JULY ANY MARTINEZ DE RIZZO

AVALIAÇÃO DE PADRÕES PARA IMPLEMENTAÇÃO DE MODELOS DE DADOS ORIENTADOS A OBJETOS EM BANCOS DE DADOS RELACIONAIS

Dissertação de mestrado apresentada ao Programa de Pós–Graduação de Engenharia Elétrica / Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a obtenção do título de mestre em Engenharia Elétrica.

Área de Concentração: Sistemas Digitais

Orientador: Prof. Dr. Jorge Rady de Almeida Jr.

São Paulo

2010

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, de dezembro de 2010.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Rizzo, July Any Martinez De
Avaliação de padrões para implementação de modelos de dados orientados a objetos em banco de dados relacionais / J.A.M. De Rizzo. -- ed.rev. -- São Paulo, 2010.
100 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Banco de dados relacionais (Implantação) 2. Baco de dados orientado a objetos 3. Modelagem de dados 4. Métricas de software I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II. t.

DEDICATÓRIA

Dedico esta dissertação à minha irmã.

AGRADECIMENTOS

Ao Professor Dr. Jorge Rady de Almeida Jr. pelo crédito e empenho em mim depositados.

Aos professores que me ajudaram na caminhada rumo ao aprendizado de Banco de Dados e ao gosto pela atividade de pesquisa, em especial a Prof. Dra. Rosângela Ap. Dellosso Penteado, a Prof. Dra. Marilde Terezinha Prado Santos, ao Prof. Dr. Mauro Biajiz (*in memorian*), ao Prof. Dr. Jander Moreira, a Prof. Dra. Solange Nice Alves de Souza e ao Prof. Dr. Sidney da Silva Viana.

A todos os amigos e colegas que direta ou indiretamente me apoiaram e aos que colaboraram com dicas, conselhos e idéias para a execução deste trabalho.

Aos meus pais.

RESUMO

A questão da implementação de modelos de dados que utilizam a orientação a objetos constitui-se, ainda, em um assunto não totalmente consolidado. Dessa forma, nesta dissertação realiza-se uma sistematização relativa à implementação de um banco de dados relacional representado por um diagrama de classes. Este trabalho apresenta como foco principal uma avaliação de métricas do mapeamento de três tipos de relacionamento de um modelo orientado a objetos, Herança, Agregação/Composição e Associação, quando aplicados a um modelo relacional de banco de dados. Para isso, foram avaliados ao todo sete padrões de mapeamento desses relacionamentos para a modelagem relacional, sendo dois padrões de Herança, dois de Agregação e dois de Associação, além de análise de estudos empíricos relacionados ao tema. Ambas as formas de modelagem, relacional e orientada a objetos, são compatíveis quando analisadas suas modelagens conceituais. Assim, avalia-se a adequação da implementação dos modelos orientados a objetos em um banco de dados relacional após a aplicação dos padrões de mapeamento. Como resultado deste trabalho, é apresentada uma proposta de análise de métricas da aplicação dos padrões de mapeamento em um modelo apropriado para implementação em um banco de dados relacional. Algumas das métricas avaliadas são desnormalização, método de armazenamento lógico alinhado à estratégia de indexação, alta disponibilidade e uso de métodos de replicação, custo de acesso a dados, espaço em disco e flexibilidade e custo de manutenção.

Palavras chaves: Banco de dados, padrões, mapeamento, modelagem, implementação, métricas de avaliação.

ABSTRACT

Implementation of object-oriented data models constitutes in a not fully consolidated subject yet. Thus, this work performs an evaluation about a relational database implementation represented by a class diagram. The main focus of this paper is to present a systematic metric evaluation for the mapping of three relationships types of an object-oriented model, Inheritance, Aggregation / Composition and Association, when applied to a relational database model. For this purpose, seven mapping patterns that transform these relationships into a relational model notation were evaluated, two patterns of Inheritance, two of Aggregation, and two of Association, besides the analysis of empirical studies related to the topic. Both forms of modeling, relational and object-oriented, are considered compatible when their conceptual modeling is analyzed. So this paper evaluates the adequation of the object-oriented models implementation in a relational database after the appliance of the mapping standards. As a result of this work, it is presented an analysis of metrics proposal from the mapping patterns application in a suitable model for implementation in a relational database. Some of the evaluated metrics are denormalization, logical storage method aligned to indexing strategy, high availability and use of replication methods, cost of access to data, disk space and flexibility and maintenance costs.

Keywords: Database, patterns, mapping, modeling, implementation, evaluation metrics.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

BD: Banco de Dados

BLOB: *Binary Large Object*

CPU: *Central Processing Unit*

CRUD: *Create, Read, Update and Delete*

ER: Entidade-Relacionamento

ISAM: *Indexed Sequential Access Method*

ISBN: *International Standard Book Number*

OID: *Object Identificator* (Identificador do objeto)

OO: Orientado a Objeto

UML: *Unified Modeling Language*

RAID: *Redundant Array of Independent Disks*

SBD: Sistema de Banco de Dados

SGBD: Sistema Gerenciador de Banco de Dados

XML: *Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

O modelo orientado a objetos e o modelo relacional representam paradigmas de desenvolvimento diferentes, sendo assim, não totalmente compatíveis. O modelo relacional foi criado por Edgar Frank Codd em 1970 e é um modelo de dados baseado em lógica e na teoria de conjuntos. A principal proposição do modelo relacional é que todos os dados são representados como relações matemáticas, isto é, um subconjunto do produto cartesiano de *n* conjuntos, compondo diversos conjuntos. Enquanto isso, o modelo orientado a objetos organiza os sistemas como uma coleção de objetos que integram estruturas de dados e comportamentos do sistema.

