "um modelo de dados abstrato, que descreve a estrutura de um banco de dados de forma independente de um SGBD particular". Também cita que a abordagem mais popular é a entidade-relacionamento (ER), este tipo de modelagem, proposto por Chen (1976), será o modelo abordado neste estudo.

2.2.1 Modelagem entidade-relacionamento

Conforme Silberschatz, Korth e Sudarshan (2006), o modelo ER baseia-se no entendimento de que o mundo real é formado por objetos chamados entidades, os quais possuem relacionamentos entre eles. Em seu clássico artigo, Chen (1976) propõe o modelo ER como um modelo que incorpora informações semanticamente importantes para o problema em questão.

2.2.2 Entidades

Segundo Elmasri e Navathe (2010, p. 202, tradução nossa), a entidade é o objeto básico do modelo ER, e pode ser definida como "uma coisa no mundo real de existência independente", podendo ser tanto algo físico ou abstrato. A coleção de um tipo particular de entidade é chamada de conjunto de entidades.

Uma pessoa, um departamento, um cliente e um endereço são exemplos de entidades. Em um Diagrama de Entidade-Relacionamento (DER), entidades são representadas por retângulos, exemplificados na Figura 4. (HEUSER, 2001, p. 12).

Figura 4: Representação de entidades em um DER



Fonte: Adaptado de Heuser (2001, p. 12)

2.2.3 Atributos

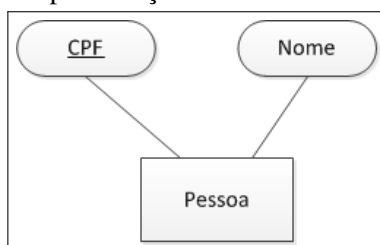
Silberschatz, Korth e Sudarshan (2006, p. 22) definem que "uma entidade é representada por um conjunto de atributos. Atributos são propriedades descritivas de cada membro de um conjunto de entidades". Para cada atributo, existe um conjunto de valores que o atributo se enquadra, denominando-se domínio do atributo.

Conforme Elmasri e Navathe (2010, p. 208), uma importante restrição que as entidades devem possuir são as restrições de unicidade em atributos. Uma entidade, na maioria das vezes, irá possuir um ou mais atributos que serão distintos para cada entidade no conjunto de entidades.

Este atributo é conhecido como atributo chave ou atributo identificador, e seus valores podem ser utilizados para identificar unicamente cada entidade. No caso de uma entidade Pessoa, pode-se definir que o CPF de cada pessoa é considerado um atributo chave, tendo em vista que ele é único. Logo, o atributo faz parte da chave primária da entidade.

Desta forma, atributos ajudam a detalhar informações as quais se deseja representar a partir das entidades. Exemplificando, a entidade Pessoa pode ter seu CPF e Nome como atributos, sendo que o CPF é um atributo chave. Atributos são representados por elipses nos diagramas, sendo que os atributos identificadores tem seu nome sublinhado. Esses conceitos são apresentados na Figura 5. (RAMAKRISHNAN; GEHRKE, 2003, p. 29).

Figura 5: Representação de atributos em um DER



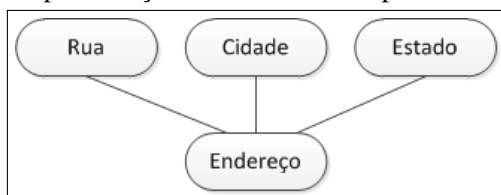
Fonte: Adaptado de Ramakrishnan e Gehrke (2003, p. 30)

2.2.3.1 Tipos de atributos

Para Silberschatz, Korth e Sudarshan (2006, p. 23), atributos podem ser caracterizados pelos seguintes tipos:

- **simples** ou **compostos**: Atributos simples não podem ser divididos em partes, diferentemente de atributos compostos, os quais podem se dividir em outros atributos. Por exemplo, o atributo Idade na entidade Pessoa pode ser considerado um atributo simples, enquanto o atributo Endereço poderia ser separado em Rua, Cidade e Estado, classificando-o como um atributo composto. Atributos compostos são expressos com seus conectados a seus atributos como apresentado na Figura 6:

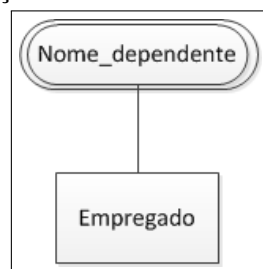
Figura 6: Representação de atributos compostos em um DER



Fonte: Adaptado de Elmasri e Navathe (2010, p. 223)

- **monovalorados** ou **multivalorados**: Diz-se que um atributo é monovalorado quando apenas valores simples são atribuídos. Quando pode-se empregar um conjunto de valores para determinado atributo, considera-se que o atributo é multivalorado. Atributos multivalorados são representados por uma elipse dupla, conforme o exemplo da Figura 7, onde define-se que um Empregado pode ter vários dependentes.

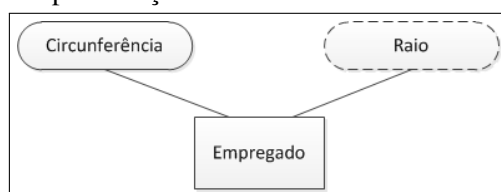
Figura 7: Representação de atributos multivalorados em um DER



Fonte: Adaptado de Elmasri e Navathe (2010, p. 223)

- **nulos**: valores nulos para um atributo podem ser usados quando não se possui um valor para o atributo. Nulo pode significar que o atributo não é aplicável, como por exemplo o Número da carteira de habilitação de uma pessoa que não possui habilitação para dirigir, assim como pode significar um valor desconhecido ou omitido. Não existe representação para o atributo nulo.
- **derivados**: quando o valor de um atributo pode ser derivado de outros atributos ou entidades relacionados. Atributos derivados são expressos como uma elipse com a sua borda pontilhada, como ilustra a Figura 8, onde é possível derivar o Raio de uma Esfera a partir de sua Circunferência.

Figura 8: Representação de atributos derivados em um DER



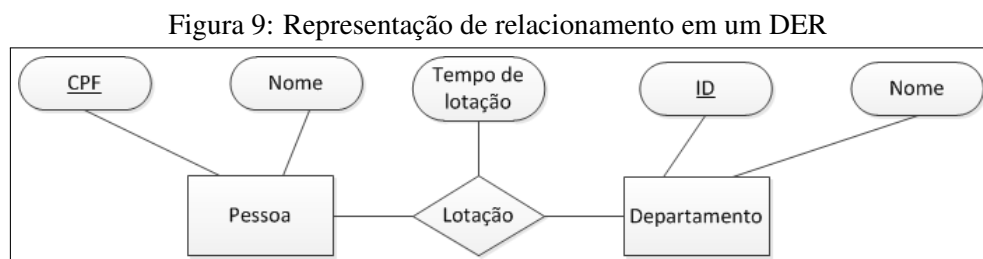
Fonte: Adaptado de Elmasri e Navathe (2010, p. 223)

2.2.4 Relacionamentos

Silberschatz, Korth e Sudarshan (2006, p. 25) definem que "um relacionamento é uma associação entre uma ou várias entidades". Heuser (2001, p. 13) exemplifica: um relacionamento

pode ocorrer quando se deseja saber quais pessoas estão associadas a um determinado departamento em uma organização.

Relacionamentos são representados por um losango em um DER, e similarmente às entidades, também podem possuir atributos. (RAMAKRISHNAN; GEHRKE, 2003, p. 31). Esses conceitos estão ilustrados na Figura 9.

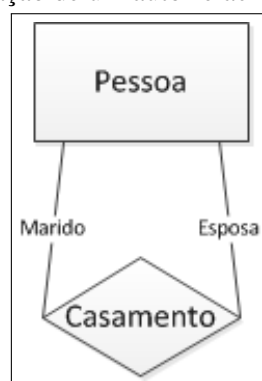


Fonte: Adaptado de Ramakrishnan e Gehrke (2003, p. 30)

2.2.4.1 Auto-relacionamentos

Não é obrigatório que relacionamentos sejam entre entidades diferentes, pois podem existir casos onde uma entidade se relaciona com ela mesma e, neste cenário, tem-se um auto-relacionamento. Porém, um conceito adicional é necessário, pois deve-se identificar o papel da entidade do relacionamento. Por exemplo, o relacionamento Casamento entre duas Pessoas, deve ser identificado pelos papéis de Marido e Esposa, como apontado pela Figura 10.

Figura 10: Representação de um auto-relacionamento em um DER



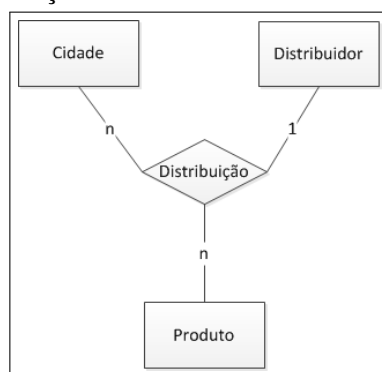
Fonte: Adaptado de Heuser (2001, p. 15)

Neste caso, lê-se que uma Pessoa pode exercer o papel de Marido em um Casamento, assim como uma Pessoa pode exercer o papel de Esposa em um Casamento.

2.2.4.2 Grau de relacionamentos

O grau de um relacionamento é o número de entidades que participam de um relacionamento. Em sua grande maioria, relacionamentos apenas possuem duas entidades, logo, diz-se que o relacionamento é binário. Em outros casos, um relacionamento pode ser ternário, possuindo três entidades. (ELMASRI; NAVATHE, 2010, p. 213). A Figura 11 descreve um relacionamento ternário entre as entidades Cidade, Distribuidor e Produto.

Figura 11: Representação de um relacionamento ternário em um DER



Fonte: Adaptado de Heuser (2001, p. 19)

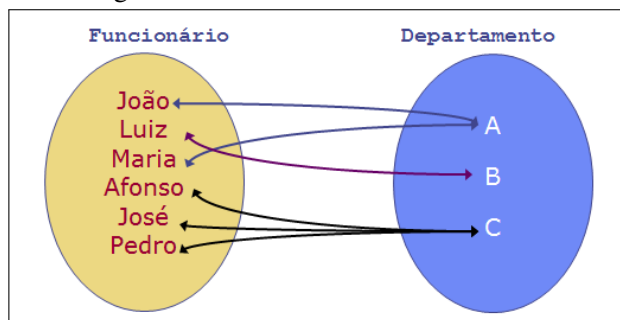
Neste exemplo, observa-se que a entidade Produto se relaciona a partir de uma Distribuição, tanto para a entidade Cidade como a entidade Distribuidor. Pode-se interpretar que um Produto é distribuído em Cidades, e que um Produto é distribuído por um Distribuidor.

2.2.4.3 Cardinalidade de relacionamentos

Uma propriedade importante de um relacionamento é a cardinalidade, isto é, o número de ocorrências das entidades associadas. Por ocorrências, entende-se as associações particulares dentro de um relacionamento.

A Figura 12 apresenta um diagrama de ocorrências, o qual representa as ocorrências de Funcionários e Departamentos em um relacionamento Lotação. Diz-se que existe uma ocorrência do relacionamento Lotação quando é formado um par entre a ocorrência da entidade Funcionário e uma ocorrência da entidade Departamento.

Figura 12: Diagrama de ocorrências no relacionamento Lotação



Fonte: Müller (2011a)

Cardinalidades podem ser classificados em dois tipos: a cardinalidade máxima e a mínima. (HEUSER, 2001, p. 15). Considerando o exemplo da Figura 9, pode-se considerar que cada Pessoa está lotada em apenas um Departamento, e um Departamento pode ter várias Pessoas. Portanto, pode-se dizer que a entidade Pessoa tem cardinalidade máxima **1** no relacionamento *lotação*, enquanto a entidade Departamento tem cardinalidade máxima de **n**. Na cardinalidade máxima, apenas permite-se a cardinalidade **1** e **n**.

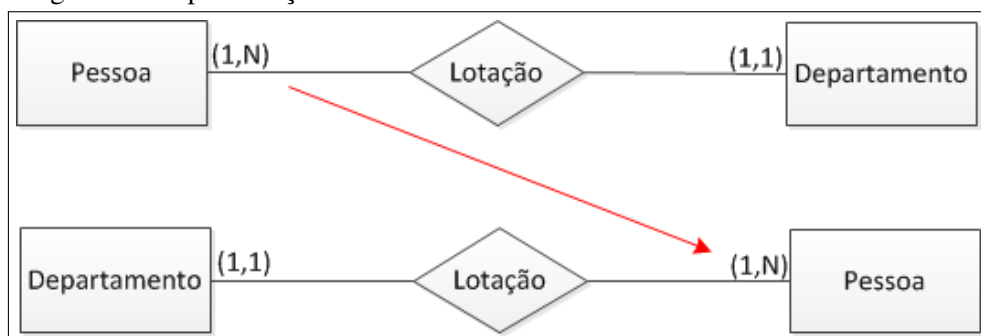
Ainda, conforme Heuser (2001, p. 20), é possível acrescentar em um modelo ER informações sobre a cardinalidade mínima de um relacionamento, isto é, o número mínimo de ocorrências das entidades associadas. Consideram-se apenas duas cardinalidades mínimas:

- **cardinalidade mínima 1**, também conhecida como associação obrigatória, já que se faz obrigatória a associação de uma ocorrência de entidade a cada ocorrência da entidade em questão;
- **cardinalidade mínima 0**, também conhecida como associação opcional, já que não é necessário associar uma ocorrência de entidade a cada ocorrência da entidade em questão.

Em um DER, as cardinalidades de um relacionamento são representadas próximas à entidade oposta a qual o relacionamento se refere. Ela é escrita de forma similar aos pontos utilizados no produto cartesiano, sendo que a ordenação segue a cardinalidade mínima e após a máxima.

Para facilitar o entendimento, a Figura 13 apresenta um DER que descreve um cenário onde um Departamento contém no mínimo uma Pessoa e não ha um limite máximo de Pessoas que podem estar lotadas no Departamento, e onde uma Pessoa deve estar lotada em apenas e obrigatoriamente em um Departamento.

Figura 13: Representação das cardinalidades de um relacionamento em um DER



Fonte: Adaptado de Heuser (2001, p. 16)

2.2.4.4 Mapeamento de cardinalidades e ocorrências

Segundo Silberschatz, Korth e Sudarshan (2006, p. 29), o mapeamento de cardinalidades expressa a quantidade de ocorrências de uma entidade na qual outra entidade pode estar associada via um relacionamento, classificando-as em:

- **um para um (1:1):** uma entidade em A está associada a no máximo a uma entidade em B, e uma entidade em B está associada a no máximo uma entidade em A;
- **um para muitos (1:N):** uma entidade em A está associada a várias entidades em B. Uma entidade em B, entretanto, deve estar associada no máximo a uma entidade em A;
- **muitos para um (N:1):** uma entidade em A está associada a no máximo uma entidade em B. Uma entidade em B, entretanto, pode estar associada a um número qualquer de entidades em A;
- **muitos para muitos (N:N):** uma entidade em A está associada a qualquer número de entidades em B e uma entidade em B está associada a um número qualquer de entidades em A.

2.2.5 Entidades fracas

Silberschatz, Korth e Sudarshan (2006, p. 38) apontam que, em alguns casos, os atributos de uma entidade podem não ser suficientes para formar uma chave primária, isto é, possuir atributos identificadores, neste caso, diz-se que a entidade é fraca. Entidades que possuem um ou mais atributos identificadores próprios, isto é, que possuem chave primária, são denominadas entidades fortes.

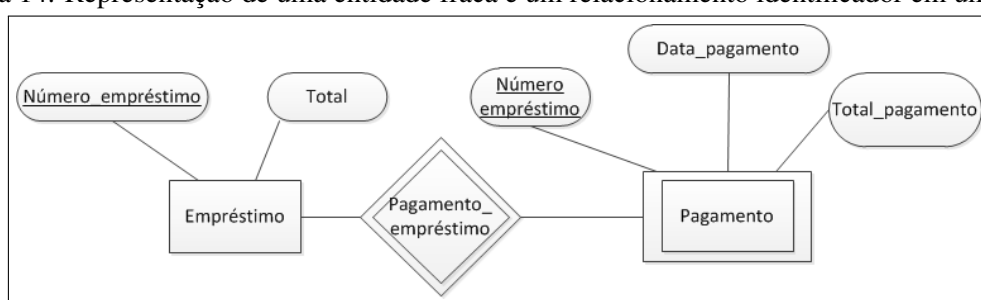
Entidades fracas apenas têm um significado quando fazem parte de um relacionamento, isto é, ocorrências de uma entidade fraca dependem de uma entidade forte para existir, define-se, então, que este é um relacionamento identificador. (HEUSER, 2001, p. 25)

Ainda, conforme Silberschatz, Korth e Sudarshan (2006, p. 38), como a entidade fraca não

possui chave primária, é preciso criar algum meio de distinguir os dados da entidade, neste caso, utiliza-se um atributo identificador para permitir esta distinção. Logo, pode-se dizer que a chave primária de uma entidade fraca é formada pela chave primária de sua entidade forte e seu atributo identificador.

Para exemplificar, supõe-se o uso de duas entidades Empréstimo e Pagamento, onde para um Empréstimo, é possível ter diversos Pagamentos, os quais dependem do Empréstimo para existirem e são discernidos a partir de uma Data. Entidades fracas em um DER são representadas por um retângulo duplo e relacionamentos identificadores são representados por um losango duplo, conforme ilustrado na Figura 14.

Figura 14: Representação de uma entidade fraca e um relacionamento identificador em um DER



Fonte: Adaptado de Silberschatz, Korth e Sudarshan (2006, p. 39)

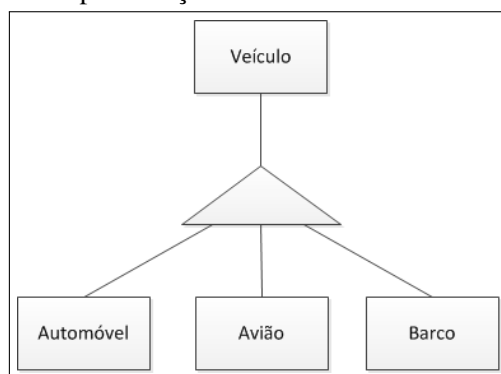
Observa-se que, como a entidade Pagamento é uma entidade fraca, o atributo identificador Número empréstimo de sua entidade forte Empréstimo também torna-se atributo identificador da entidade Pagamento.

2.2.6 Generalização/especialização

Conforme Heuser (2001, p. 28), é possível que uma entidade genérica possua um subconjunto de entidades com propriedades particulares, chamadas de entidades especializadas. A ideia do uso de generalização e especialização vem em conjunto com a herança de propriedades, onde uma entidade especializada possui, além de seus atributos, os atributos de sua entidade genérica.

Aponta-se como exemplo, a entidade genérica Veículo, as quais as entidades especializadas Automóvel, Avião e Barco herdam seus atributos. Generalizações e especializações são representadas por um triângulo isósceles em um DER, conforme a Figura 15.

Figura 15: Representação de relacionamento em um DER



Fonte: Adaptado de Heuser (2001, p. 31)

2.3 O modelo lógico

O modelo lógico é o modelo de dados que representa a estrutura de dados de um SGBD comercial. Neste estudo será enfatizado o modelo relacional, que é a abordagem mais popular de projeto e implementação de SGBDs. (ELMASRI; NAVATHE, 2010, p. 9).

2.3.1 Modelo relacional

Elmasri e Navathe (2010, p. 59) apontam que o modelo relacional, introduzido por Codd (1970), atraiu atenção devido a sua simplicidade e sua fundamentação na matemática. O modelo utiliza o conceito de relações matemáticas, as quais se assimilam a estrutura de uma tabela, juntamente com a teoria dos conjuntos e a lógica de primeira ordem.

Silberschatz, Korth e Sudarshan (2006, p. 61) definem que "um banco de dados relacional é como uma coleção de tabelas, cada uma das quais com um nome único". Uma tabela é formada por linhas e colunas. As linhas de uma tabela contém um conjunto de dados relacionados, e representam um fato que relata a uma entidade ou relacionamento. As colunas e o nome da tabela são utilizados para a interpretação dos dados armazenados nas linhas. (ELMASRI; NAVATHE, 2010, p. 60).

Para exemplo, considera-se a tabela Estudante, apresentada na Figura 16. Suas colunas Numero_estudante, Nome e Curso auxiliam a interpretar os dados de cada linha, baseado nos valores para cada coluna. Observa-se também que os valores de cada coluna são do mesmo tipo de dado.

Figura 16: Tabela Estudante

Numero_estudante	Nome	Curso
17	Daniel	1
8	Vinicius	2

Fonte: Adaptado de Elmasri e Navathe (2010, p. 8)

Na definição formal da terminologia do modelo, uma linha é chamada de tupla, uma coluna é chamada de atributo, e uma tabela é chamada de relação. O tipo de dado que descreve os valores em cada coluna é representado por um domínio. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 62). Este estudo utilizará a definição formal, isto é: tuplas, domínios, atributos e relações.

Elmasri e Navathe (2010, p. 61) definem que "um domínio é um conjunto de valores atômicos", e Silberschatz, Korth e Sudarshan (2006, p. 61) complementam, definindo que "um domínio é atômico se elementos desse domínio são considerados unidades indivisíveis". As restrições de domínio em uma coluna se referem aos valores que podem existir em uma coluna, dado o domínio o qual a coluna está sendo associada. (RAMAKRISHNAN; GEHRKE, 2003, p. 69).

Considerando a relação Estudante, podemos inferir que o domínio do atributo Nome deve apenas conter caracteres. Outros exemplos de domínio podem incluir o número de um telefone (10 dígitos), a idade de um empregado (números naturais) e o formato de um CPF (11 dígitos).

A Figura 17 apresenta a relação Estudante, indicando suas tuplas e atributos. Para aprimorar o exemplo, foi adicionado o atributo Telefone, o qual possui um domínio de 10 dígitos e admite valores nulos, ou seja, que são inexistentes ou desconhecidos.

Figura 17: Relação Estudante

Estudante			
Numero_estudante	Nome	Telefone	Curso
17	Daniel	5481125695	1
8	Vinicius	5185961274	1
13	Leonardo	nulo	2

Fonte: Elaborado pelo autor

2.3.2 Chaves

Para estabelecer relações entre tuplas de relações, é necessário o conceito de chave. Por definição, uma relação é um conjunto de tuplas. Todos os elementos de um conjunto devem ser distintos, logo, o mesmo se aplica a uma relação: todas suas tuplas devem ser distintas. Em um

banco de dados relacional, existem três tipos de chave: a chave primária, a chave alternativa e a chave estrangeira. (HEUSER, 2001; ELMASRI; NAVATHE, 2010).

2.3.2.1 Chave primária

De acordo com Heuser (2001, p. 73), a chave primária é um atributo ou conjunto de atributos cujos valores distinguem tuplas em uma relação. Para exemplificar, pode-se definir que a relação Estudante tem como chave primária o atributo Numero_estudante, tendo em vista que ele será único na relação, conforme é ilustrado na Figura 18.

Figura 18: Relação Estudante com chave primária

Estudante			
Numero_estudante	Nome	Telefone	CodigoCurso
17	Daniel Silva	5481125695	1
8	Vinicius Almeida	5185961274	1
13	Leonardo Oliveira	nulo	2

Fonte: Elaborado pelo autor

2.3.2.2 Chave estrangeira

Conforme Heuser (2001, p. 74), a chave estrangeira é um atributo ou um conjunto de atributos cujos valores necessariamente pertencem a chave primária de outra tabela. A chave estrangeira é responsável por permitir o uso de relacionamentos em um banco de dados relacional.

Introduz-se a relação Curso, para assim, exemplificar o uso da chave estrangeira. A relação Curso armazena informações dos cursos de informática de uma universidade, tendo como atributos o CodigoCurso e o Nome, sendo CodigoCurso chave primária, conforme apresenta a Figura 19.

Figura 19: Relação Estudante e Curso

Estudante			
Numero_estudante	Nome	Telefone	CodigoCurso
17	Daniel Silva	5481125695	1
8	Vinicius Almeida	5185961274	1
13	Leonardo Oliveira	nulo	2

Curso	
CodigoCurso	Nome
1	Ciência da Computação
1	Ciência da Computação
2	Sistemas de Informação

Fonte: Elaborado pelo autor

No banco de dados apresentado pela Figura 19, estabelece-se que todo estudante deve pertencer a um curso. Diz-se que o atributo CodigoCurso é uma chave estrangeira em relação à relação Estudante, logo, valores de CodigoCurso na relação Estudante devem aparecer na relação Curso.

Heuser (2001, p. 74) ainda lista que a chave estrangeira impõe restrições que devem ser garantidas em determinadas situações, tais como:

- quando da inclusão de uma linha na relação que contém a chave estrangeira: O valor da chave estrangeira deve constar no atributo da chave primária a qual se faz referência, no caso de exemplo da Figura 19, um Estudante somente poderia ser adicionado a um curso que já existe;
- quando da alteração do valor da chave estrangeira: A alteração do valor de um atributo da chave primária deve ser refletido na chave estrangeira;
- quando da exclusão de uma linha da tabela que contém a chave primária referenciada pela chave estrangeira: Deve ser garantido que tuplas as quais serão eliminadas na relação que contém a chave primária não existam na relação que contém a chave estrangeira. No caso de exemplo da Figura 19, não seria possível eliminar um Curso caso ainda existam Estudantes relacionados a ele.

Heuser (2001, p. 74) ainda alerta que o termo estrangeira pode ser enganoso, tendo em vista que chaves estrangeiras podem referenciar a chave primária da própria relação.

2.3.2.3 Chave alternativa

Em certos casos, mais de um atributo ou conjunto de atributos podem diferenciar as tuplas de uma relação além da chave primária, neste caso, diz-se que estes outros atributos fazem parte

da chave estrangeira da relação. Tem-se como exemplo a Figura 20, onde um Empregado, além de seu código, também possui o seu CPF como chave alternativa, tendo em vista que ele é único na relação. (HEUSER, 2001, p. 75)

Figura 20: A chave alternativa CPF na relação Empregado

Empregado		
CodigoEmpregado	Nome	CPF
1	João da Silva	68471943409
5	Maria Pereira	17572734766
22	Leonardo Souza	29478359215

Fonte: Adptado de Heuser (2001, p. 75)

2.3.3 Restrições de integridade

Segundo Elmasri e Navathe (2010, p. 68), em um banco de dados relacional, é comum existirem diversas restrições para as relações e suas tuplas. Tais restrições são derivadas do cenário do mundo real onde o banco de dados será aplicado. Pode-se dividi-las em três categorias:

- Restrições implícitas, as quais são inerentes ao modelo de dados;
- Restrições explícitas, as quais são expressas diretamente no esquema do modelo de dados;
- Restrições de regra de negócio, as quais não podem ser diretamente expressas no modelo de banco de dados, e devem ser feitas pela aplicação que está utilizando o banco de dados.

2.3.4 Especificação de um banco de dados relacional

Silberschatz, Korth e Sudarshan (2006, p. 63) afirmam que é necessário diferenciar o esquema do banco de dados e uma instância de banco de dados, que é o estado do banco de dados em determinado momento.

O esquema de banco de dados é uma coleção de relações (ELMASRI; NAVATHE, 2010, p. 70). Silberschatz, Korth e Sudarshan (2006) apontam que existe uma convenção para nomear as relações e seus atributos, utilizando o nome da relação com a primeira letra maiúscula, seguido de um sinal de igualdade e a lista dos atributos entre parênteses, como apresentado na Sintaxe 1:

Sintaxe 1: Convenções para nomear relações - exemplo Estudante

Estudante = (Numero_estudante, Nome, Telefone, CodigoCurso)

Fonte: Elaborado pelo autor

Heuser (2001, p. 77) amplia o conceito, e indica que o esquema de banco de dados também deve evidenciar restrições de integridade. A sintaxe é modificada: remove-se o sinal de igualdade, sublinha-se atributos de chave primária, e indica-se restrições de chave estrangeira, conforme o exemplo da Sintaxe 2:

Sintaxe 2: Convenções para nomear relações - exemplo Curso

Estudante(Numero_estudante, Nome, Telefone, CodigoCurso)
CodigoCurso referencia Curso
Curso(CodigoCurso, Nome)

Fonte: Elaborado pelo autor

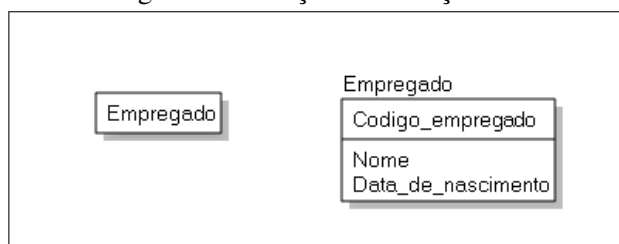
2.3.5 Notações para a modelagem relacional

Existem diversos modelos que podem ser utilizados para a modelagem lógica. Entre os mais populares, encontram-se o Information Engineering (IE), popularmente conhecido por notação "pé-de-galinha", e o Integration DEFinition for Information Modeling (IDEF1X) (TEOREY et al., 2010, p. 20), os quais se tornaram populares devido ao uso de ferramentas CASE. (EL-MASRI; NAVATHE, 2010, p. 345). Este estudo terá como foco o modelo IE.

O modelo Information Engineering, originalmente foi desenvolvido por Martin e Finkelstein (1981) e aprimorado por Martin e McClure (1985). (HAY, 1999).

Halpin e Morgan (2008) explicam a notação IE, onde relações são representadas por retângulos, e seus atributos aparecem em um compartimento abaixo, o qual é dividido entre atributos de chave primária na parte de cima e atributos comuns na parte de baixo, conforme mostra a Figura 21.

Figura 21: Relações na notação IE



Fonte: Adaptado de Halpin e Morgan (2008, p. 318)

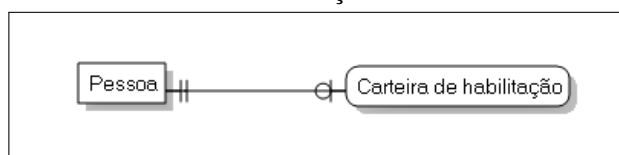
Relacionamentos são representados por linhas conectando as relações, com seu nome escrito próximo às linhas. O modelo IE apenas considera relacionamentos binários.

A cardinalidade e opcionalidade de um relacionamento são indicadas ao final da linha. Representa-se a opcionalidade do papel de uma relação em um relacionamento utilizando um círculo, ao fim do outro lado da linha, expressando assim cardinalidade mínima 0. Para simbolizar a obrigatoriedade, utiliza-se uma pequena linha reta e é colocada no final da linha, expressando assim, cardinalidade mínima 1.

Para indicar a cardinalidade máxima 1, também utiliza-se uma pequena linha reta. Desta forma a combinação do círculo com uma linha reta significa "pelo menos um", enquanto a combinação de duas linhas retas expressa "exatamente um", entretanto, algumas variações da notação utilizam apenas uma reta para expressar "exatamente um".

A Figura 22 ilustra esses conceitos, onde uma Pessoa pode ter ou não uma Carteira de habilitação, e uma Carteira de habilitação pertence a uma e somente uma Pessoa.

Figura 22: Relacionamento entre as relações Pessoa e Carteira de habilitação

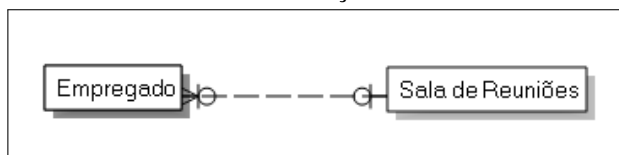


Fonte: Elaborado pelo autor

Para designar uma cardinalidade máxima n, utiliza-se três linhas em formato de "pé-de-galinha", conforme ilustra a Figura 23, onde uma Sala de Reuniões pode ser ocupada por nenhum ou vários Empregados, e Empregados podem estar ou não em uma Sala de Reuniões.

Relacionamentos identificadores são representados por uma linha contínua, enquanto relacionamentos não-identificadores são representados por uma linha pontilhada, ainda, relacionamentos não-identificadores opcionais possuem um losango.

Figura 23: Relacionamento entre as relações Sala de Reuniões e Empregado

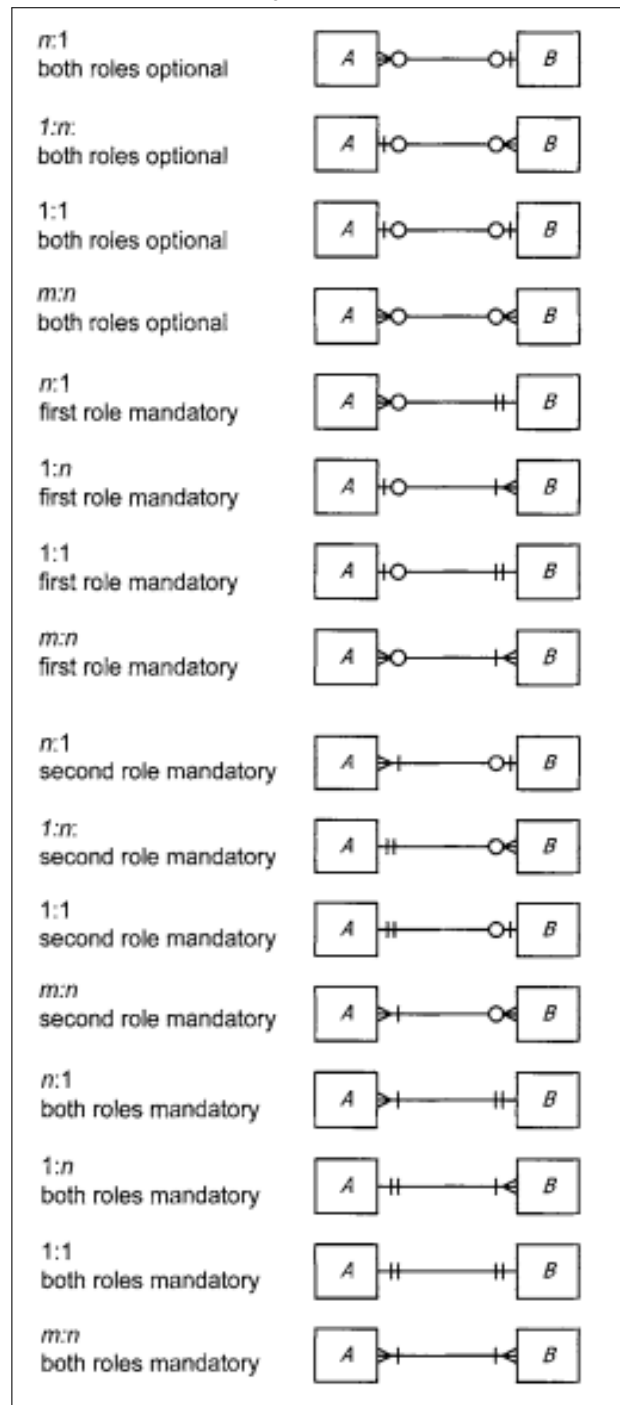


Fonte: Elaborado pelo autor

Observa-se também que como os dois lados da relação se mostram opcionais, o relacionamento é não-identificador. Algumas representações utilizam uma linha pontilhada para representar tal característica.

Desta forma, diversas possíveis combinações podem ser feitas, como mostra a Figura 24.

Figura 24: Possíveis combinações de relacionamento na notação IE

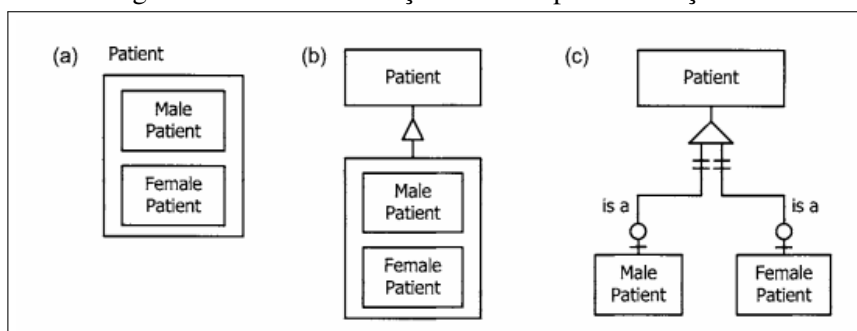


Fonte: Halpin e Morgan (2008, p. 322)

2.3.5.1 Subtipos

Existem diversas formas de representar a herança ou subtipagem utilizando a notação IE, tendo em vista que não há definição formal a ser utilizada, todavia, a Figura 25 apresenta possíveis notações.

Figura 25: Possíveis notações de subtipos na notação IE

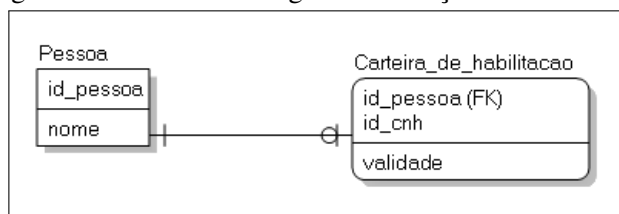


Fonte: Halpin e Morgan (2008, p. 322)

2.3.5.2 Chave estrangeira

Na notação IDEF1X, quando um atributo referencia uma chave estrangeira, o mesmo é indicado com um símbolo de "(FK)" ao seu lado, sinalizando assim uma *Foreign Key* (FK), isto é, uma chave estrangeira. (SONG; EVANS; PARK, 1995). Algumas ferramentas CASE, como o ERWin (SYBASE, 2013), utilizam de propriedades da notação IDEF1X para expressar a chave estrangeira na notação IE, conforme ilustra a Figura 26.