Atualmente, o modelo relacional apresenta larga difusão no ambiente de tecnologia da informação, constituindo-se em um modelo tecnológico estratégico de controle de informação. Por outro lado, o ambiente de tecnologia da informação requer soluções para sistemas com comportamentos complexos e específicos, nos quais o processo com seus métodos e validações tem importância destacada e os dados assumem caráter secundário. Por esta lógica, os dados devem ser acessados por meio de navegação, em uma hierarquia natural, e com base nas representações de negócio.

Neste contexto é que o modelo de orientação a objetos assume status de modelo mais adequado para o desenvolvimento de aplicações empresariais, e o modelo relacional surge como uma estrutura de dados que precisa ser integrada com o domínio das aplicações orientadas a objetos.

Estudos como os realizados em (YUGOPUSPITO, 99) dizem que a diferença da modelagem de um Banco de Dados Relacional para a de um Banco de Dados Orientado a Objetos pode ser definida por um processo de Transformação somado a um processo de Adição de informação, como ilustra a **Figura 1**.

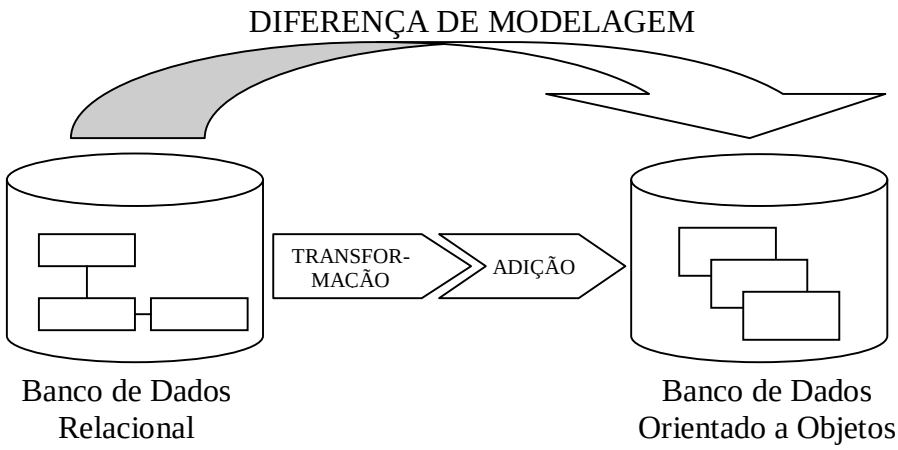

Figura 1. Modelagem em Bancos de Dados Relacional e OO

A Transformação consiste em transpor a forma de representação, sem que isso leve à perda ou distorção de informação. Por exemplo, a relação entre duas entidades é chamada de relacionamento, que é ilustrado na **Figura 2**.

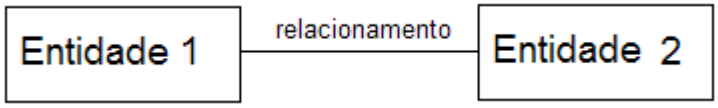

Figura 2. Relacionamento

Por sua vez, o diagrama de classe da UML tem vários tipos de relacionamentos entre duas classes, como Associação, Agregação, Composição e Herança (também chamada Generalização ou Hierarquia), ilustrados na **Figura 3**. Estas conexões também são chamadas de relacionamentos (OMG, 99) e é este o processo de Transformação visto anteriormente.

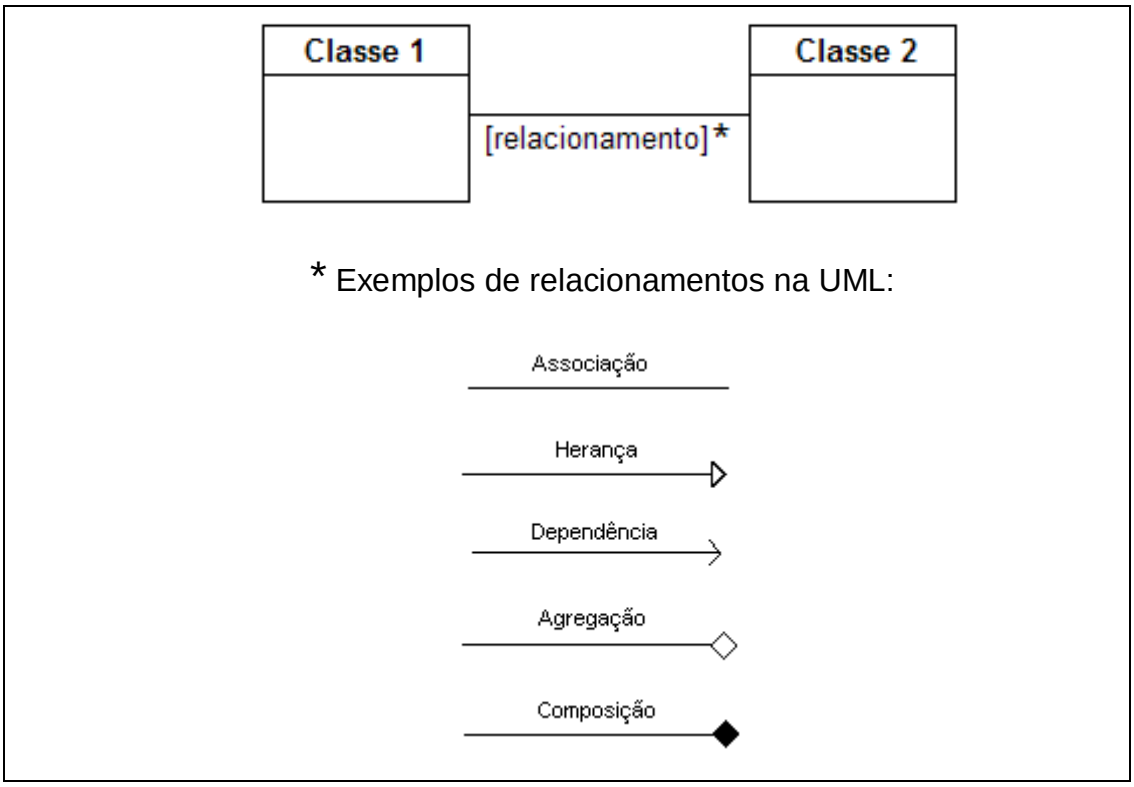

Figura 3. Relacionamento na UML

A Adição representa todos os componentes que não existem nos modelos relacionais, mas que são definições integradas do diagrama UML, isto é, informações adicionais que devem ser adicionadas ao modelo de dados.

Mapear objetos em tabelas é um problema que tem ocorrido desde que as pessoas querem programar em linguagem orientada a objeto, mas têm que usar bancos de dados relacionais, em lugar de bancos de dados orientados a objetos, devido a problemas como linguagem de acesso, interoperabilidade ou falta de suporte de SGBDs. Mapear objetos em tabelas é apenas um grupo de questões que ocorre nas camadas de acesso objeto/relacional.

1.1. Justificativa

Os projetos de desenvolvimento de aplicações de negócios mais modernos utilizam a tecnologia de objetos, como Java ou C# para compilar o aplicativo de software e bancos de dados relacionais para armazenar os dados. Isso não quer dizer que não haja outras opções. Aplicações construídas com linguagens procedurais e sistemas que utilizam bancos de dados orientados a objeto ou XML para armazenar dados também são utilizados. No entanto, tecnologias relacionais e orientadas a objeto são as mais reconhecidas e trabalhadas no âmbito de bancos de dados.

Há uma diferença de impedância entre as tecnologias relacional e de objeto, tecnologias estas que as equipes de projeto geralmente usam para construir sistemas de software. Scott Ambler (AMBLER, 03) considera simples superar esta diferença de impedância, sendo preciso para isto, compreender o processo de mapeamento de objetos para bancos de dados relacionais e entender como implementar esses mapeamentos. Considerar simples esta compreensão não significa que a aplicação do método seja fácil.

Essa impedância, que é descrita por alguns autores, é analisada neste trabalho.

1.2. Objetivos do Trabalho

O objetivo principal deste trabalho é analisar os padrões de mapeamento de objetos em tabelas e os relacionamentos entre elas, em um banco de dados relacional. Para se atingir tal objetivo, foram estudados alguns padrões de mapeamento de Herança, Agregação e Associações.

Visando atingir esse objetivo, pode-se afirmar que outros objetivos parciais desta dissertação são:

- Definição das relações entre classes da orientação a objetos a serem mapeadas em tabelas, bem como o detalhamento de padrões que fazem mapeamentos de modelos OO para relacional;

- Determinação de métricas a serem avaliadas para cada um desses padrões e a respectiva análise das métricas para cada padrão;
- Proposta de análise dos padrões conforme as métricas; e
- Mapeamento de segmentos de modelos orientados a objetos em um banco de dados relacional, utilizando-se esses padrões e a avaliação dos mesmos conforme a proposta.

1.3. Conteúdo do Trabalho

O tema central desta dissertação consiste na avaliação de alguns padrões que transformam e mapeiam relacionamentos de modelos orientados a objetos em modelos relacionais, adequados para implementação de bancos de dados relacionais. Esta dissertação está dividida em cinco Capítulos:

O Capítulo 2, Padrões de Mapeamento, detalha três padrões de mapeamento de Herança, dois padrões para mapeamento de Agregação e dois padrões para mapeamento de Associações. A descrição de alguns trabalhos relacionados ao tema desta dissertação também está presente neste Capítulo.

O Capítulo 3, Métricas de Avaliação, apresenta a proposta desta dissertação e a metodologia utilizada para execução do caso prático.

O Capítulo 4, Estudo de Caso, demonstra a aplicação da proposta teórica de avaliação de métricas para aplicação de padrões através de casos práticos de modelagem de bancos de dados, utilizando-se os padrões descritos e aplicando-se a análise de métricas proposta no Capítulo anterior.

No Capítulo 5 são detalhadas as Conclusões desta dissertação bem como suas contribuições e as pesquisas e trabalhos futuros a serem realizados.

2 PADRÕES DE MAPEAMENTO

Neste Capítulo são detalhados alguns padrões de mapeamento, assunto diretamente ligado ao tema da dissertação.

Os padrões de projeto de software representam problemas e as soluções para esses problemas de uma maneira formal. Eles foram inspirados por um livro escrito pelo arquiteto Christopher Alexander, que diz que "cada padrão descreve um problema que ocorre repetidamente em nosso meio, e então descreve o núcleo da solução para esse problema, de tal forma que se possa usar essa solução um milhão de vezes, sem nunca fazê-lo da mesma maneira duas vezes" (ALEXANDER ET AL. 77).