Figura 26: A chave estrangeira na notação IE no ERWin



Fonte: Elaborado pelo autor

2.4 O modelo físico

O modelo físico visa definir especificações para o armazenamento e acesso ao banco de dados, utilizando estratégias como índices, clusters e domínio específico do atributo. (HALPIN; MORGAN, 2008, p. 32). Neste passo, o modelo lógico é transcrito para a SQL, a qual é detalhada na seção 2.7. Com isso, é necessária a definição do domínio de uma coluna, independente para cada SGBD. A definição de domínio é descrita na seção 2.3.1. (TEOREY et al., 2010, p. 6).

Segundo Teorey et al. (2010, p. 190), a modelagem conceitual mapeia as regras de negócio e os requisitos funcionais do banco de dados; o modelo lógico define as entidades e relacionamentos, aplicando normalização; e o modelo físico transforma o modelo lógico em objetos do banco de dados, isto é, tabelas, índices e restrições de integridade.

2.5 Transformação entre modelos

Silberschatz, Korth e Sudarshan (2006, p. 51) apontam que tanto o modelo ER quanto o modelo relacional são abstratos, nesse sentido, é possível converter o modelo ER em um modelo relacional.

Elmasri e Navathe (2010, p. 287) definem uma série de passos para a transformação do modelo ER para o modelo relacional:

1. **mapeamento de entidades fortes:** para cada entidade forte E , criar uma relação R que contenha todos atributos simples de E . Para os atributos compostos, apenas utilizar os atributos simples que os compõe. Atributos identificadores da entidade forte E se tornarão atributos que fazem parte da chave primária na relação R .
2. **mapeamento de entidades fracas:** para cada entidade fraca W que pertença a uma entidade forte E , criar uma relação R que contenha todos os atributos simples e atributos simples que compõe os atributos compostos da entidade fraca W .

Além disso, incluir como chave estrangeira em R , todos os atributos identificadores da entidade forte E . A chave primária da entidade fraca W será composta pelos seus atributos identificadores juntamente com os atributos identificadores da entidade forte E .

Se existe uma entidade E_2 a qual sua entidade forte também seja uma entidade fraca E_1 , então mapear primeiramente a chave primária da entidade E_1 .

3. **mapeamento de relacionamentos binário 1:1:** Para cada relacionamento binário 1:1 R , identifica-se as entidades participantes S e T que a compõe. Existem três possíveis abordagens para esse mapeamento: (i) uso de chave estrangeira (ou adição de coluna); (ii) fusão de relações e; (iii) referência cruzada (ou tabela própria).
 - (a) **uso de chave estrangeira:** escolha uma das relações, por exemplo, S , e inclua a chave primária de T como chave estrangeira em S . Idealmente utiliza-se uma entidade que tenha participação total de R no papel de S , isto é, que todo S contenha um T .
 - (b) **fusão de relações:** é possível fundir duas entidades em uma única relação R , tal abordagem é indicada quando ambas entidades possuem participação total no relacionamento.
 - (c) **referência cruzada:** cria-se uma terceira relação R , a qual serve como uma referência cruzada entre as chaves primárias da relação S e T . Tal abordagem também é chamada de tabela própria (HEUSER, 2001, p. 90), pois cada tupla em R se relaciona a uma tupla de S e T . A relação R irá possuir como chave estrangeira as chaves primárias de S e T . A chave primária de R será ou a chave primária de S ou a chave primária de T .

4. **mapeamento de relacionamentos binário 1:N:** para cada relacionamento binário 1:N R, identifica-se a entidade S que faz parte do lado "n"(muitos) do relacionamento. Como cada tupla de S está relacionada a pelo menos uma tupla de T, a chave primária de T se torna chave estrangeira em S.

Como uma abordagem alternativa, também é possível utilizar a referência cruzada utilizada em relacionamentos 1:1, porém, este tipo de abordagem é evitada devido ao número de excessivo de valores nulos que causa.

5. **mapeamento de relacionamentos binário N:N:** para cada relacionamento binário N:N R, cria-se uma nova relação S para representar R.

A chave estrangeira de S será as chaves primárias das relações que estão participando do relacionamento, esse conjunto formará também a chave primária de S. Também incluem-se os atributos simples ou atributos simples de um atributo composto do relacionamento R como atributos da relação S.

6. **mapeamento de relacionamentos com grau maior que 2:** para cada relacionamento com grau maior que 2 R, cria-se uma nova relação S para representar R.

Inclue-se como chave estrangeira de S as chaves primárias das relações que estão participando do relacionamento. Também incluem-se os atributos simples ou atributos simples de um atributo composto do relacionamento R como atributos da relação S.

A chave primária de S será a combinação de todas as chaves estrangeiras. Contudo, caso caso a cardinalidade de alguma entidade E do relacionamento R seja 1, então a chave primária de S não deve incluir a chave estrangeira de E.

7. **mapeamento de atributos multivalorados:** para cada atributo multivalorado A, cria-se uma nova relação R. A relação R irá conter um atributo correspondente a A, assim como um atributo K, que é chave estrangeira em R, da relação ou relacionamento que contém o atributo multivalorado A. A chave primária de R será a combinação de A e K.

8. **mapeamento especializações/generalizações:** (HEUSER, 2001) cita que existem duas possíveis alternativas para mapear especializações e generalizações: (i) uso de uma relação para cada entidade e; (ii) uso de uma única relação para toda hierarquia de generalização/especialização.

- (a) **uma relação por hierarquia:** Nesta alternativa, todas as relações as quais fazem parte da especialização/generalização serão fundidas em uma única relação, sendo que a chave primária corresponderá à chave primária da relação mais genérica. Haverá também colunas referentes a cada atributo da entidade genérica e da(s) entidade(s) especializadas, assim como atributos referentes aos relacionamentos, que devem ser utilizados utilizando a técnica de adição de coluna.

- (b) **uma relação por entidade especializada:** Nesta outra alternativa, cria-se uma relação para cada generalização/especialização, aplicando as regras correspondentes apresentadas anteriormente. Porém, deve ser incluída nas entidades especializadas a chave primária da entidade genérica.

(HEUSER, 2001) apresenta uma visão geral das regras utilizadas para relacionamentos binários, conforme ilustra a Figura 27.

Figura 27: Regras para a transformação de relacionamentos binários do modelo ER para o modelo relacional

Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição coluna	Fusão tabelas
Relacionamentos 1:1			
	±	✓	×
	×	±	✓
	×	±	✓
Relacionamentos 1:n			
	±	✓	×
	±	✓	×
	×	✓	×
	×	✓	×
Relacionamentos n:n			
	✓	×	×
	✓	×	×
	✓	×	×
✓ Alternativa preferida ± Pode ser usada × Não usar			

Sendo assim, caso se deseja realizar a transformação do modelo relacional para o modelo ER, pode-se utilizar técnicas de engenharia reversa utilizando estes passos.

2.6 Álgebra relacional

Um modelo de dados, além de conter as definições da estrutura e restrições dos dados, também deve incluir um conjunto de operações para manipular o banco de dados. No modelo relacional, esse conjunto de operações é expresso a partir da álgebra relacional.

A partir de expressões escritas em álgebra relacional, é possível descrever as operações com o intuito de realizar consultas no banco de dados. O resultado das consultas é uma nova relação, a qual pode ser manipulada utilizando novamente operações em álgebra relacional. Uma sequência de operações forma uma expressão em álgebra relacional.

A álgebra relacional fornece uma fundamentação formal para as operações do modelo relacional, assim como seus conceitos são utilizados como base para o SQL, no que se deve a sua implementação e otimização no processo de consulta ao banco de dados. (ELMASRI; NAVATHE, 2010, p. 145)

As operações em álgebra relacional podem ser classificadas e listadas como (ELMASRI; NAVATHE, 2010; SILBERSCHATZ; KORTH; SUDARSHAN, 2006; MÜLLER, 2011b):

- fundamentais;
 - seleção;
 - projeção;
 - união;
 - diferença;
 - interseção;
 - produto cartesiano;
- acessórias / utilitárias;
 - renomeação;
 - designação;
- adicionais;
 - junção;
 - divisão;
- estendidas;
 - junções externas;

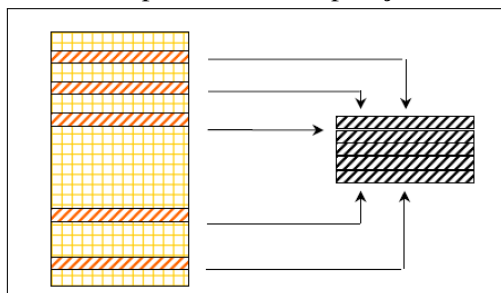
- projeção generalizada;
- funções agregadas;
- modificadores;
 - inclusão;
 - alteração;
 - exclusão.

2.6.1 Seleção

Silberschatz, Korth e Sudarshan (2006, p. 69) definem que a "operação de seleção seleciona tuplas que satisfaçam um determinado predicado". Utiliza-se a letra grega minúscula sigma (σ) como notação para a seleção, o predicado aparece subscrito a sigma, com o argumento da relação provido entre parênteses, após o sigma.

A Figura 28 mostra o comportamento da operação de seleção.

Figura 28: Comportamento da operação de seleção



Fonte: Müller (2011b)

Para selecionar as tuplas da relação *Empregado*, cujo Departamento é o de compras, escreve-se como é apresentado na Sintaxe 3:

Sintaxe 3: Operação de seleção

$$\sigma_{\text{departamento}=\text{"compras"}}(\textit{Empregado})$$

Fonte: Elaborado pelo autor

Também é possível utilizar no predicado comparações do tipo igualdade(=), desigualdade(\neq), menor que (<), maior que (>), menor ou igual que (\leq), maior ou igual que (\geq), assim como os conectivos 'e' (\wedge) e 'ou' (\vee). O exemplo da Sintaxe 4 ilustra seu uso, onde se seleciona tuplas da relação *Empregados*, as quais sejam do departamento de compras e também tenham idade maior que 35 anos.

Sintaxe 4: Operação de seleção com comparações

$$\sigma_{\text{departamento}=\text{"compras"} \wedge \text{idade} > 35}(\text{Empregado})$$

Fonte: Elaborado pelo autor

A Figura 29 mostra o resultado da operação de seleção sobre uma relação R.

Figura 29: Resultado da operação de seleção

Relação r

r			
A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$\sigma_{A=B \wedge D > 5}(r)$

$\sigma_{A=B \wedge D > 5}(r)$			
A	B	C	D
α	α	1	7
β	β	23	10

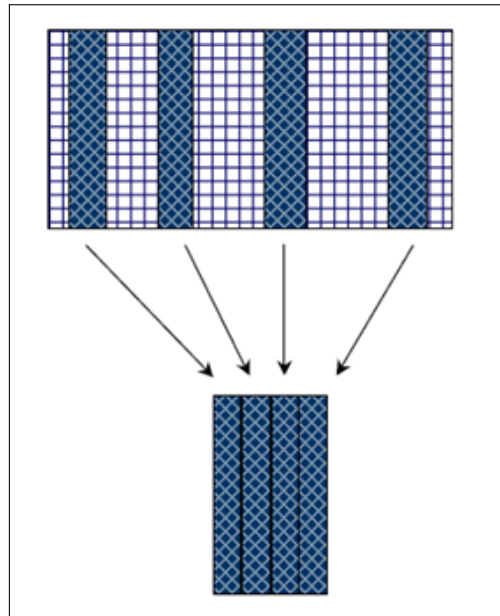
Fonte: Müller (2011b)

2.6.2 Projeção

A operação de projeção permite filtrar quais atributos de uma dada relação devem ser exibidos. Como a relação resultante é um conjunto, tuplas repetidas serão eliminadas. A projeção é denotada pela letra grega pi maiúscula (Π), os atributos desejados são subscritos em Π . (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 70).

A Figura 30 mostra o funcionamento da operação de projeção.

Figura 30: Funcionamento da operação de projeção



Fonte: Müller (2011b)

Para exemplificar, poderia-se utilizar a operação de projeção para apenas exibir o Nome e Idade de um Empregado, como segue na Sintaxe 5:

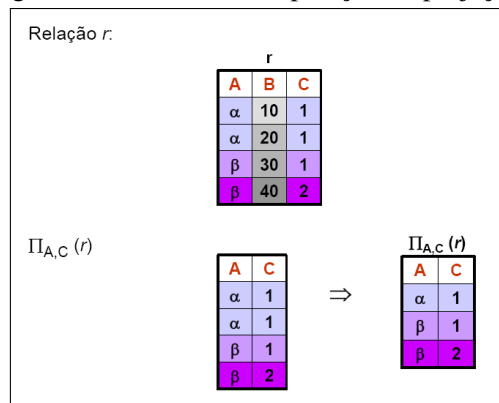
Sintaxe 5: Operação de projeção

$$\Pi_{nome, idade}(Empregado)$$

Fonte: Elaborado pelo autor

A Figura 31 ilustra o resultado da operação de projeção sobre uma relação R.

Figura 31: Resultado da operação de projeção



Fonte: Müller (2011b)

2.6.3 A operação relacional composta

Como o resultado de uma operação relacional é uma relação, é possível combinar operações em álgebra relacional para formar uma expressão em álgebra relacional, contanto que o resultado de uma operação seja do mesmo tipo que sua entrada. No exemplo a seguir, cria-se uma expressão a qual utiliza os operadores de seleção e projeção, consultando o Nome dos Empregados que tem menos de 60 anos, apresentada pela Sintaxe 6:

Sintaxe 6: Operação relacional composta

$$\Pi_{nome}(\sigma_{idade < 60}(Empregado))$$

Fonte: Elaborado pelo autor

2.6.4 União

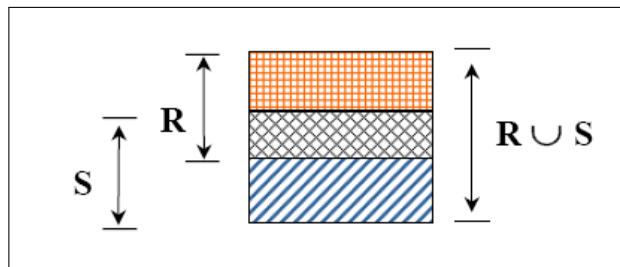
A operação de união, denotada por $R \cup S$, resulta em uma relação que contém todas as tuplas que estão tanto em R quanto em S. Duplicatas são eliminadas. (ELMASRI; NAVATHE, 2010, p. 153).

Porém, conforme Silberschatz, Korth e Sudarshan (2006, p. 72) apontam, para uma operação de união $R \cup S$ ser válida, são necessárias duas condições:

1. as relações R e S devem possuir o mesmo número de atributos;
2. os domínios do i-ésimo atributo de R e o i-ésimo atributo de S devem ser os mesmos, para todo i.

A Figura 32 mostra o funcionamento da operação de união.

Figura 32: Funcionamento da operação de união



Fonte: Müller (2011b)

Como exemplo, mostrado pela Sintaxe 7 buscaram-se as Matrículas dos Empregados que trabalham no Departamento de compras e de Gerentes de Empregados que trabalham no Departamento de vendas.

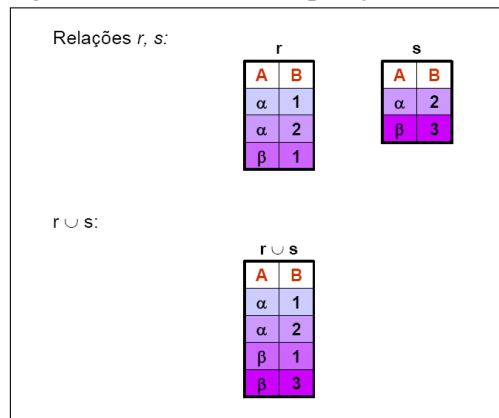
Sintaxe 7: Operação de união

$$\Pi_{matricula}(\sigma_{departamento="compras"}(Empregado)) \cup \Pi_{gerente}(\sigma_{departamento="vendas"}(Empregado))$$

Fonte: Elaborado pelo autor

A Figura 33 apresenta o resultado da operação de união sobre uma relação R.

Figura 33: Resultado da operação de união



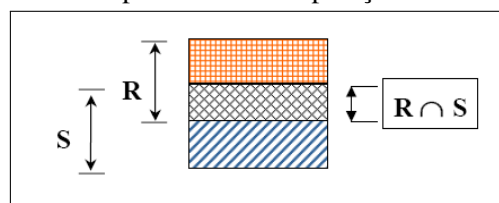
Fonte: Müller (2011b)

2.6.5 Intersecção

A operação de intersecção, denotada por $R \cap S$, resulta em uma relação que inclui todas as tuplas que estão ambas em R e S. (ELMASRI; NAVATHE, 2010, p. 153).

A Figura 34 apresenta o comportamento da operação de intersecção.

Figura 34: Comportamento da operação de intersecção



Fonte: Müller (2011b)

Exemplifica-se a operação de intersecção tomando como exemplo a necessidade de consultar os Empregados do setor de compras e que são Gerentes, como ilustra a Sintaxe 8:

Sintaxe 8: Operação relacional de intersecção

$$\Pi_{matricula}(\sigma_{departamento='compras'}(Empregado)) \\ \cap \\ \Pi_{gerente}(Emprgeneregado))$$

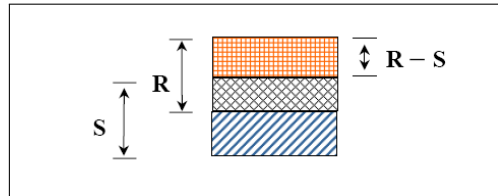
Fonte: Elaborado pelo autor

2.6.6 Diferença

A operação de diferença, denotada por $R - S$, resulta em uma relação que inclui todas as tuplas que estão em R mas não estão em S. (ELMASRI; NAVATHE, 2010, p. 153).

A Figura 35 ilustra o funcionamento da operação de diferença.

Figura 35: Funcionamento da operação de diferença



Fonte: Müller (2011b)

Como exemplo apresentado na Sintaxe 9, supõe-se uma operação de intersecção que visa consultar os Empregados do setor de compras e que não são Gerentes:

Sintaxe 9: Operação de diferença

$$\Pi_{matricula}(\sigma_{departamento='compras'}(Empregado)) \\ - \\ \Pi_{gerente}(Empregado))$$

Fonte: Elaborado pelo autor

A Figura 36 mostra o resultado da operação de diferença.

Figura 36: Resultado da operação de diferença

Relações r, s :

r		s	
A	B	A	B
α	1	α	2
α	2	β	3
β	1		

$r - s$:

$r - s$	
A	B
α	1
β	1

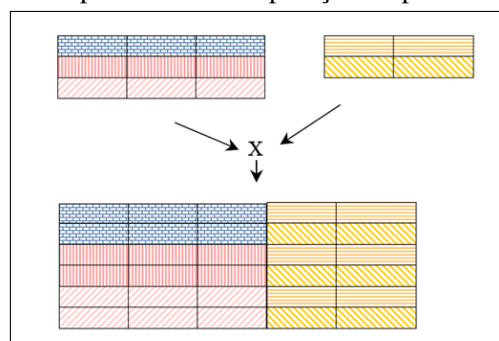
Fonte: Müller (2011b)

2.6.7 Produto cartesiano

Silberschatz, Korth e Sudarshan (2006, p. 73) explicam que a operação de produto cartesiano, representada por $R_1 \times R_2$, permite combinar informações entre duas relações. Caso o mesmo nome de atributo apareça nas duas relações, anexa-se ao nome de atributo a relação a qual esse atributo se origina. Por exemplo, o esquema da relação para $R = Empregado \times Departamento$ seria: *(empregado.nome, empregado.idade, empregado.codDepartamento, departamento.codDepartamento, departamento.numprojetos)*.