No mundo do software, os padrões foram introduzidos pelo livro "Design Patterns: Elements of Reusable Object-Oriented Software", (GAMMA ET AL., 94), que descreve 23 padrões de projeto subdivididos pelos seus propósitos: de criação, estrutural ou comportamental.

Em geral, um padrão tem quatro elementos essenciais: o nome do padrão, o problema, a solução e as consequências.

O **nome do padrão** é um identificador que pode ser utilizado para descrever um problema de projeto, suas soluções e consequências, em uma ou duas palavras.

O **problema** descreve quando aplicar o padrão, explicando o problema e seu contexto.

A **solução** descreve os elementos que compõem o projeto, seus relacionamentos, responsabilidades e colaborações.

As **consequências** são os resultados e os acertos da aplicação do padrão. As consequências são críticas para se avaliar alternativas de projeto e para a compreensão dos custos e benefícios da aplicação do padrão. As consequências para o software, muitas vezes, ocupam espaço e tempo de correção e também têm impacto sobre a flexibilidade de um sistema, extensibilidade e portabilidade. É neste item que se pode avaliar a qualidade da aplicação do padrão de mapeamento de objetos no modelo relacional.

A seguir são descritos padrões de mapeamento de objetos em tabelas do modelo relacional, conforme os quatro elementos essenciais descritos

anteriormente. Estes padrões estão definidos nos trabalhos de Fowler (FOWLER, 02), Keller (KELLER, 97) e nas idéias detalhadas por Scott Ambler (AMBLER, 97).

A descrição dos padrões neste Capítulo limita-se aos três primeiros elementos essenciais: nome do padrão, problema e solução. As consequências são definidas e detalhadas no Capítulo 3 desta dissertação, fazendo parte da Proposta deste trabalho.

2.1. Padrões para mapeamento de Herança

A seguir são descritos três padrões de mapeamento de Herança.

2.1.1. Herança de Classes em Tabela Única

Nome do Padrão: **Herança de Classes em Tabela Única**, também definido como **Single Table Inheritance** (FOWLER, 02) e **One Inheritance Tree One Table** (KELLER, 97).

Problema: Como mapear uma hierarquia de herança de classes para tabelas em um banco de dados relacional.

Solução:

Usar a união de todos os atributos de todas as classes da hierarquia da herança como colunas em uma tabela única. Utilizam-se valores nulos para preencher campos não utilizados em cada registro.

Quando se carrega um objeto em memória, é necessário conhecer qual classe instanciar. Para isto, é necessário um campo na tabela que indique que classe deve ser utilizada, podendo ser o nome da classe ou um código numérico. Este campo deve ser interpretado previamente, antes da instanciação da classe e armazenado corretamente em cada construtor.

A **Figura 4** ilustra uma herança entre classe de jogadores. Um jogador de boliche (JogBoliche) herda as características da classe de jogadores de críquete (JogCríquete), que por sua vez herda as características da classe base, Jogador. A classe de jogadores de futebol (JogFutebol) também herda características da classe base. A aplicação do padrão “Herança de Classes em Tabela Única” faz com que

todos os atributos das classes envolvidas na herança sejam mapeados na única tabela criada. O campo “tipo” da tabela “Jogador” é o campo que indica a classe a ser instanciada, como relatado no parágrafo anterior.

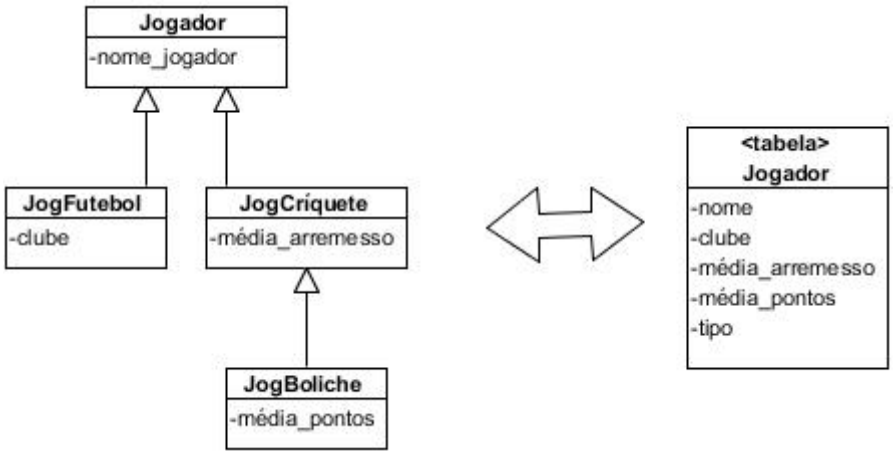

Figura 4. Aplicação do Padrão: Herança de Classes em Tabela Única

Ambler chama a aplicação do mapeamento descrito por este padrão de estratégia, e dá a ela o nome de **Map Hierarchy To A Single Table** (AMBLER, 97).

2.1.2. Herança de Classes em Tabelas para cada Classe

Nome do Padrão: **Herança de Classes em Tabelas para cada Classe**, também definido como **Class Table Inheritance** (FOWLER, 02), **Root-Leaf Mapping** (BROWN, 03) e **One Class One Table** (KELLER, 97).

Problema: Como mapear uma hierarquia de herança de classes para tabelas em um banco de dados relacional.

Solução:

Mapear os atributos de cada classe para tabelas separadas. Uma das questões é de como associar as linhas correspondentes das tabelas do banco de dados. Uma possível solução é utilizar uma chave primária comum para unir as linhas das tabelas correspondentes.

A **Figura 5** ilustra uma herança entre classe de jogadores. Um jogador de boliche (JogBoliche) herda as características da classe de jogadores de críquete (JogCríquete), que por sua vez herda as características da classe base (Jogador). A classe de jogadores de futebol (JogFutebol) também herda características da classe base. A aplicação do padrão “Herança de Classes em Tabelas para cada Classe” faz com que cada classe seja mapeada em uma tabela, desta maneira, as quatro classes envolvidas na herança são mapeadas em quatro tabelas distintas. Os atributos de cada classe ficam mapeados nas respectivas tabelas criadas.

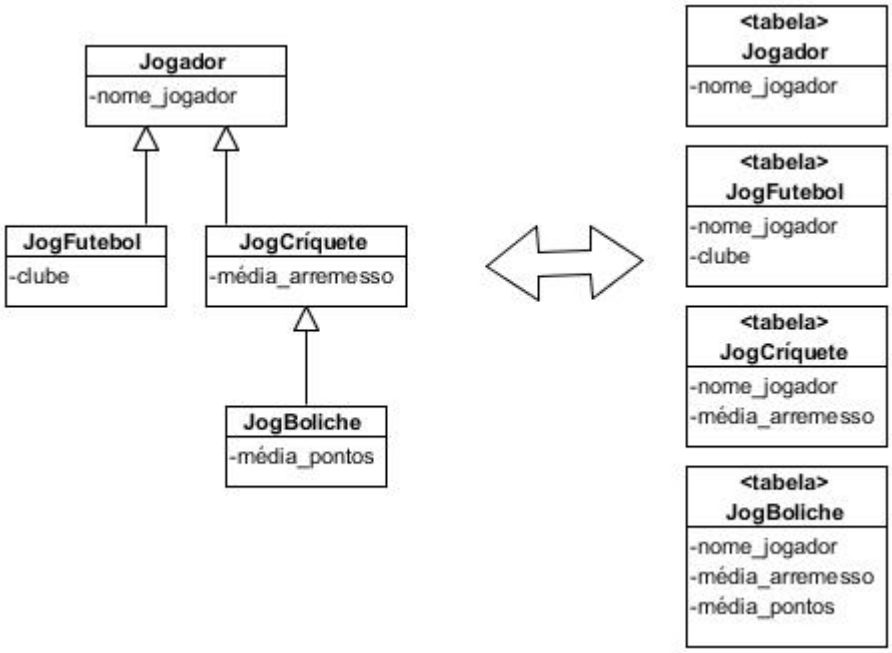

Figura 5. Aplicação do Padrão: Herança de Classes em Tabelas para cada Classe

Ambler chama a aplicação do mapeamento descrito por este padrão de estratégia, e dá a ela o nome de **Map Each Class To Its Own Table** (AMBLER, 97).

2.1.3. Herança de Classes em Tabelas para cada Classe Concreta

Nome do Padrão: **Herança de Classes em Tabelas para cada Classe Concreta**, também definido como **Concrete Table Inheritance** (FOWLER, 02) e **One Inheritance Path One Table** (KELLER, 97).

Problema: Como mapear uma hierarquia de herança de classes para tabelas em um banco de dados relacional.

Solução:

Mapear os atributos de cada classe para uma tabela separada. Para cada tabela, adicionar todos os atributos de todas as superclasses que a classe em questão herda. Uma questão a ser analisada diz respeito aos identificadores das tabelas. Neste caso deve-se considerar que a chave não é única apenas para a tabela, e sim para todo o contexto da hierarquia. Esta questão pode ser endereçada através de padrões de identidade. É necessário lembrar que o mecanismo de alocação de chaves primárias dos bancos de dados relacionais considera a unicidade apenas no contexto de uma tabela.

A **Figura 6** ilustra uma herança entre classe de jogadores. Um jogador de boliche (JogBoliche) herda as características da classe de jogadores de críquete (JogCríquete), que por sua vez herda as características da classe base (Jogador). A classe de jogadores de futebol (JogFutebol) também herda características da classe base. A aplicação do padrão “Herança de Classes em Tabelas para cada Classe Concreta” faz com que cada classe concreta seja mapeada em uma tabela. Neste contexto, a classe base é considerada uma classe abstrata. Desta maneira, as três classes que herdam as características da classe base são mapeadas em três tabelas distintas. Os atributos de cada classe ficam mapeados nas respectivas tabelas criadas e os atributos da classe base são replicados em todas as tabelas criadas.

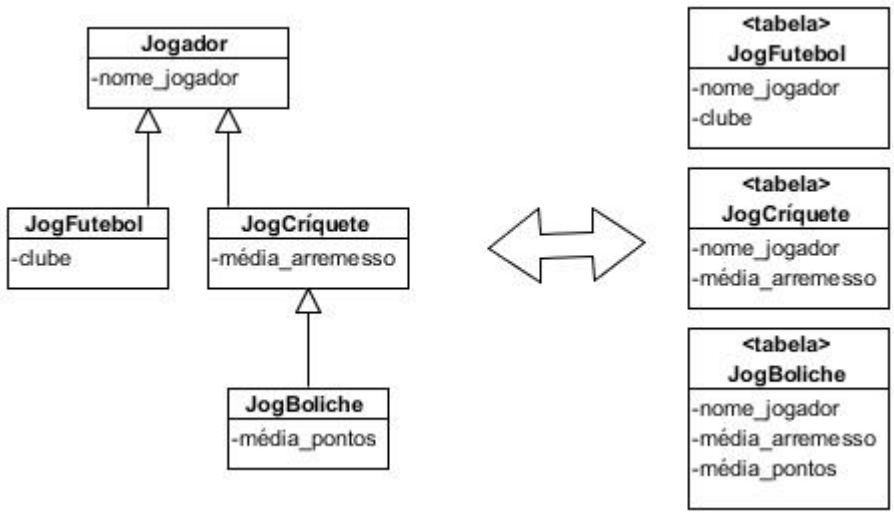

Figura 6. Aplicação do Padrão: Herança de Classes em Tabelas para cada Classe Concreta