A relação R resultante da operação do produto cartesiano entre $R_1 \times R_2$ irá possuir cada tupla de R_1 combinada com cada tupla de R_2 , como ilustra a Figura 37.

Figura 37: Comportamento da operação de produto cartesiano



Fonte: Müller (2011b)

Tendo n_1 tuplas em R_1 e n_2 tuplas em R_2 , então existem $n_1 * n_2$ modos de escolher um par de tuplas, uma tupla para cada relação; deste modo, existem $n_1 * n_2$ tuplas em R .

Como exemplo mostrado na Sintaxe 10, obtém-se o nome de cada Projeto que Empregados do Departamento "pesquisa" estão associados.

Sintaxe 10: Operação relacional de produto cartesiano

$$\Pi_{projeto.nome}(\sigma_{empregado.codprojeto=projeto.codprojeto}(\sigma_{empregado.depto="pesquisa"}(Empregado \times Projeto)))$$

Fonte: Elaborado pelo autor

A Figura 38 apresenta o resultado da operação de produto cartesiano.

Figura 38: Resultado da operação de produto cartesiano

Relações r e s:

r	
A	B
α	1
β	2

s		
C	D	E
α	10	a
α	13	a
β	20	b
γ	10	b

r x s:

A	B	C	D	E
α	1	α	10	a
α	1	α	13	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	α	13	a
β	2	β	20	b
β	2	γ	10	b

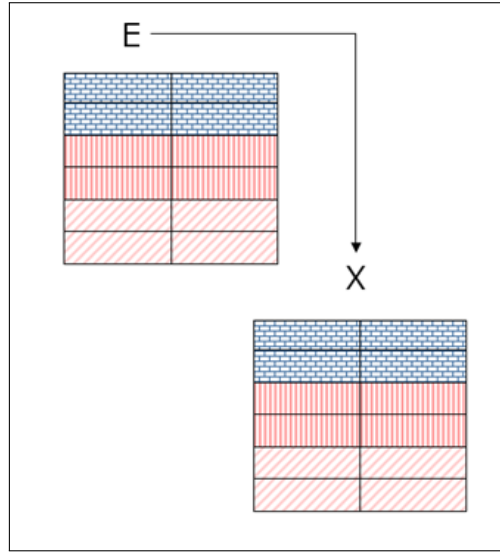
Fonte: Müller (2011b)

2.6.8 Renomeação

O resultado de uma expressão em álgebra relacional não possui um nome que possa ser utilizado como referência. A relação de rename, representada pela letra minúscula grega rho(ρ), permite que tal tarefa seja permitida. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 76)

A Figura 39 ilustra o comportamento da operação de renomeação.

Figura 39: Funcionamento da operação de renomeação



Fonte: Müller (2011b)

Dada uma expressão álgebra relacional E , a expressão $\rho_x(E)$ tem por resultado a expressão E sob o nome x . Também é possível utilizar a operação de rename sobre os atributos de uma relação, seguindo a seguinte estrutura: $\rho_{x(A_1, A_2, \dots, A_n)}(E)$

Como exemplo ilustrado na Sintaxe 11, renomeia-se o atributo matrícula para identificador:

Sintaxe 11: Renomeação

$$\rho_{identificador}(\Pi_{matricula}(Empregado))$$

Fonte: Elaborado pelo autor

2.6.9 Designação

Certas vezes é conveniente utilizar a relação resultante de uma expressão em álgebra relacional como uma variável temporária. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 83). Esta operação, denotada por \leftarrow , é escrita como mostrado na Sintaxe 12:

Sintaxe 12: Designação

$$temp \leftarrow \sigma_{idade > 20}(Empregado)$$

Fonte: Elaborado pelo autor

2.6.10 Junção natural

A operação de junção natural, representada por \bowtie , é utilizada para combinar tuplas relacionadas de duas relações. Ela permite a utilização de relacionamentos em um banco de dados relacional. (ELMASRI; NAVATHE, 2010, p. 157).

A operação de junção natural combina seleções e um produto cartesiano dentro de uma operação: elas formam um produto cartesiano de seus dois argumentos, fazendo uma seleção de seus atributos em equivalência e remove os atributos em duplicidade. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 79).

Como exemplo, Silberschatz, Korth e Sudarshan (2006, p. 79) citam uma consulta para "encontrar todos os nomes dos clientes que tenham um empréstimo no banco e encontrar o total emprestado", primeiramente, exibe-se a partir da Sintaxe 13 a consulta utilizando uma expressão que utiliza a operação de produto cartesiano:

Sintaxe 13: Operação com produto cartesiano para exemplificar junção natural

$$\Pi_{\text{nomecliente, emprestimo.numemprestimo, total}}(\sigma_{\text{devedor.numemprestimo=emprestimo.numemprestimo}}(\text{Devedor} \times \text{Emprestimo}))$$

Fonte: Elaborado pelo autor

Utilizando o mesmo exemplo da consulta anterior, é apresentada na Sintaxe 14 uma expressão que contém a operação de junção natural:

Sintaxe 14: Junção natural

$$\Pi_{\text{nomecliente, numemprestimo, total}}(\text{Devedor} \bowtie \text{Emprestimo})$$

Fonte: Elaborado pelo autor

2.6.11 Junção externa

Silberschatz, Korth e Sudarshan (2006, p. 92) explicam que a junção externa é "uma extensão da operação de junção para tratar informações omitidas". A Figura 40 mostra as relações Empregado (emp) e Empregado Tempo Integral (emp_ti).

Figura 40: Relações Empregado e Empregado Tempo Integral

EMP		
Nome	Rua	Cidade
João	Afonso Pena	Rio de Janeiro
Saul	Teresa	Petrópolis
Hiran	Pedro Ernesto	Niterói
Marisa	Lopes Quintas	Rio de Janeiro

EMP_TI	
Nome	Salário
João	5300
Saul	1600
Marisa	4000
Josefa	2500

Fonte: Müller (2011b)

Supõe-se uma consulta a qual se deseja descobrir todas as informações dos empregados em tempo integral. Para isso, poder-se-ia utilizar de uma junção natural, porém, como exibido na Figura 41, perde-se informações de Empregados que não trabalham em tempo integral.

Figura 41: Relações Empregado e Empregado Tempo Integral

EMP ⋈ EMP_TI	Nome	Rua	Cidade	Salário
	João	Afonso Pena	Rio de Janeiro	5300
	Saul	Teresa	Petrópolis	1600
	Marisa	Lopes Quintas	Rio de Janeiro	4000

Fonte: Müller (2011b)

Para evitar a perda de informações, pode-se utilizar a junção externa. Existem três formas de usar essa operação:

1. junção externa à esquerda, detonada por $\bowtie\leftarrow$;
2. junção externa à direita, detonada por $\rightarrow\bowtie$;
3. junção externa total, detonada por \bowtie .

A junção externa à esquerda utiliza todas as tuplas da relação à esquerda que não formam par com tuplas da relação à direita, preenchendo com valores nulos os atributos da relação à direita, como mostra a Figura 42.


Figura 42: Junção externa à esquerda

EMP ⋈ _l EMP_TI	Nome	Rua	Cidade	Salário
	João	Afonso Pena	Rio de Janeiro	5300
	Saul	Teresa	Petrópolis	1600
	Marisa	Lopes Quintas	Rio de Janeiro	4000
	Hiran	Pedro Ernesto	Niterói	nulo

Fonte: Müller (2011b)

A junção externa à direita é simétrica à junção externa à esquerda: as tuplas da relação à direita que não formam par com tuplas da relação à esquerda, e preenche-se com valores nulos os atributos da relação à esquerda, como mostra a Figura 43.


Figura 43: Junção externa à direita

	Nome	Rua	Cidade	Salário
	João	Afonso Pena	Rio de Janeiro	5300
	Saul	Teresa	Petrópolis	1600
	Marisa	Lopes Quintas	Rio de Janeiro	4000
	Josefa	Nulo	Nulo	2500

Fonte: Müller (2011b)

A junção externa total aplica ambas as operações de junção externa à direita e junção externa à esquerda. Assim, preenchendo tanto as relações da esquerda que não encontram um par à direita, assim como da direita que não encontram um par à esquerda, como aponta a Figura 44.

Figura 44: Junção externa total

	Nome	Rua	Cidade	Salário
	João	Afonso Pena	Rio de Janeiro	5300
	Saul	Teresa	Petrópolis	1600
	Marisa	Lopes Quintas	Rio de Janeiro	4000
	Hiran	Pedro Ernesto	Niterói	nulo
	Josefa	Nulo	Nulo	2500

Fonte: Müller (2011b)

2.6.12 Divisão

A operação de divisão, denotada por \div , pode ser utilizada em consultas que empregam as frase 'para todos'. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 81).

Como exemplo, Silberschatz, Korth e Sudarshan (2006, p. 81) citam o desejo de "encontrar todos os clientes que tenham conta em todas as agências localizadas no Brooklyn". Para obter as agências no Brooklyn, utiliza-se a expressão apresentada na Sintaxe 15 :

Sintaxe 15: Obtendo agências do Brooklyn

Fonte: Elaborado pelo autor

Para encontrar os pares de clientes que tem uma conta em uma agência, mostrado na Sintaxe 16:

Sintaxe 16: Obtendo clientes

$$\Pi_{nomecliente, nomeagencia}(Depositante \bowtie Conta)$$

Fonte: Elaborado pelo autor

Agora, pode-se utilizar a operação de divisão para encontrar os clientes que aparecem em r_2 , com todos os nomes de agência em r_1 , sendo assim, a expressão poderia ser escrita da seguinte forma apresentada na Sintaxe 17:

Sintaxe 17: Operação de divisão

$$\frac{\Pi_{nomecliente, nomeagencia}(Depositante \bowtie Conta)}{\Pi_{nomeagencia}(\sigma_{cidadeagencia="Brooklyn"}(Agencia))}$$

Fonte: Elaborado pelo autor

Formalmente, diz-se que a operação de divisão se aplica a duas relações $R(Z) \div S(X)$, onde os atributos de R são um subconjunto dos atributos de S , isto é, $X \subseteq Z$, e tem como resultado a relação $T(Y)$, a qual inclui uma tupla t se as tuplas t_R que aparecerem em R com $t_R[Y] = t$, e com $t_R[X] = t_s$ para toda tupla t_s em S . Isto é, para uma tupla t aparecer no resultado T da operação de divisão, os valores em t devem aparecer em R em combinação com cada tupla de S . (ELMASRI; NAVATHE, 2010, p. 162).

A Figura 45 mostra o comportamento da operação de divisão.

Figura 45: Comportamento da operação de divisão

S		R	
A1		A1	A2
a1		a1	b1
a2		a2	b1
a3		a3	b1
		a4	b1
		a1	b2
		a3	b2
		a2	b3
		a3	b3
		a4	b3
		a1	b4
		a2	b4
		a3	b4

R ÷ S	
A2	
b1	
b4	

Fonte: Müller (2011b)

2.6.13 Projeção generalizada

A projeção generaliza permite que uma operação de projeção envolva funções aritméticas e valores constantes em seus atributos. (ELMASRI; NAVATHE, 2010, p. 166). O exemplo da Sintaxe 18 apresenta a obtenção do valor do Salário de um Empregado descontando as Deduções:

Sintaxe 18: Operação de projeção generalizada

$\Pi_{salario-deduccoes}(Empregado)$

Fonte: Elaborado pelo autor

2.6.14 Funções agregadas

Em certos cenários, é interessante a utilização de funções matemática de agregação, como por exemplo, a soma do Salário de todos os Empregados. (ELMASRI; NAVATHE, 2010, p. 166).

Silberschatz, Korth e Sudarshan (2006) elencam e explicam as funções agregadas:

- **sum**: retorna a soma de uma coleção de valores;
- **avg**: retorna a média de dos valores;
- **count**: retorna o número de elementos da coleção ;
- **min**: retorna o valor mínimo de uma coleção de valores;
- **max**: retorna o valor máximo de uma coleção de valores;
- **count**: retorna o numero de elementos em uma coleção ;
- **count-distinct**: retorna o numero de elementos distintos em uma coleção.

Expressões agregadas são utilizadas da seguinte forma:

$\langle \text{nome da função} \rangle_{\text{atributos}}(\text{Relação})$

Os exemplos da Sintaxe 19 mostra a obtenção da média salarial dos Empregados e o Empregado mais velho:

Sintaxe 19: Funções agregadas

$$\begin{array}{l} avg_{salario}(Empregado) \\ max_{idade}(Empregado) \end{array}$$

Fonte: Elaborado pelo autor

2.6.15 Inclusão

Pode-se inserir tuplas ou resultados de consultas em uma relação, contanto que os atributos das tuplas a serem inseridas sejam do mesmo domínio dos atributos da relação. Na álgebra relacional, uma inserção é expressa como $R \leftarrow R \cup E$, onde R é uma relação e E, uma expressão em álgebra relacional. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 98).

No exemplo ilustrado na Sintaxe 20 , insere-se as informação de que Maria Silva tem 2000 reais na conta 25892.

Sintaxe 20: Operação de inclusão

$$conta \leftarrow conta \cup ("Maria Silva", 2000, 25892)$$

Fonte: Elaborado pelo autor

2.6.16 Atualização

Silberschatz, Korth e Sudarshan (2006, p. 98) indicam que é possível alterar o valor de uma tupla sem mudar todos os seus valores, utiliza-se um operador de projeção para esta tarefa, como apresentado na Sintaxe 21

Sintaxe 21: Operador de atualização

$$R \leftarrow \Pi_{x(A_1, A_2, \dots, A_n)}(R)$$

Fonte: Elaborado pelo autor

Onde cada A_i é o i-ésimo atributo de R, se o i-ésimo atributo não for atualizado, ou, se o i-ésimo atributo for atualizado. A_i é uma expressão que apenas envolve constantes e atributos de R, definindo novos valores.

Ilustra-se o uso do operador de atualização na Sintaxe 22 , onde um Empregado ganha um aumento no Salário de 5

Sintaxe 22: Operação de atualização

$$Empregado \leftarrow \Pi_{nome, idade, salario, salario \leftarrow salario * 1.05}(Empregado)$$

Fonte: Elaborado pelo autor

2.6.17 Exclusão

Em álgebra relacional, é possível excluir tuplas inteiras utilizando uma expressão similar a consulta $R \leftarrow R - E$, onde R é uma relação E , uma consulta em álgebra relacional. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 97)

Como exemplo apresentado na Sintaxe 23, imagina-se uma empresa que esteja passando por dificuldades e decide demitir todos os funcionários que ganham mais de 5 mil reais.

Sintaxe 23: Operação de exclusão

$$Empregado \leftarrow Empregado - \sigma_{salario > 5000}(Empregado)$$

Fonte: Elaborado pelo autor

2.7 SQL

A seção 2.6 apresentou a álgebra relacional, que proporciona uma notação concisa para operações em banco de dados; porém, os usuários de um banco de dados necessitam de uma linguagem mais simples. Neste sentido, surgiu a linguagem comercial mais utilizada no mercado, a Structure Query Language (SQL). A SQL permite realizar consultas ao banco de dados, definir estruturas de dados, modificar dados no banco de dados e definir especificações de segurança. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 109).

Conforme Silberschatz, Korth e Sudarshan (2006, p. 110), a linguagem SQL possui diversas diretrizes:

- **Data Definition Language (DDL):** a linguagem de definição de dados permite definir, modificar e excluir relações, assim como a criação de índices;
- **Data Manipulation Language (DML):** a linguagem de manipulação de dados se baseia na álgebra relacional, e além de proporcionar consultas ao banco de dados, também provê formas de inserir, modificar e excluir tuplas em um banco de dados;
- **Embedded DML:** a incorporação de comandos SQL é utilizada em linguagens de programação de uso geral;

- **Visões:** a SQL permite definir visões, as quais são relações definidas a partir de consultas. (RAMAKRISHNAN; GEHRKE, 2003, p. 59);
- **Autorização:** é possível especificar direitos de acesso à relações e visões;
- **Integridade:** é possível especificar regras de integridade para os dados que serão armazenados no banco de dados;
- **Transações:** a SQL inclui comandos para controlar o começo e fim de transações.

A SQL utiliza termos diferenciados, os quais serão utilizados para esta seção: relações são tabelas, tuplas são linhas e atributos são colunas. Este estudo irá abordar somente os tópicos relacionados a DDL, DML e restrições de integridade.

2.7.1 Esquemas

Um esquema (schema) em SQL, agrupa tabelas e outros construtos que pertencem ao mesmo banco de dados. Um esquema é identificado por seu nome, um identificador de autorização, o qual identifica o usuário que é proprietário do esquema, assim como descritores para cada elemento no esquema.

Um esquema é definido a partir do comando *CREATE SCHEMA*, o qual pode incluir a definição de todos os elementos, ou, pode-se apenas definir o nome e a autorização, assim, definindo os elementos posteriormente. (ELMASRI; NAVATHE, 2010, p. 89). Como exemplo, cria-se o esquema *EMPRESA* com autorização para o usuário *SILVA*, conforme apresentado na Sintaxe 24.

Sintaxe 24: Definição de um esquema em SQL

<pre>CREATE SCHEMA EMPRESA AUTHORIZATION "SILVA" ;</pre>
--

Fonte: Elaborado pelo autor

2.7.2 Data Definition Language

O comando *CREATE TABLE* especifica uma nova tabela a partir de um nome, em conjunto com suas colunas e restrições de integridade. Primeiramente, declara-se o nome da tabela; após, para cada coluna, define-se um tipo de dados para especificar o domínio dos valores, assim como restrições de atributos como a nulidade; por fim, declara-se restrições de integridade envolvendo chaves. (ELMASRI; NAVATHE, 2010, p. 90).

A Sintaxe 25 mostra a criação da tabela *Empregado*, com os atributos de *IdEmpregado*, *Nome*, *Idade* e *Gênero*.

Sintaxe 25: Criação de uma tabela em SQL

```
CREATE TABLE empregado
(
    idempregado INT NOT NULL,
    name        VARCHAR(15) NOT NULL,
    idade       VARCHAR(15) NOT NULL,
    genero      VARCHAR(1) ,
    PRIMARY KEY(idempregado)
);
```

Fonte: Elaborado pelo autor

Considera-se que o comando que será declarado já está implicitamente especificado no esquema. Caso fosse necessário explicitar o esquema, seria utilizado a forma da Sintaxe 26.