Ambler chama a aplicação do mapeamento descrito por este padrão de estratégia, e dá a ela o nome de **Map Each Concrete Class To Its Own Table** (AMBLER, 97).

2.2. Padrões para mapeamento de Agregação

A seguir são descritos dois padrões de mapeamento de Agregação.

2.2.1. Agregação em Tabela Única

Nome do Padrão: **Agregação em Tabela Única**, também definido como **Single Table Aggregation** (KELLER, 97).

Problema: Como mapear uma agregação para tabelas em um banco de dados relacional.

Solução:

Tanto os atributos do objeto agregador quanto o objeto agregado são transformados em uma tabela do modelo físico de dados e os atributos do objeto agregado são integrados a esta tabela.

A **Figura 7** ilustra uma agregação entre a classe de clientes (Cliente) e a classe de tipo de endereço (tipoEndereço). O cliente tem um endereço de fatura e um endereço de entrega. Estes atributos referenciam a classe tipo de endereço, agregada à classe cliente. Com a aplicação do padrão “Agregação em Tabela Única” é criada uma única tabela, com todos os atributos da classe agregadora, exceto aqueles que fazem referência a classes agregadas, pois neste caso, substituem-se estes atributos por todos os atributos da classe agregada. Desta maneira, o endereço de fatura é substituído pelos atributos da classe tipo de endereço.

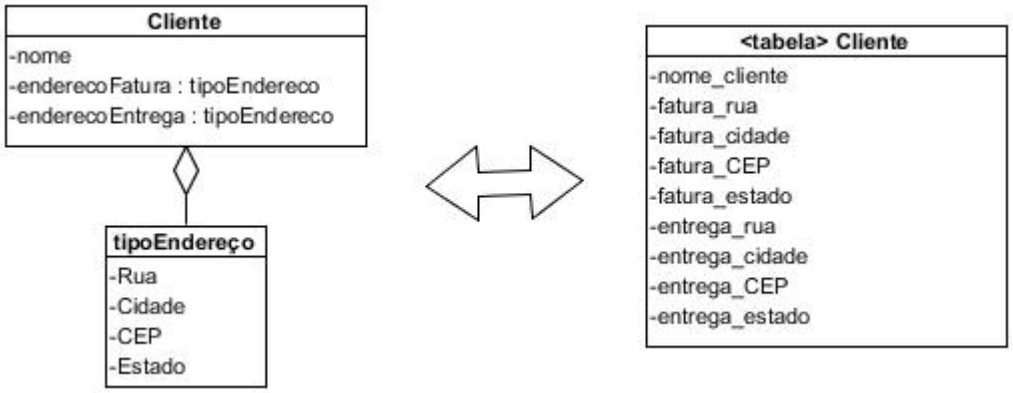

Figura 7. Aplicação do Padrão: Agregação em Tabela Única

2.2.2. Agregação em Chave Estrangeira

Nome do Padrão: **Agregação em Chave Estrangeira**, também definido como **Foreign Key Aggregation** (KELLER, 97).

Problema: Como mapear uma agregação para tabelas em um banco de dados relacional.

Solução:

Criar uma tabela para cada classe do tipo agregado. Inserir um “Identificador de Objeto Sintético” na tabela e usar esta identidade de objeto na tabela do objeto agregador para fazer uma ligação da chave estrangeira para o objeto agregado.

A **Figura 8** ilustra uma agregação entre classe de clientes (Cliente) e a classe de tipo de endereço (tipoEndereço). O cliente tem um endereço de fatura e um endereço de entrega. Estes atributos referenciam a classe tipo de endereço, agregada à classe cliente. Com a aplicação do padrão “Agregação em Chave Estrangeira” são criadas duas tabelas, uma para cada classe. Na tabela referente à classe agregada é criada uma chave para identificação do registro, tal como um OID, identificador de objeto, na modelagem OO. A tabela do objeto agregador, por sua vez, recebe um campo com esta mesma identificação para fazer uma associação da chave estrangeira para o objeto agregado.

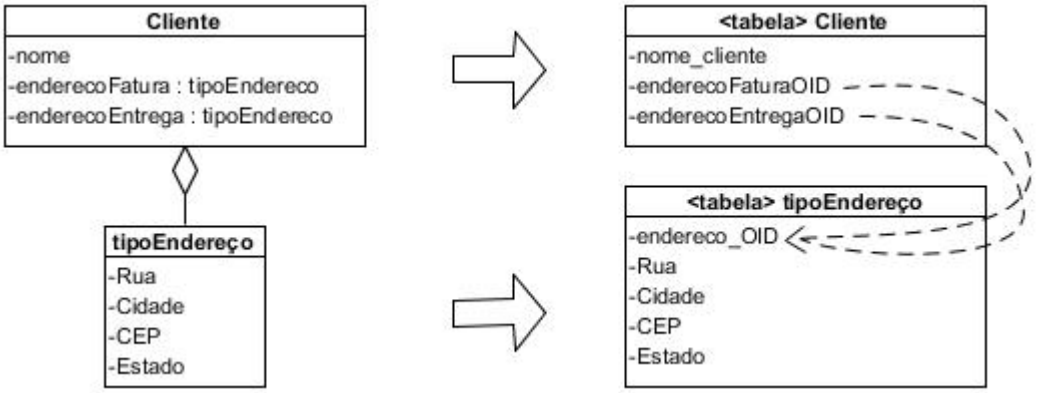

Figura 8. Aplicação do Padrão: Agregação em Chave Estrangeira

2.3. Padrões para mapeamento de Associações

A seguir são descritos dois padrões de mapeamento de Associações.

2.3.1. Associação 1:n em Chave Estrangeira

Nome do Padrão: **Associação 1:n em Chave Estrangeira**, também definido como **Foreign Key Mapping** (FOWLER, 02) e **Foreign Key Association** (KELLER, 97).

Problema: Como mapear uma associação de objetos para tabelas do modelo relacional.

Solução:

Deve-se considerar que cada objeto possui uma chave do banco de dados na tabela apropriada, e que os objetos estão representados em tabelas independentes, quando dois objetos estão relacionados por uma associação. Esta associação pode ser mapeada como uma chave estrangeira no banco de dados. A implementação significa inserir o identificador da chave do objeto proprietário, como uma chave estrangeira na tabela do objeto dependente.

A **Figura 9** ilustra uma associação 1:n entre as classes de Artista (Artista) e Álbum (Álbum). Assim, um artista grava vários álbuns, e vários álbuns são gravados por um artista. Com a aplicação do padrão “Associação 1:n em Chave Estrangeira”, cada classe é mapeada em uma tabela e cada tabela recebe um atributo de identificação (um número inteiro que identifica unicamente um registro da mesma). A

tabela Álbum, mapeada para o lado n da associação, recebe um novo atributo, ou seja, a chave estrangeira, responsável por identificar a qual artista aquele álbum pertence.

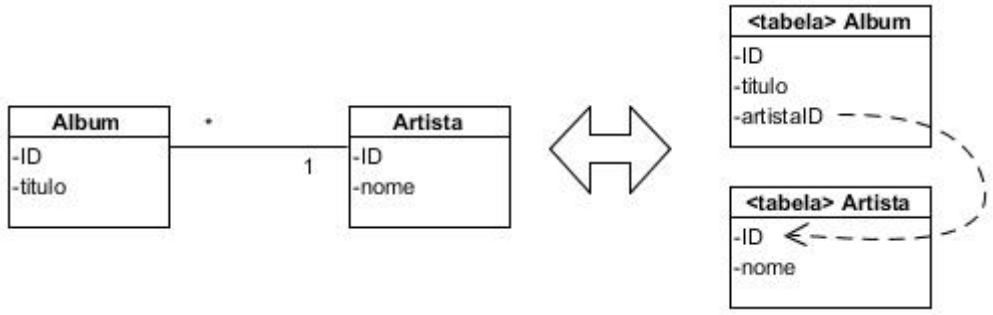

Figura 9. Aplicação do Padrão: Associação 1:n em Chave Estrangeira

Ambler chama a aplicação do mapeamento descrito por este padrão de estratégia, e dá a ela o nome de **One-To-Many Mappings** (AMBLER,97).

2.3.2. Associação m:n em Tabela

Nome do Padrão: **Associação m:n em Tabela**, também definido como **Association Table Mapping** (FOWLER, 02) e **Association Table** (KELLER, 97).

Problema: Como mapear associações n:m para tabelas relacionais.

Solução:

A idéia básica por trás deste padrão é utilizar uma tabela de ligação para armazenar a associação. Esta tabela tem somente os identificadores das chaves estrangeiras para as duas tabelas que são associadas, e tem uma linha para cada par de objetos associados.

A **Figura 10** ilustra uma associação m:n entre as classes de Empregado e Habilidade, assim, um empregado contém várias habilidades, e uma habilidade pode estar presente em diversos empregados. Com a aplicação do padrão “Associação m:n em Tabela” são criadas duas tabelas, uma para cada classe envolvida na relação, cada uma com seu atributo de identificação única de um registro, a chave primária. Para fazer o relacionamento entre as tabelas, uma terceira tabela é criada e nela são inseridas as chaves primárias das duas tabelas relacionadas.

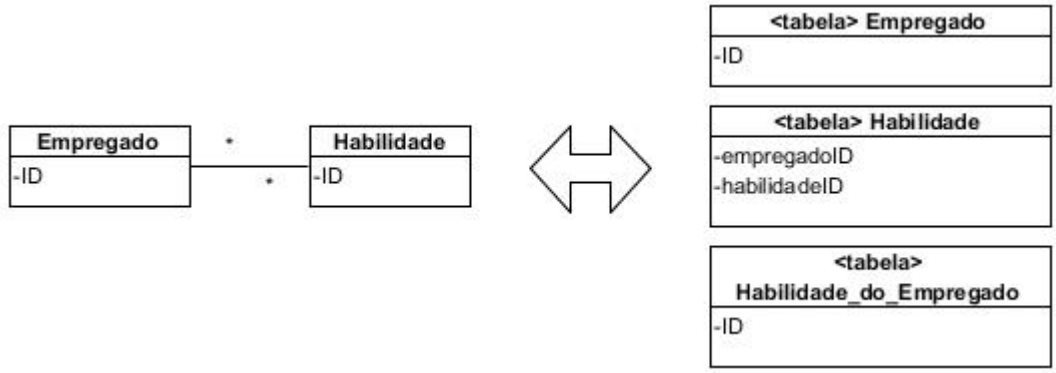

Figura 10. Aplicação do Padrão: Associação m:n em Tabela

Ambler chama a aplicação do mapeamento descrito por este padrão de estratégia, e dá a ela o nome de **Many-To-Many Mappings** (AMBLER,97).