Sintaxe 26: Criação de uma tabela em SQL denotando um esquema

```
CREATE TABLE empresa . empregado
(
    ...
);
```

Fonte: Elaborado pelo autor

2.7.2.1 Tipos de dados

Conforme Silberschatz, Korth e Sudarshan (2006, p. 138), o padrão SQL aceita uma variedade para tipos de domínios para atributos, sendo eles:

- **char(n)**: é uma cadeia de caracteres de tamanho n fixo;
- **varchar(n)**: é uma cadeia de caracteres com tamanho variável, com tamanho n máximo;
- **int**: é um inteiro;
- **smallint**: é um inteiro pequeno;
- **numeric(p,d)**: é um número de ponto fixo com precisão, onde p consiste no número dígitos, e d consiste nos pontos decimais;
- **real, double precision**: é um número de ponto flutuante;
- **float(n)**: é um número de ponto flutuante porém com precisão definida por n;
- **date**: é uma data que contém o ano (quatro dígitos), mês e dia do mês;
- **time**: representa o horário, em horas, minutos e segundos.

Sintaxe 27: Criação de um domínio em SQL

```
CREATE DOMAIN nome_pessoa CHAR(20) ;
```

Fonte: Silberschatz, Korth e Sudarshan (2006, p. 139)

Também é possível definir domínios utilizando a cláusula *CREATE DOMAIN*, como no exemplo da Sintaxe 27:

2.7.2.2 Restrições de integridade

Elmasri e Navathe (2010, p. 94) explicam que a SQL permite diversos tipos de restrições. Restrições de nulidade podem ser aplicadas às colunas de uma tabela, a qual é especificada a partir da cláusula *NOT NULL* ou *NULL*. Também é possível definir um valor *default* (padrão) para um atributo, através do acréscimo da cláusula *DEFAULT < valor >* na definição de um atributo. Além disso, é possível adicionar a cláusula *CHECK* com o intuito de restringir os valores de um atributo ou domínio.

Existem cláusulas especiais para a definição de chaves e integridade referencial no comando *CREATE TABLE*. A cláusula *PRIMARY KEY* especifica quais colunas fazem parte da chave primária da tabela, como apontado pela Sintaxe 28, onde as colunas *Id_empresa* e *Id_planta* são as colunas da chave primária da tabela *Planta*.

Sintaxe 28: Declaração de chave primária em SQL

```
CREATE TABLE Planta
(
    id_empresa    INT          NOT NULL,
    id_planta     INT          NOT NULL,
    cidade        VARCHAR(15) NOT NULL
    PRIMARY KEY(id_empresa , id_planta)
);
```

Fonte: Elaborado pelo autor

A cláusula *UNIQUE* especifica chaves alternativos, ilustrada na Sintaxe 29, onde a coluna nome deve ser única.

Sintaxe 29: Declaração de chave alternativa em SQL

```
Name VARCHAR(15) UNIQUE;
```

Fonte: Elaborado pelo autor

Restrições de integridade são definidas a partir da cláusula *FOREIGN KEY*, como apresentado na Sintaxe 30, onde a coluna *id_departamento* da tabela *Empregado* referencia a tabela *Departamento*.

Sintaxe 30: Declaração de chave estrangeira em SQL

```
CREATE TABLE departamento
(
    id_departamento INT NOT NULL,
    nome              VARCHAR(15) NOT NULL,
    PRIMARY KEY(id_departamento)
);

CREATE TABLE empregado
(
    id_empregado      INT NOT NULL,
    nome              VARCHAR(15) NOT NULL,
    idade             INT NOT NULL,
    genero             VARCHAR(1) NULL,
    id_departamento  INT NOT NULL,
    PRIMARY KEY(id_empregado)
    FOREIGN KEY(id_departamento) REFERENCES departamento(id_departamento)
);
```

Fonte: Elaborado pelo autor

2.7.3 Data Manipulation Language

Como já apontado anteriormente, a Data Manipulation Language permite formas de inserir, modificar, excluir ou consultar linhas em um banco de dados. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 110). Estes conceitos são explorados nesta seção.

2.7.3.1 Inserção, modificação e exclusão em SQL

Em SQL, três comandos podem ser utilizados para realizar as operações de inserção, modificação e exclusão, sendo eles, respectivamente: *INSERT*, *UPDATE* e *DELETE*. (EL-MASRI; NAVATHE, 2010, p. 107).

2.7.3.1.1 Inserção

Silberschatz, Korth e Sudarshan (2006, p. 130) explicam que é possível inserir dados em uma tabela a partir de uma linha ou resultado de uma consulta, contanto que os valores das colunas para as linhas a inserir pertençam ao domínio das colunas da tabela, e estejam na ordem correta.

O comando *INSERT* exibido na Sintaxe 31, ilustra a inserção de uma linha na tabela Empregado, com os dados de seu Nome, Idade, Gênero e Departamento.

Sintaxe 31: Inserção em SQL

```
INSERT INTO Empregado VALUES(1, 'Eduardo', 22, 'M', 42);
```

Fonte: Elaborado pelo autor

Existe uma outra forma de inserir valores em uma tabela, nos casos onde certas colunas podem aceitar valores nulos. Necessita-se especificar explicitamente a qual coluna o valor está sendo atribuído, conforme o exemplo da Sintaxe 32, onde um valor para a coluna Gênero não é atribuído.

Sintaxe 32: Inserção alternativa em SQL

```
INSERT INTO Empregado(id_empregado, nome, idade, id_departamento)  
VALUES(1, "Eduardo", 22, 42);
```

Fonte: Elaborado pelo autor

SGBDs que implementam SQL realizam validações referentes à integridade referencial quando executam comandos de inserção. Caso a inserção não seja válida, a operação é rejeitada. (ELMASRI; NAVATHE, 2010, p. 108).

2.7.3.1.2 Modificação

Em determinados casos, deseja-se alterar apenas alguns valores de uma linha. Para isso, o comando *UPDATE* pode ser utilizado. O exemplo da Sintaxe 33 apresenta o aumento de 5% do Salário de todos os Empregados. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 132).

Sintaxe 33: Modificação em SQL

```
UPDATE empregado  
SET salario = salario * 1,05;
```

Fonte: Elaborado pelo autor

Também é possível especificar a cláusula *WHERE* para limitar quais linhas da tabela devem ser modificadas, como exemplificado na Sintaxe 34, onde se define que apenas Contas com Saldo maior que 10.000 reais devem ser atualizadas.

Sintaxe 34: Modificação alternativa em SQL

```
UPDATE conta
SET saldo = saldo * 1,05
WHERE saldo > 10000;
```

Fonte: Silberschatz, Korth e Sudarshan (2006, p. 132)

2.7.3.1.3 Exclusão

Para remover linhas de uma tabela, utiliza-se o comando *DELETE*, o qual inclui uma cláusula *WHERE* para selecionar quais linhas que serão removidas. O exemplo da Sintaxe 35, exclui todos os Empregados pertencentes ao Departamento 42. (ELMASRI; NAVATHE, 2010, p. 109).

Sintaxe 35: Remoção em SQL

```
DELETE FROM Empregado
WHERE id_departamento = 42;
```

Fonte: Elaborado pelo autor

2.7.3.2 Consulta

O SQL provê um comando básico para realizar consultas a um banco de dados, o comando *SELECT*, o qual difere em alguns pontos em relação ao operador de seleção da álgebra relacional, apresentado na seção 2.6.1. (ELMASRI; NAVATHE, 2010, p. 97). Uma das principais diferenças é que consultas SQL não retornam um conjunto, pois permitem repetições no resultado de uma consulta.

Conforme Elmasri e Navathe (2010, p. 98), o formato mais básico do comando *SELECT* é o bloco *SELECT-FROM-WHERE*, formado pelas cláusulas *SELECT*, *FROM* e *WHERE*, conforme ilustrado na Sintaxe 36:

Sintaxe 36: Bloco SELECT-FROM-WHERE

```
SELECT < lista de colunas > FROM < lista de tabelas > WHERE <
condicao >;
```

Fonte: Elaborado pelo autor

onde:

- a <lista de colunas> é uma lista das colunas a serem obtidas pela consulta;

- a <lista de tabelas> contém a lista das tabelas a serem processadas e;
- a <condição> especifica uma expressão booleana para limitar quais linhas devem ser obtidas pela consulta.

A Sintaxe 37 exemplifica o *SELECT*, mostrando uma consulta ao nome e gênero dos Empregados que tem mais de 30 anos.

Sintaxe 37: Consulta básica em SQL

```
SELECT nome, idade  
FROM empregado  
WHERE idade > 30;
```

Fonte: Elaborado pelo autor

Assim como apontado na álgebra relacional, as cláusulas *WHERE* também podem utilizar operadores de comparação tais como igualdade(=), diferente (! =),menor que (<), maior que (>), menor ou igual que (≤), maior igual que(≥), todavia, os conectivos 'e' e 'ou' são representados por *AND* e *OR*, respectivamente. A Sintaxe 38 ilustra o uso dos operadores.

Sintaxe 38: Consulta básica com conectivos em SQL

```
SELECT nome, idade  
FROM empregado  
WHERE idade > 30 AND genero = 'M' ;
```

Fonte: Elaborado pelo autor

Na Sintaxe 39, apresenta-se uma consulta SQL onde a condição de seleção é se o nome do departamento é o de pesquisa, além disso, a condição *numerodepartamento = deptonum* é chamada de condição de junção, pois ela combina as linhas da tabela Departamento que tenham o valor de *numerodepartamento* igual ao valor de *deptonum* da tabela Empregado.

Sintaxe 39: Consulta com junção em SQL

```
CREATE TABLE departamento
(
    numerodepartamento INT NOT NULL,
    dnome                VARCHAR(15) NOT NULL
    PRIMARY KEY(numerodepartamento)
);

CREATE TABLE empregado
(
    id_empregado INT NOT NULL,
    enome        VARCHAR(15) NOT NULL,
    idade        INT NOT NULL,
    genero       VARCHAR(1) NULL,
    deptonum     INT NOT NULL,
    PRIMARY KEY(id_empregado)
    FOREIGN KEY(deptonum) REFERENCES departamento(numerodepartamento)
);

SELECT enome, dnome, idade
FROM Empregado, Departamento
WHERE Dnome = "pesquisa" AND deptonum = numerodepartamento;
```

Fonte: Adaptado de Elmasri e Navathe (2010, p. 100)

Em SQL, um mesmo nome de coluna pode existir em tabelas diferentes. Neste cenário, quando realizamos uma consulta que utiliza tabelas que contém tabelas com colunas homônimas, deve-se qualificar o nome da coluna com um pré-fixo relacionado à tabela a qual a coluna pertence. O exemplo da Sintaxe 40 apresenta tal uso, onde a tabela Empregado e a tabela Departamento possuem uma coluna com o mesmo nome, a coluna Numerodepartamento.

Sintaxe 40: Consulta em SQL

```

CREATE TABLE departamento
(
    numerodepartamento INT NOT NULL,
    dnome VARCHAR(15) NOT NULL
    PRIMARY KEY(numerodepartamento)
);

CREATE TABLE empregado
(
    id_empregado INT NOT NULL,
    enome VARCHAR(15) NOT NULL,
    idade INT NOT NULL,
    genero VARCHAR(1) NULL,
    numerodepartamento INT NOT NULL,
    PRIMARY KEY(id_empregado)
    FOREIGN KEY(numerodepartamento) REFERENCES departamento(
        numerodepartamento)
);

SELECT enome, dnome, idade
FROM Empregado, Departamento
WHERE Dnome = "pesquisa" AND
    Empregado.numerodepartamento = Departamento.numerodepartamento;

```

Fonte: Adaptado de Elmasri e Navathe (2010, p. 100)

Para obter todas as colunas de uma determinada tabela, especifica-se um asterisco (*), que significa todas as colunas. O exemplo da Sintaxe 41 mostra o uso do asterisco, onde se busca todas as colunas da tabela Empregado.

Sintaxe 41: Consulta em SQL - Exemplo 2

```

SELECT *
FROM Empregado
WHERE idade > 30 ;

```

Fonte: Elaborado pelo autor

2.7.3.3 Operações em conjuntos

Em SQL, os operadores *union*, *intersect* e *except* correspondem respectivamente aos operadores \cup , \cap e $-$ da álgebra relacional. Assim como na álgebra relacional, os participantes das operações devem ser compatíveis no que se concede a seu conjunto de colunas. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 118).

2.7.3.3.1 Union

Conforme Silberschatz, Korth e Sudarshan (2006, p. 118) define que, "a operação de union, ao contrário da cláusula select automaticamente elimina as repetições". A Sintaxe 42 mostra uma consulta SQL a qual utiliza o operador union; onde se deseja encontrar todos os clientes do banco que possuem empréstimos, uma conta, ou ambos.

Sintaxe 42: Operador de união

```
(SELECT nomecliente  
FROM depositante)  
UNION  
(SELECT nome_cliente  
FROM devedor);
```

Fonte: Silberschatz, Korth e Sudarshan (2006, p. 118)

2.7.3.3.2 Intersect

Segundo Silberschatz, Korth e Sudarshan (2006, p. 119), a operação de intersecção também elimina duplicatas. A Sintaxe 43 exemplifica o uso da operação de intersecção, onde se deseja encontrar todos os clientes que tenham tanto empréstimos quanto contas no banco.

Sintaxe 43: Operador de intersecção

```
(SELECT nomecliente  
FROM depositante)  
INTERSECT  
(SELECT nome_cliente  
FROM devedor);
```

Fonte: Silberschatz, Korth e Sudarshan (2006, p. 118)

2.7.3.3.3 Except

A operação exceto também elimina duplicatas. A Sintaxe 44 apresenta um possível uso do operador except, onde se deseja encontrar todos os clientes que tenham uma conta e nenhum empréstimo no banco. (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 119).

Sintaxe 44: Operador except

```
(SELECT nomecliente  
FROM depositante)  
EXCEPT  
(SELECT nome_cliente  
FROM devedor);
```

Fonte: Silberschatz, Korth e Sudarshan (2006, p. 118)

2.7.3.4 Funções agregadas

Segundo Silberschatz, Korth e Sudarshan (2006, p. 120), as funções agregadas da SQL, similares às funções agregadas da álgebra relacional, tomam uma coleção de valores como entrada e retornam um simples valor. A SQL fornece cinco funções agregadas:

1. **avg**: média (average);
2. **min**: mínimo (minimum);
3. **max**: máximo (maximum);
4. **sum**: total (total);
5. **count**: contagem(count);

Ilustra-se o uso de uma função agregada na Sintaxe 45 , onde se busca a média dos saldos em contas na agência 295.

Sintaxe 45: Funções agregadas em SQL

```
SELECT avg(saldo)  
FROM conta  
where agencia = 295;
```

Fonte: Silberschatz, Korth e Sudarshan (2006, p. 118)

Em certos casos, Silberschatz, Korth e Sudarshan (2006, p. 120) relata que em alguns casos é necessário utilizar funções agregadas em um conjunto de linhas, o que se torna possível com a utilização da cláusula *group by*, como exemplificado na Sintaxe 46, onde se encontra a média dos saldos nas contas de cada uma das agência de um banco.

Sintaxe 46: Funções agregadas em SQL - exemplo 2

```
SELECT nomeagencia , avg( saldo )  
FROM conta  
GROUP BY nomeagencia ;
```

Fonte: Silberschatz, Korth e Sudarshan (2006, p. 118)

2.8 Ferramentas CASE

O acrônimo CASE, se refere à *Computer Aided Software Engineering*, e significa , em uma tradução literal, "Engenharia de Software Auxiliada por Computador". Ferramentas CASE são um conjunto de ferramentas informáticas e conjuntos de técnicas que auxiliam o profissional de tecnologia da informação no desenvolvimento de aplicações, diminuindo esforço e complexidade, e melhorando o controle do projeto. (SILVA; VIDEIRA, 2001, p. 397).

Segundo Lyytinen e Tahvanainen (1992, p. 2), inicialmente, nos meados da década de 70, ferramentas CASE tinham o intuito de automatizar completamente o processo de fabricação de software, porém, tal abordagem não funcionou, e poucas ferramentas obtiveram sucesso. Mais tarde, nos anos 80 e 90, com a introdução das interfaces gráficas, elas ficaram mais populares, pois incluíam suporte à diagramas de fluxo de dados, estados, transição ou à notações de modelo de dados. Apesar da evolução destas ferramentas, elas ainda apresentam uma série de desvantagens:

- ferramentas CASE não possuem boa integração com outras ferramentas CASE;
- a metodologia utilizada por certas ferramentas não estão suficientemente formalizadas;
- em grande parte, não possuem suporte ao uso concorrente de vários usuários, dificultando assim, o desenvolvimento em times.

Nos últimos anos, diversas ferramentas relacionadas à engenharia de software e modelagem de banco de dados emergiram. Com a popularização das linguagens orientadas a objeto e da UML, o uso de ferramentas CASE se tornou frequente. (SILVA; VIDEIRA, 2001, p. 397).

2.8.1 Ferramentas CASE para modelagem de banco de dados

Segundo Elmasri e Navathe (2010, p. 343), diversas ferramentas CASE oferecem suporte para a modelagem de banco de dados, tipicamente oferecendo as seguintes funcionalidades:

- **diagramação:** permite que o usuário desenhe esquemas conceituais e lógicas baseado em uma notação específica, permitindo mapear entidade, relacionamentos, cardinalidade, restrições, chaves, entre outros. Algumas disponibilizam mecanismos para utilizar especialização/generalização;

- **mapeamento de modelos:** a partir de um modelo, a ferramenta pode gerar instruções SQL para a automação da criação de esquemas de banco de dados;
- **normalização:** a ferramenta aplica as regras de normalização sobre um banco de dados.

Ainda, conforme Elmasri e Navathe (2010, p. 344), também é desejável que uma ferramenta CASE para banco de dados ofereça:

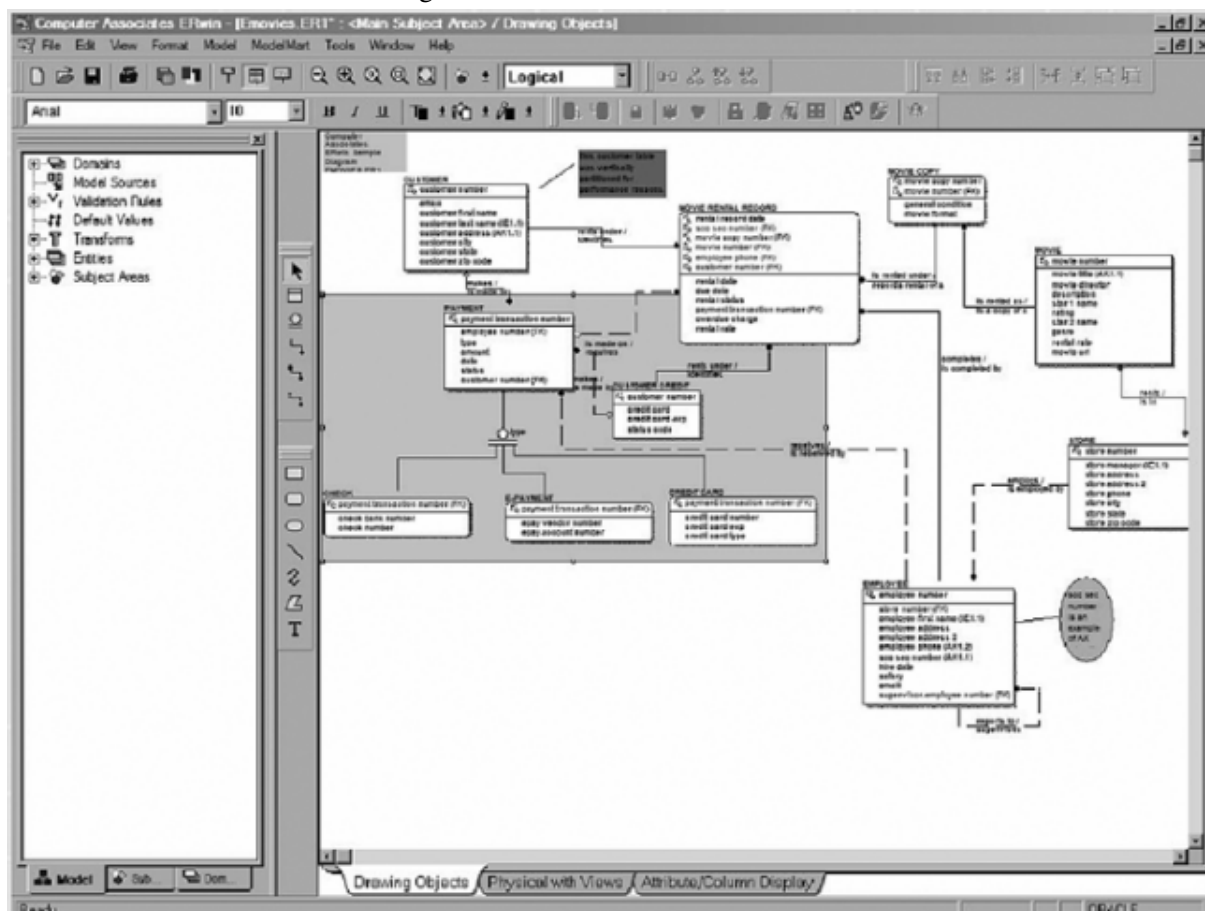
- uma interface gráfica simples de utilizar;
- componentes analíticos que detectem problemas de restrições e analisem possíveis modelagens alternativas;
- componentes heurísticos para analisar modelagens alternativas;
- comparação das possíveis modelagens alternativas;
- verificação da coesão do modelo de dados.

Teorey et al. (2010, p. 190) explicam que, apesar dos vários passos para a criação de um projeto de banco de dados, grande parte das ferramentas CASE focam nas etapas de análise e modelo lógico, e nesses pontos que ferramentas CASE para banco de dados se mostra, particularmente eficientes.

Teorey et al. (2010, p. 190) descrevem ainda que, entre as ferramentas CASE mais utilizadas comercialmente, pode-se listar o AllFusion ERwin (SYBASE, 2013), que possui um longo histórico e diversas características tais como um bom suporte à geração do modelo físico para diversas plataformas, da mesma maneira, podem-se citar os concorrentes Power Designer da SAP Sybase (ASSOCIATES, 2013), o Rational Rose Data Modeler da IBM (IBM, 2013), e o SQL Developer Data Modeler (ORACLE, 2013), entre outros que podem ser encontrados online (ANSWERS, 2013).

A Figura 46 mostra o uso de uma ferramenta CASE para modelagem de banco de dados, neste caso, a ferramenta ERWin.

Figura 46: Ferramenta CASE ERWin



Fonte: Teorey et al. (2010, p. 190)

2.9 Educação a distância

Nova e Alves (2003) definem educação a distância (EAD) como uma modalidade de transmissão de conhecimento onde não há a presença dos agentes envolvidos, porém, conforme Beloni (2001, p. 25) existem várias definições de educação a distância, dentre as quais se destacam a de Kahl e Cropley (1986):

Educação a distância é uma espécie de educação baseada em procedimentos que permitem o estabelecimento de processos de ensino e aprendizagem mesmo onde não existe contato face a face entre professores e aprendentes - ela permite um alto grau de aprendizagem individualizada.

E de Tight (1988):

Educação a distância se refere àquelas formas de aprendizagem organizada, baseadas na separação física entre os aprendentes e os que estão envolvidos

na organização de sua aprendizagem. Esta separação pode aplicar-se a todo o processo de aprendizagem ou a apenas certos estágios ou elementos deste processo.

Moore e Kearsley (2011, p. 95) sintetizam que o uso da tecnologia auxiliou em grande parte a popularidade da educação a distância, e cada vez mais, com a possibilidade de utilizar redes sociais e aplicações de dispositivos móveis, a natureza da educação a distância se modifica e amplia sua extensão. Em uma sociedade cada vez mais interconectada, e com maior acesso à internet, a educação a distância se difundiu em todo o mundo, transformando assim o cenário educacional, e ampliando a difusão do conhecimento. (NOVA; ALVES, 2003).

3 TRABALHOS RELACIONADOS

Neste capítulo, sumariza-se os trabalhos que propõe uma abordagem similar a este.

3.1 ACME-DB

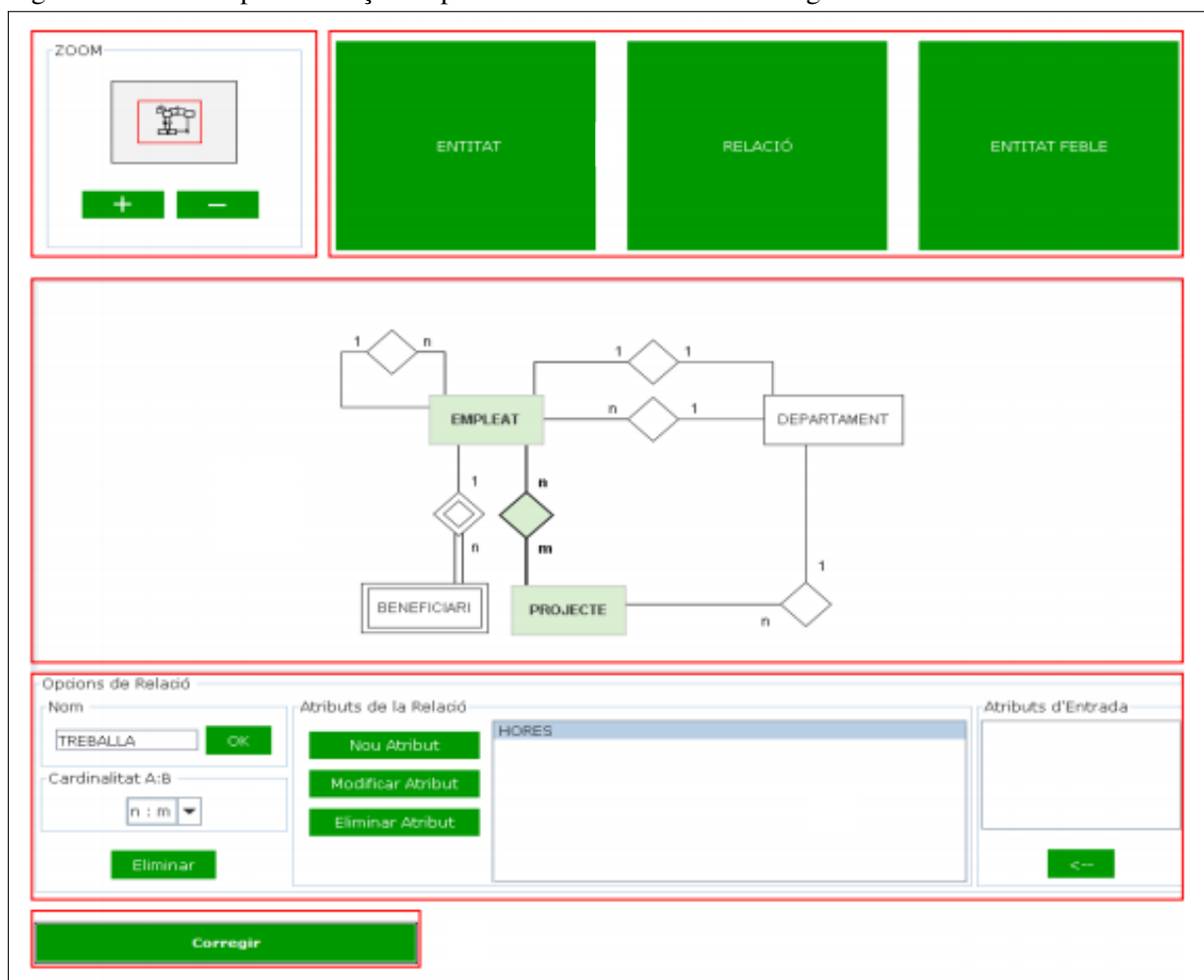
O ACME-DB, elaborado por Masó, Gesa e Matemática Aplicada (2010), é um ambiente de avaliação baseada em computador para alunos da disciplina de banco de dados. Este ambiente apresenta diversos módulos, que abordam grande parte do que é coberto na disciplina de banco de dados, sendo eles:

- módulo de diagramas ER;
- módulo de diagramas de classes;
- modulo de esquemas para banco de dados relacional;
- modulo para normalização de banco de dados;
- módulo de sentenças SQL.

Porém, a ferramenta não permite a diagramação dos modelos ER. O aluno deve entrar com os dados a partir de formulários, a ferramenta então, gera os diagramas a partir de uma ferramenta de grafos.

A Figura 47 apresenta a interface gráfica para a solução de problemas envolvendo a modelagem ER.

Figura 47: Interface para a solução de problemas envolvendo a modelagem ER na ferramenta ACME-DB



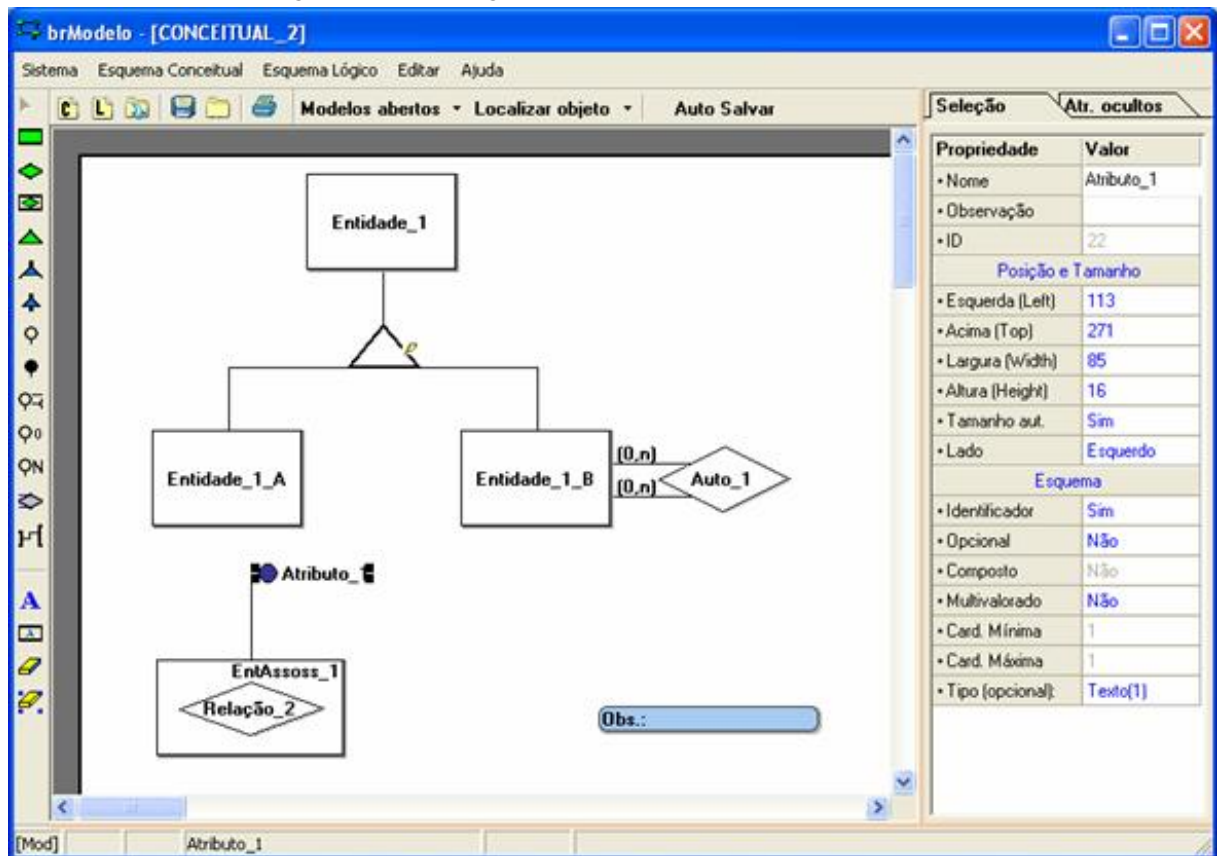
Fonte: Adaptado de Masó, Gesa e Matemàtica Aplicada (2010)

3.2 BrModelo

O BrModelo é uma ferramenta de ensino de banco de dados relacional voltada para a modelagem conceitual, lógica e física. Permite a diagramação de modelos conceituais e lógicos, assim como transformação entre os mesmos. Também permite a geração de um esquema em SQL. (CÂNDIDO, 2008).

A Figura 48 apresenta a modelagem conceitual na ferramenta BrModelo.

Figura 48: Modelagem conceitual na ferramenta BrModelo



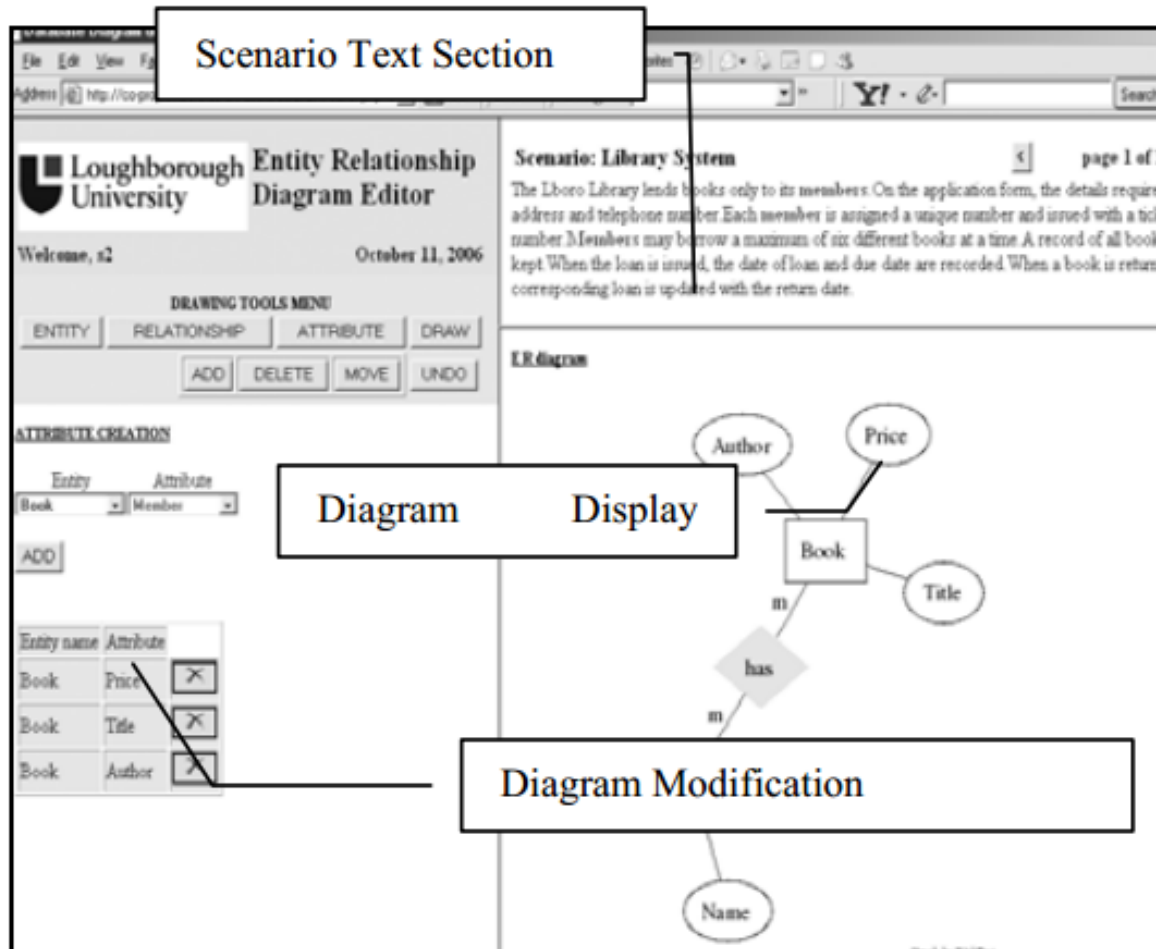
Fonte: CÂNDIDO (2008)

3.3 A web-based semi-automatic assessment tool for conceptual database diagram

No trabalho de Batmaz e Hinde (2007), utiliza-se uma abordagem similar ao ACME-DB, onde o aluno cria o modelo de dados conceitual de um banco de dados a partir de formulários, e o sistema então gera os diagramas e realiza uma correção semi-automática da resposta do aluno.

A Figura 49 apresenta o uso da ferramenta elaborada por Batmaz e Hinde (2007).

Figura 49: Ferramenta elaborada por Batmaz e Hinde (2007)



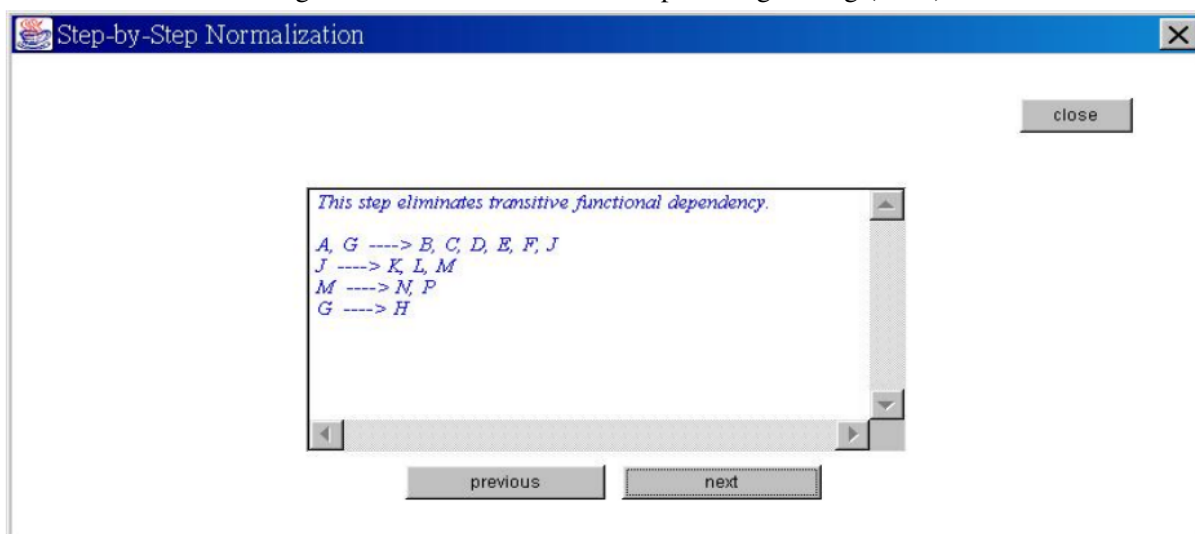
Fonte: Batmaz e Hinde (2007)

3.4 A web-based tool to enhance teaching/learning database normalization

Kung e Tung (2006) apresentam o uso de uma ferramenta criada com o intuito de auxiliar os alunos da disciplina de banco de dados a entender o processo de normalização de um banco de dados. A partir de uma ferramenta disponibilizada na web, o estudante informa as relações a serem normalizadas e pode visualizar um passo-a-passo da normalização; conclui-se que a ferramenta teve um impacto positivo sobre o desempenho dos alunos.

A Figura 50 mostra o uso da ferramenta de normalização elaborada por Kung e Tung (2006).

Figura 50: Ferramenta elaborada por Kung e Tung (2006)



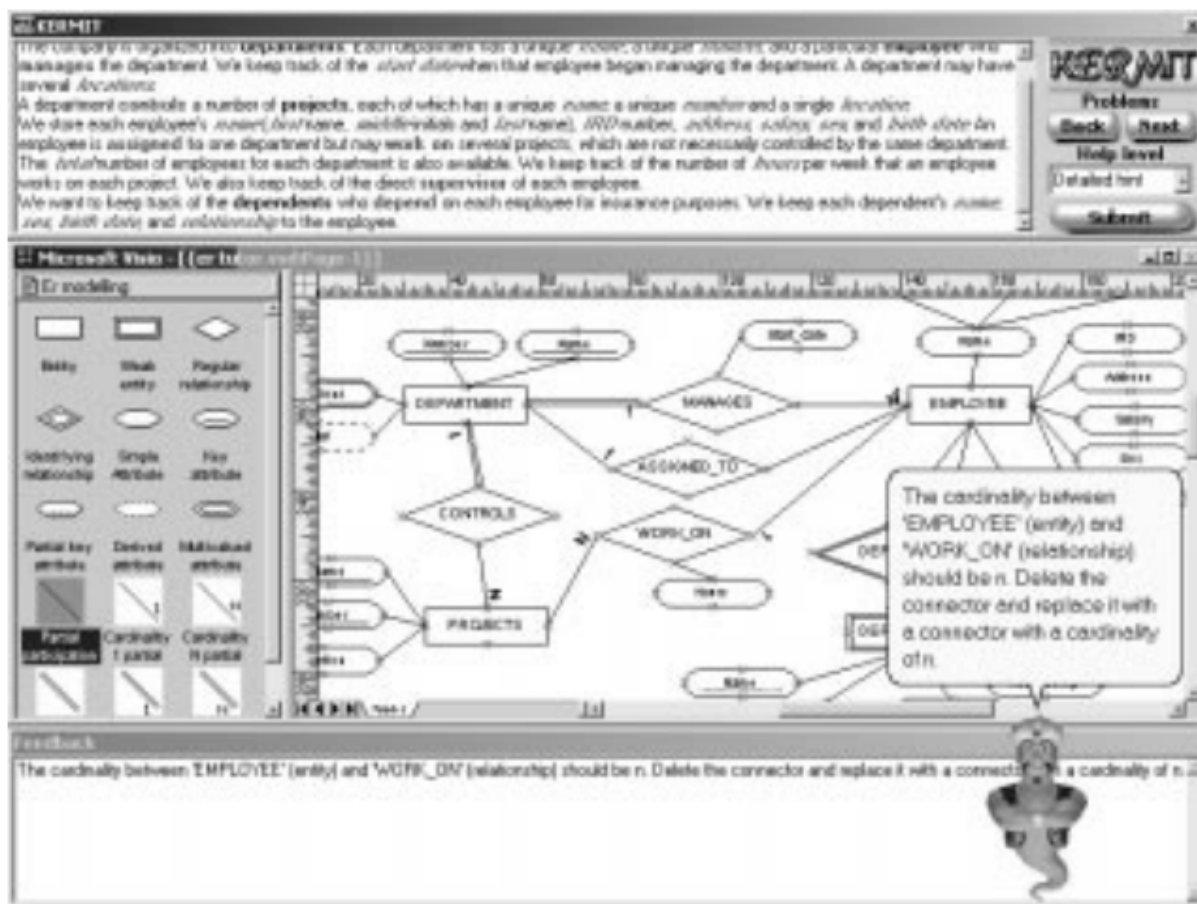
Fonte: Kung e Tung (2006)

3.5 KERMIT

O KERMIT, elaborado por (SURAWEEERA; MITROVIC, 2002), é uma ferramenta-tutor para o ensino de modelagem conceitual utilizando o modelo entidade-relacionamento. Como a criação de um modelo de dados para o mesmo problema pode variar, utiliza-se um modelo baseado em integridades para validações.

A Figura 51 apresenta o uso da ferramenta KERMIT.

Figura 51: O KERMIT



Fonte: Suraweera e Mitrovic (2002)

3.6 Um protótipo de ambiente de apoio ao processo de ensino aprendizagem de banco de dados: módulo SQL

Pereira (2011) apresenta o uso de um protótipo que realiza a correção automática para comandos SQL, o aluno pode criar, inserir, excluir e ou consultar dados, e o sistema avalia se os resultados das expressões é o correto.

A Figura 52 ilustra o uso do protótipo de ensino de SQL.

Figura 52: O protótipo de ambiente de apoio ao processo de ensino aprendizagem de banco de dados: módulo SQL

Fonte: Pereira (2011)

3.7 An automatic correction tool for relational algebra queries

Soler et al. (2007) apresentam uma ferramenta a qual, dado um problema de álgebra relacional descrito por um professor, os alunos devem resolvê-lo utilizando expressões em álgebra relacional. A ferramenta valida a entrada do aluno e fornece *feedback* instantâneo, apontando os erros e em como resolvê-los.

A Figura 53 apresenta o uso da ferramenta de correção automática de consultas em álgebra relacional.

Figura 53: Ferramenta de correção automática de consultas em álgebra relacional

ACME > Courses > Subject > Problem > Visualization of the descriptor

Retrieve the name and address of all employees who work for the "Research" department. (a)

R1 ← σ_{DNAME="Research"}(DEPARTMENT)

R2 ← R1 ⋈ DNUMBER=DNO (EMPLOYEE)

(d)

(g) New relations

R1
R2

(b) Initial relations

EMPLOYEE
DEPARTMENT
PROJECT
WORKS_ON
DEPENDENTS

(c) Columns

SSN
DNO
SUPERSSN
FNAME
LNAME
BDATE
ADDRESS
SEX

clean Remove relation

Relation : R3 ← π_{FNAME,LNAME,ADDRESS}(R2) (e)

Enter

Correct (h)

Subject	Problem Number	Status	Result Errors	Syntax Errors	Limit Date
6	8	Unsolved	0	0	22/12/2007

(l)

See Sent Solutions Prints this page

Fonte: Soler et al. (2007)

4 METODOLOGIA DE PESQUISA

Esta pesquisa será de natureza aplicada, pois irá gerar conhecimento para a resolução de problemas específicos. (KAUARK; MANHÃES; SOUZA, 2010).

O estudo envolverá como estratégia de pesquisa o estudo de caso múltiplo, que segundo Schramm (1971) e citado por Yin (2005, p. 31), um estudo de caso "[...] tenta esclarecer uma decisão ou um conjunto de decisões: o motivo pelo qual foram tomadas, como foram implementadas e com quais resultados.". O estudo de caso auxilia a responder a questões do tipo "porque" e "como", lidando de forma simples com um cenário complexo, ajudando a entender o fenômeno observado e suas relações dentro de um contexto. (STAKE, 1995; MERRIAM, 1998).

Para Gil (2002, p. 139), "de modo geral, considera-se que a utilização de múltiplos casos proporciona evidências inseridas em diferentes contextos, concorrendo para a elaboração de uma pesquisa de melhor qualidade."

Os estudos de caso serão aplicados à três grupos:

1. Alunos da disciplina de banco de dados que utilizarão o protótipo;
2. Professor da disciplina de banco de dados que contém os alunos supracitados;
3. Professores de banco de dados.

Nos estudos de caso de alunos da disciplina e professores de bancos de dados, será utilizada uma abordagem quantitativa, sendo que a técnica de coleta de dados terá como base questionários.

A pesquisa quantitativa é baseada na medida em escala numérica de variáveis objetivas, com o intuito de analisar resultados e utilizá-los para o uso de técnicas estatísticas; é particularmente útil quando se deseja provar o quão mais satisfatório é o uso de um sistema em relação às alternativas. (WAINER, 2007).

Questionários, por sua vez, são constituídos por uma série de perguntas que podem ser respondidas sem a presença do entrevistador e tem por vantagem economizarem tempo, atingirem um maior número de pessoas e obterem respostas mais rápidas e precisas. (MARCONI; LAKATOS, 2003, p. 201).

Para a construção das perguntas, será utilizada a escala Likert, proposta por Likert (1932). A escala provê meios de entender a opinião ou grau de concordância do entrevistado em relação à uma afirmação ou um conjunto de afirmações, indica-se esta opinião ou grau através de uma escala ordinal, tipicamente formada por cinco itens: não concordo totalmente, não concordo, indiferente, concordo e concordo totalmente. (BERTRAM, 2006; MCIVER; CARMINES, 1981)

A escala tornou-se um arquétipo importante para a mensuração qualitativa devido à possibilidade de uma interpretação semântica e interpretação dos resultados. (PEREIRA, 1999).

Para o estudo de caso do professor da disciplina de banco de dados que utilizará o protótipo deste estudo, será utilizada uma abordagem qualitativa, com a utilização de entrevistas.

Silva e Menezes (2001) explicam que a pesquisa qualitativa

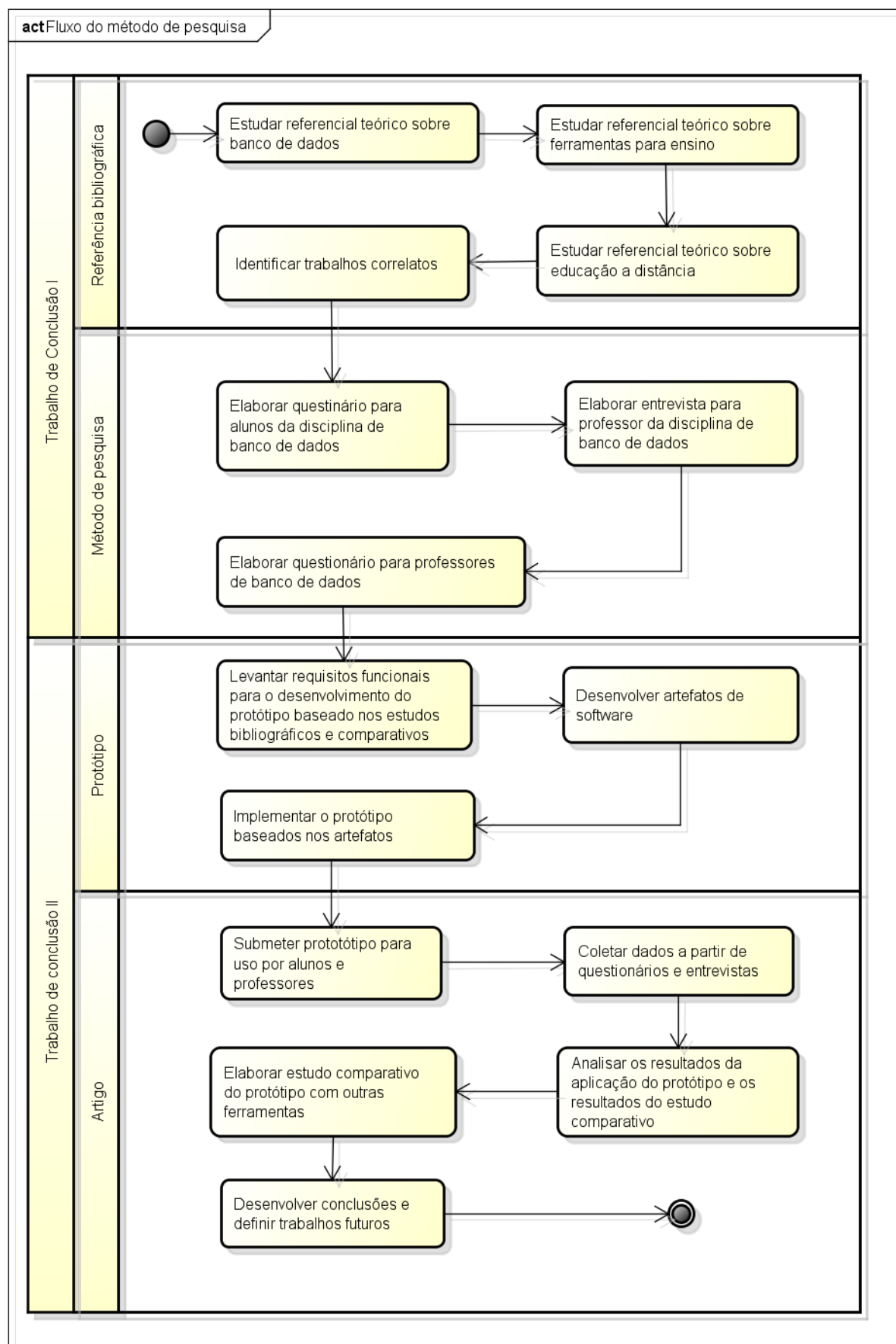
considera que há uma relação dinâmica entre o mundo real e o sujeito, isto é, um vínculo indissociável entre o mundo objetivo e a subjetividade do sujeito que não pode ser traduzido em números. [...]. É descritiva. Os pesquisadores tendem a analisar seus dados indutivamente. O processo e seu significado são os focos principais de abordagem.

Métodos qualitativos em ciência da computação se caracterizam em estudar um sistema no ambiente onde ele está sendo usado. (WAINER, 2007). As pesquisas qualitativas, em sua maioria, utilizam a realização de entrevistas, sendo que a entrevista é um encontro entre duas pessoas, com o objetivo de obter informações sobre um assunto, proporcionando verbalmente a informação necessária. (DUARTE, 2002).

Ainda, será elaborado um estudo comparativo do protótipo elaborado em relação às ferramentas CASE e acadêmicas de banco de dados com base em suas documentações. O estudo comparativo busca semelhanças e diferenças entre os fatos, gerando conclusões em suas correlações. (VASQUES, 2008).

A Figura 54 apresenta o fluxo da metodologia que será utilizado neste trabalho.

Figura 54: Fluxo do método de pesquisa



5 PROTÓTIPO E TECNOLOGIAS UTILIZADAS

Dado que o protótipo será disponibilizado via web, para o *front-end* será utilizado o HTML5 e Dart¹, uma linguagem de script estruturada voltada para web.

O HTML5 é nova geração do HTML (Hypertext Markup Language), pois provê novos recursos e padrões para o desenvolvimento de aplicações web modernas, e já é suportado na maioria dos navegadores. A principal escolha do HTML5 se dá à introdução de um novo elemento, o Canvas, que é um bitmap que pode ser utilizado para renderizar gráficos e imagens. (W3C, 2013; PILGRIM, 2010).

O Dart é uma linguagem de script que é orientada a objetos, estruturada e com tipagem opcional, e tem sintaxe semelhante a linguagem C e o Java. Sua escolha se dá por diversos motivos, tais como: sua simplicidade e escabilidade; diversas ferramentas para programar e depurar o código; diversas bibliotecas; conversão para JavaScript, facilitando assim a portabilidade; familiaridade do autor com a linguagem Java. (MOKKAPATI, 2012; DART, 2013; BRIGHT, 2013).

Para o *back-end*, será utilizado o Ruby on Rails², que é um framework de desenvolvimento web com o intuito de aumentar a produtividade do programador. Por ser *open-source* e se adaptar rapidamente às novas tecnologias web, tem uma grande comunidade e é utilizado por uma diversa gama de empresas. (BACHLE; KIRCHBERG, 2007; HARTL, 2010).

Os serviços serão disponibilizados e hospedados a partir da plataforma Heroku³, uma plataforma para distribuir programas na nuvem. A escolha do Heroku se deve a sua integração intrínseca com a plataforma Ruby on Rails, e por suas facilidades, no que se concede a sua simplicidade e automação da gerência do website. Será utilizado o banco de dados open-source PostgreSQL⁴, que é o banco de dados padrão no Heroku, por isso a sua predileção. (KEMP; GYGER, 2013; HEROKU, 2013).

As consultas de SQL e álgebra relacional utilizarão *schemas* do mesmo banco de dados e, as consultas em álgebras relacional, serão processadas utilizando a biblioteca RA (Relational Algebra Interpreter)⁵, que é um interpretador de álgebra relacional, que traduz consultas em álgebra relacional para SQL.

Dado as limitações de prototipagem, será garantido apenas a execução do protótipo no navegador Google Chrome.

A Figura 55 ilustra a arquitetura proposta neste trabalho.

¹<http://www.dartlang.org/>

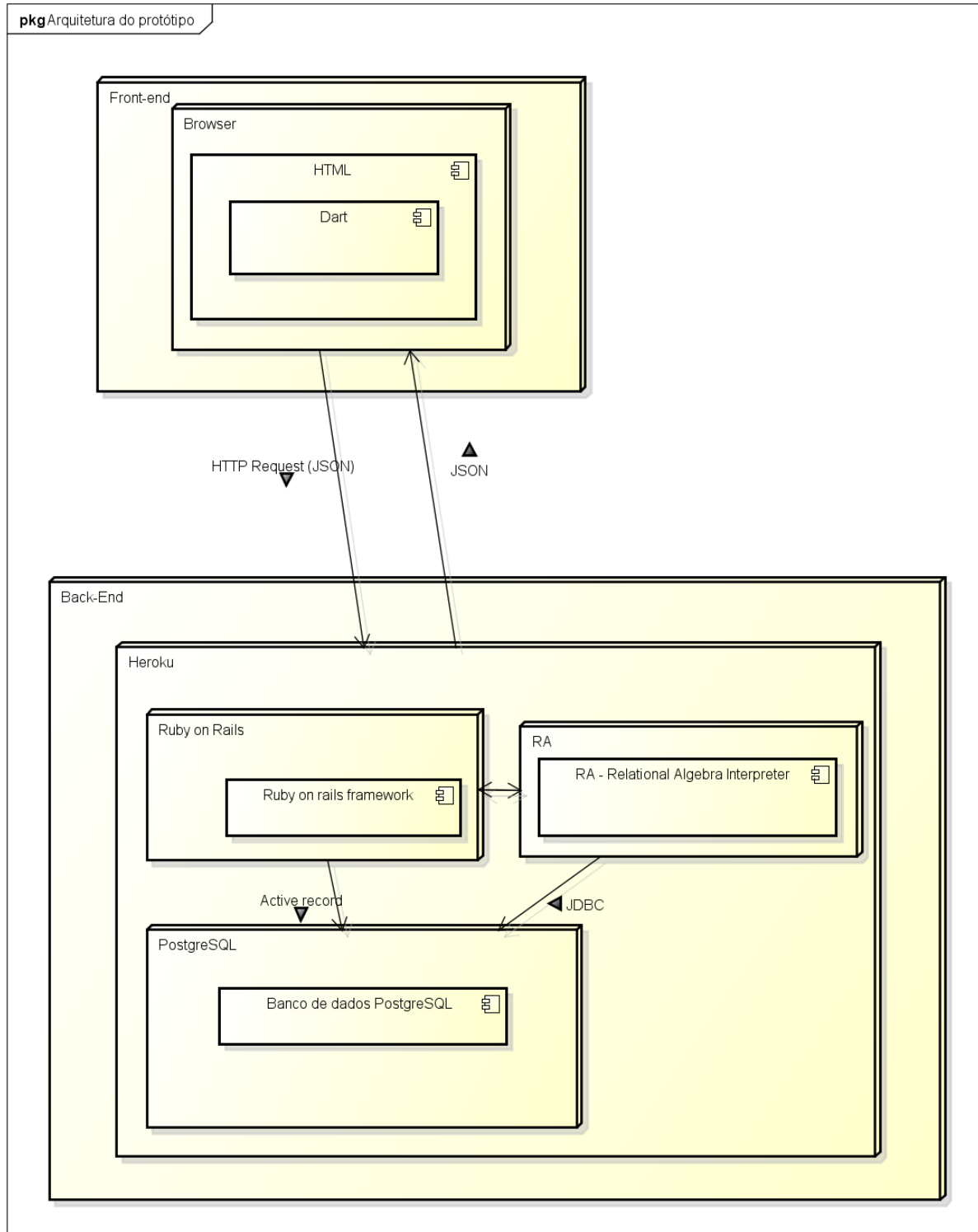
²<http://rubyonrails.org/>

³<https://www.heroku.com/>

⁴<http://www.postgresql.org/>

⁵<https://www.cs.duke.edu/junyang/ra/>

Figura 55: Arquitetura do protótipo



6 CONCLUSÃO

Neste capítulo, abordam-se as conclusões atingidas nesse trabalho, assim como, as próximas etapas a serem realizadas.

6.1 Conclusão final

Através do estudo realizado sobre o referencial teórico, evidencia-se que o tema de banco de dados é extenso e detalhado, porém, este conhecimento é essencial para os profissionais de tecnologia da informação.

Observa-se também que, a modelagem de dados é uma etapa importante na construção de um software, assim como sua implantação em um banco de dados e o modo de realizar consultas às informações em um banco de dados.

A partir da sumarização dos trabalhos relacionados a este, observa-se que a disciplina de banco de dados pode ser ministrada com o auxílio de ferramentas educacionais, constatado que sua utilização é benéfica.

6.2 Trabalhos futuros

Neste sentido, será dada continuidade a este trabalho, com o desenvolvimento do protótipo, questionários e entrevistas, assim como o estudo comparativo, para desta forma, analisar os resultados da aplicação do uso da ferramenta.

6.3 Cronograma

A Figura 56 aponta o cronograma que foi e será seguido na execução deste estudo.

Figura 56: Cronograma

Atividades/Período	ago/13	set/13	out/13	nov/13	mar/14	abr/14	mai/14	jun/14
Entrega da proposta								
Estudo referencial teórico banco de dados								
Estudo referencial teórico ferramentas para ensino								
Estudo referencial teórico educação a distância								
Elaboração dos questionários								
Elaboração da entrevista								
Identificação de estudo correlatos								
Entrega da monografia								
Elaboração de requisitos funcionais								
Desenvolvimento de artefatos de software								
Implementação do protótipo								
Elaboração de estudo comparativo								
Coleta de dados a partir de questionários e entrevistas								
Desenvolver conclusões e definir trabalhos futuros								
Entrega do artigo								
Apresentação à banca								

Fonte: Elaborado pelo autor

REFERÊNCIAS

- ANSWERS, Database. **Data Modelling Tools**. Disponível em:<http://www.databaseanswers.org/modelling_tools.htm>. Acesso em: 17 out 2013.
- ASSOCIATES, Computer. **CA ERwin® Data Modeler Community Edition**. Disponível em:<<http://erwin.com/products/data-modeler/community-edition>>. Acesso em: 17 out 2013.
- BACHLE, Michael; KIRCHBERG, Paul. Ruby on rails. **Software, IEEE**, [S.l.], v. 24, n. 6, p. 105–108, 2007.
- BARROS, Monalisa Alves. Ferramentas interativas na educação a distância: benefícios alcançados a partir da sua utilização. , [S.l.], 2010.
- BATMAZ, F; HINDE, CJ. A web-based semi-automatic assessment tool for conceptual database diagram. In: WEB-BASED EDUCATION CONFERENCE, 2007. **Proceedings...** [S.l.: s.n.], 2007. p. 427–432.
- BELLONI, Maria Luiza. **Educação a distância**. [S.l.]: Autores Associados, 2001.
- BERTRAM, Dane. **Likert scales**. 2006.
- BOGDANOVIĆ, Miloš; STANIMIROVIĆ, Aleksandar; DAVIDOVIĆ, Nikola; STOIMENOV, Leonid. The development and usage of a relational database design tool for educational purposes. In: OF THE SIXTH , 2008. **Anais...** [S.l.: s.n.], 2008.
- BRIGHT, Peter. **JavaScript has problems. Do we need Dart to solve them?** Disponível em:<<http://arstechnica.com/business/2011/10/javascript-has-problems-can-googles-dart-solve-them/>>. Acesso em: 08 ago 2013.
- BURLESON, Donald K.; KELLY, Laurie. **Physical Database Design Using Oracle**. Boston, MA, USA: Auerbach Publications, 2004.
- CÂNDIDO, Carlos H. brModelo: ferramenta de modelagem conceitual de banco de dados. **Trabalho de Conclusão de Curso de Pós-Graduação (Banco de dados)–UFSC, Santa Catarina.**, [S.l.], v. 1, 2008.
- CASSEL, Lillian; CLEMENTS, Alan; DAVIES, Gordon; GUZDIAL, Mark; MCCAULEY, Renée; MCGETTRICK, Andrew; SLOAN, R; SNYDER, Larry; TYMANN, Paul; WEIDE, B. Computer science curriculum 2008: an interim revision of cs 2001. **Report from the interim review task force**, [S.l.], 2008.
- CHEN, Peter Pin-Shan. The entity-relationship model—toward a unified view of data. **ACM Transactions on Database Systems (TODS)**, [S.l.], v. 1, n. 1, p. 9–36, 1976.
- CODD, E. F. A relational model of data for large shared data banks. **Commun. ACM**, New York, NY, USA, v. 13, n. 6, p. 377–387, June 1970.
- DART. **Dart Technical Overview**. Disponível em:<<http://www.dartlang.org/docs/technical-overview>>. Acesso em: 08 ago 2013.

DATE, C.J. **An Introduction to Database Systems**. 8. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

DIETRICH, Suzanne; URBAN, Susan D. Database Theory In Practice: learning from cooperative group projects. In: IN PROCEEDINGS 27TH SIGCSE TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 1996. **Anais...** ACM Press, 1996. p. 112–116.

DUARTE, Rosália. Pesquisa qualitativa: reflexões sobre o trabalho de campo. **Cadernos de Pesquisa**, [S.l.], n. 115, Mar. 2002.

ELMASRI, Ramez; NAVATHE, Shamkant. **Fundamentals of Database Systems**. 6th. ed. USA: Addison-Wesley Publishing Company, 2010.

GARCIA-MOLINA, Hector; ULLMAN, Jeffrey D.; WIDOM, Jennifer. **Database Systems: the complete book**. 2. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2008.

GIL, Antonio Carlos. Como elaborar projetos de pesquisa. **São Paulo**, [S.l.], v. 5, 2002.

HALPIN, Terry; MORGAN, Tony. **Information Modeling and Relational Databases**. 2. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

HARTL, Michael. **Ruby on Rails 3 tutorial**: learn rails by example. [S.l.]: Pearson Education, 2010.

HAY, David C. A Comparison OF DATA MODELING TECHNIQUES. , [S.l.], 1999.

HEROKU. **Ruby Support**. Disponível em:<<https://devcenter.heroku.com/articles/ruby-support>>. Acesso em: 08 ago 2013.

HEUSER, Carlos Alberto. **Projeto de banco de dados**. [S.l.]: Sagra Luzzatto, 2001.

IBM. **Rational data modeler**. Disponível em:<<http://www-03.ibm.com/software/products/br/pt/datamodeler/>>. Acesso em: 17 out 2013.

KAHL, Thomas N; CROPLEY, Arthur J. Face-to-face versus distance learning: psychological consequences and practical implications. **Distance Education**, [S.l.], v. 7, n. 1, p. 38–48, 1986.

KAUARK, Fabiana da Silva; MANHÃES, Fernanda Castro; SOUZA, Carlos Henrique Medeiros de. **Metodologia da Pesquisa**: um guia prático. [S.l.: s.n.], 2010.

KEMP, C.; GYGER, B. **Professional Heroku Programming**. [S.l.]: Wiley, 2013. (Programmer to programmer).

KUNG, H; TUNG, H. A web-based tool to enhance teaching/learning database normalization. In: SOUTHERN ASSOCIATION FOR INFORMATION SYSTEMS CONFERENCE, 2006., 2006. **Proceedings...** [S.l.: s.n.], 2006. p. 251–258.

LIKERT, Rensis. A technique for the measurement of attitudes. **Archives of psychology**, [S.l.], 1932.

LYYTINEN, Kalle; TAHVANAINEN, Veli-Pekka. **Next Generation CASE Tools**. [S.l.]: IOS Press, 1992. v. 3.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. **Fundamentos de metodologia científica, Fundamentals of scientific methodology**. [S.l.]: Atlas, 2003.

MARTIN, James; FINKELSTEIN, Clive. **Information engineering**. [S.l.]: Savant Research Studies, 1981.

MARTIN, James; MCCLURE, Carma. **Diagramming techniques for analysts and programmers**. [S.l.]: Prentice-Hall, Inc., 1985.

MASÓ, J.S.; GESA, R.F.; MATEMÀTICA APLICADA, Universitat de Girona. Departament d'Informàtica i. **Entorno virtual para el aprendizaje y la evaluación automática en bases de datos**. 2010. Tese (Doutorado em Ciência da Computação) — Universitat de Girona, 2010.

MCIVER, J.; CARMINES, E.G. **Unidimensional Scaling**. [S.l.]: SAGE Publications, 1981. n. N° 24. (07).

MEC, Ministério da Educação. **Diretrizes Curriculares dos cursos de Bacharelado em Ciência da Computação, Engenharia de Computação, Engenharia de Software e Sistemas de Informação e dos cursos de Licenciatura em Computação**. [S.l.: s.n.], 2003. Disponível em: <<http://portal.mec.gov.br/index.php?Acesso>> Acesso em: 08 ago 2013.

MERRIAM, Sharan B. **Qualitative Research and Case Study Applications in Education. Revised and Expanded from "Case Study Research in Education."** [S.l.]: ERIC, 1998.

MOKKAPATI, Snigdha. **Static type checker tools for dart**. 2012. Tese (Doutorado em Ciência da Computação) — San Jose State University, 2012.

MOORE, M.G.; KEARSLEY, G. **Distance Education: a systems view of online learning**, 3rd ed.: a systems view of online learning. [S.l.]: Wadsworth Cengage Learning, 2011. (What's New in Education Series).

MÜLLER, Gilberto Irajá. **Banco de dados: modelo conceitual**. Disponível em: <<http://professor.unisinos.br/gimuller/2012-2/bd1/bd1aula1.ppt>>. Acesso em: 09 ago 2011.

MÜLLER, Gilberto Irajá. **Banco de dados: álgebra relacional**. Disponível em: <<http://professor.unisinos.br/gimuller/2012-2/bd1/bd1aula8.ppt>>. Acesso em: 09 ago 2011.

NOVA, Cristiane; ALVES, Lynn. Educação à distância: limites e possibilidades. **Alves L, Nova C, organizadoras. Educação à distância: uma nova concepção de aprendizado e interatividade**. São Paulo: Futura, [S.l.], p. 1–23, 2003.

ORACLE. **Oracle SQL Developer Data Modeler**. Disponível em: <<http://www.oracle.com/technetwork/developer-tools/datamodeler/downloads/index.html>>. Acesso em: 08 ago 2013.

PEREIRA, Juliana Alves. **Um protótipo de ambiente de apoio ao processo de ensino aprendizagem de banco de dados: módulo sql**. 2011. Tese (Doutorado em Ciência da Computação) — Universidade federal de Lavras, 2011.

PEREIRA, Juliana Alves; RESENDE, Antônio Maria Pereira. Uma análise dos ambientes de ensino de banco de dados. In: VIII SIMPOSIO BRASILEIRO DE SISTEMAS DE INFORMAÇ AO, 2012. **Anais...** [S.l.: s.n.], 2012. p. 755–766.

PEREIRA, Julio Cesar Rodrigues. **Análise de Dados Qualitativos**: estratégias metodológicas para as ciências da saúde humanas e sociais. [S.l.]: EDUSP, 1999.

PEREIRA, Otacílio J; SENA, Claudia Pinto Pereira; BITTENCOURT, Cleide Tavares; SANTOS, Gledston Carneiro da Silva; JESUS BISPO, Naize de. Um ambiente virtual de aprendizagem de banco de dados e sua contribuição para um curso de computação. , [S.l.], 2012.

PILGRIM, Mark. **HTML5**: up and running. [S.l.]: O'Reilly, 2010.

POOLET, Michelle A. **Comparative Review**: sizing up data modeling software. Disponível em:<<http://sqlmag.com/sql-server/comparative-review-sizing-data-modeling-software>>. Acesso em: 08 ago 2013.

RAMAKRISHNAN, Raghu; GEHRKE, Johannes. **Database Management Systems**. 3rd. ed. [S.l.]: McGraw-Hill Education, 2003. (McGraw-Hill higher education).

REGUERAS, Luisa M; VERDÚ, Elena; VERDÚ, María J; PÉREZ, MA; DE CASTRO, Juan P. E-learning Strategies to Support Databases Courses: a case study. In: FIRST INTERNATIONAL CONFERENCE ON TECHNOLOGY, TRAINING AND COMMUNICATION, 2007. **Anais...** [S.l.: s.n.], 2007.

ROB, P.; CORONEL, C. **Database Systems**: design: design, implementation, and management. [S.l.]: Course Technology (Thomson Learning), 2009.

SBC, Sociedade brasileira de computação. **Currículo de Referência da SBC para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia de Computação**. 2005.

SCHRAMM, Wilbur. **Notes on Case Studies of Instructional Media Projects [microform] / Wilbur Schramm**. [S.l.]: Distributed by ERIC Clearinghouse [Washington, D.C.], 1971. 43 p. p.

SILBERSCHATZ, A.; KORTH, H.F.; SUDARSHAN, S. **Sistema de banco de dados**. 5. ed. [S.l.]: Elsevier, 2006.

SILVA, Alberto Manuel Rodrigues da; VIDEIRA, Carlos Alberto Escalera. **UML, metodologias e ferramentas CASE**: linguagem de modelação uml, metodologias e ferramentas case na concepção e desenvolvimento de software. [S.l.: s.n.], 2001.

SILVA, Edna Lúcia da; MENEZES, Estera Muszkat. **Metodologia da pesquisa e elaboração de dissertação**. [S.l.: s.n.], 2001.

SIMSION, Graeme; WITT, Graham. **Data Modeling Essentials, Third Edition (Morgan Kaufmann Series in Data Management Systems) (The Morgan Kaufmann Series in Data Management Systems)**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

SOLER, Josep; BOADA, Imma; PRADOS, Ferran; POCH, Jordi; FABREGAT, Ramon. An automatic correction tool for relational Algebra Queries. In: **Computational Science and Its Applications–ICCSA 2007**. [S.l.]: Springer, 2007. p. 861–872.

SONG, Il-Yeol; EVANS, Mary; PARK, Eun K. A comparative analysis of entity-relationship diagrams. **Journal of Computer and Software Engineering**, [S.l.], v. 3, n. 4, p. 427–459, 1995.

STAKE, R.E. **The Art of Case Study Research**. [S.l.]: SAGE Publications, 1995.

SURAWEEERA, Pramuditha; MITROVIC, Antonija. KERMIT: a constraint-based tutor for database modeling. In: INTELLIGENT TUTORING SYSTEMS, 2002. **Anais...** [S.l.: s.n.], 2002. p. 377–387.

SYBASE, SAP. **SAP Sybase PowerDesigner**. Disponível em:<<http://www.sybase.com.br/products/modelingdevelopment/powerdesigner>>. Acesso em: 08 ago 2013.

TEOREY, T.J.; LIGHTSTONE, S.S.; NADEAU, T.; JAGADISH, H.V. **Database Modeling and Design**: logical design, fourth edition. [S.l.]: Elsevier Science, 2010. (The Morgan Kaufmann Series in Data Management Systems).

TIGHT, Malcolm. Defining distance education. **ICDE Bulletin**, [S.l.], v. 18, p. 56–60, 1988.

TSICHRITZIS, Dennis; KLUG, Anthony. The ANSI/X3/SPARC DBMS framework report of the study group on database management systems. **Information systems**, [S.l.], v. 3, n. 3, p. 173–191, 1978.

VASQUES, Monica Heloisa B. **Metodologia de pesquisa científica**. [S.l.]: Uninove, 2008.

W3C. **Canvas Element**. Disponível em:<<http://www.w3.org/TR/2009/WD-html5-20090825/the-canvas-element.html>>. Acesso em: 08 ago 2013.

WAINER, Jacques. Métodos de pesquisa quantitativa e qualitativa para a Ciência da Computação. **Atualização em Informática**. Org: Tomasz Kowaltowski; Karin Breitman. Rio de Janeiro: Ed. PUC-Rio, [S.l.], 2007.

WORLD, Toad. **Toad Data Modeler Community**. Disponível em:<<http://www.toadworld.com/products/toad-data-modeler/f/30/t/2813.aspx>>. Acesso em: 08 ago 2013.

YIN, Robert K. **Estudo de caso**. [S.l.]: Bookman, 2005.