2.4. Trabalhos Relacionados

Nos últimos 20 anos, diversos trabalhos vêm sendo realizados para analisar as notações gráficas que apóiam o processo de desenvolvimento de software. Essas análises são feitas por meio de experimentos, estudos empíricos ou pesquisas. Dentre eles, alguns autores analisaram e compararam o modelo Entidade-Relacionamento (ER) e suas extensões e o modelo Orientado a Objetos (OO), tais como (BOCK, 93), (PALVIA, 92), (LIAO, 00), (SHOVAL, 94), (SHOVAL, 97), (SINHA, 99), (CHAN, 05), (LEE, 98), (DE LUCIA, 08), (AGUIRRE-URRETA, 08).

Ambler (AMBLER, 97) não descreve as estratégias de mapeamento em forma de padrão (nome, problema, solução, consequência). Porém, a semelhança das estratégias de Ambler com os padrões formalizados por outros autores, citados nos itens 2.1, 2.2. e 2.3, está no fato de terem um embasamento teórico com aplicação de conceitos rígidos, diferentemente do que ocorre nas referências desta seção, que representam estudos realizados de forma empírica e cujos alguns resultados são relatados a seguir.

O trabalho de Bock e Ryan (BOCK, 93) tem como objetivo estudar a acurácia em modelos entidade relacionamento estendidos e modelagem de dados orientados a objetos. Para isso, foi feita uma análise com estudantes graduados e estudantes do último ano da faculdade inscritos em uma aula de Banco de Dados. Foi fornecida

aos alunos uma narrativa de uma situação empresarial de estruturas de dados que poderiam ser derivados e modelados. Segundo o experimento, os alunos tiveram mais facilidades em modelar o cenário utilizando-se a modelagem entidade-relacionamento estendida do que a orientação a objetos. Foram analisados os seguintes aspectos: correção na modelagem do identificador do objeto (OID) ou entidade, do relacionamento unário 1:1 e do relacionamento binário m:n.

O trabalho de Shoval e Shiran (SHOVAL, 97) tem como objetivo comparar modelos entidade relacionamento estendido e modelagem de dados orientado a objetos com relação à qualidade do modelo gerado. Foi fornecida, a estudantes avançados de turmas de Sistema de Informação, uma narrativa de uma situação empresarial de estruturas de dados que deveriam de modeladas. Pela exatidão do modelo, os autores mostraram que a modelagem entidade relacionamento estendida foi mais bem realizada pelos indivíduos quando comparada à modelagem orientada a objetos nos quesitos relacionamento unário 1:1, relacionamento ternário 1:m:n e relacionamento ternário m:n:o. Quando analisada a preferência dos indivíduos, a modelagem entidade-relacionamento estendida também levou vantagem. Quando comparado o tempo demandado para execução da tarefa os autores também notaram que este era menor quando o sistema era modelado em notação entidade-relacionamento.

Segundo Lee e Choi (LEE, 98) a análise comparativa foi feita com estudantes graduados e não graduados, que foram avaliados conforme a acurácia do modelo gerado dada uma narrativa ou um formulário de uma situação empresarial, tal como os trabalhos anteriormente citados. Os indíviduos foram separados em quatro grupos: alunos sem conhecimento de banco de dados dotados de uma narrativa de um cenário de negócio; alunos graduados, com conhecimento de banco de dados, dotados de uma narrativa de um cenário de negócio; alunos sem conhecimento de banco de dados dotados de um formulário descrevendo um cenário de negócio; alunos graduados, com conhecimento de banco de dados, dotados de um formulário descrevendo um cenário de negócio. Em seu trabalho, que tinha como objetivo ser um estudo comparativo entre diferentes técnicas de modelagem de dados conceitual, Lee e Choi concluíram que independentemente do nível de conhecimento dos alunos e graduados e independentemente da forma como a especificação é dada, os grupos tiveram comportamento semelhante. A modelagem

entidade-relacionamento estendida se mostrou mais eficiente e efetiva quando comparados os aspectos de mapeamento entidade/objeto, relacionamento binário 1:m, relacionamento binário m:n, relacionamento ternário m:n:o, relacionamento binário 1:1 e generalização. A facilidade de uso analisada também foi mais alta quando se referiu à modelagem entidade-relacionamento estendida, bem como o tempo de modelagem que fora menor.

Outros trabalhos também utilizaram métodos empíricos para comparar as diferentes modelagens, como o caso do trabalho de Liao e Palvia (LIAO, 00) que analisaram o impacto dos modelos de dados e a complexidade das tarefas no desempenho de analistas que necessitam realizar a tarefa de modelagem. Os autores avaliam com base na eficiência do modelo gerado pelos indivíduos que o modelo entidade-relacionamento estendido é melhor. O trabalho de Chan et al (CHAN, 05), por sua vez, avaliou o desempenho de usuários finais novatos em questão de modelagem de dados, escrita de consultas e compreensão do modelo, e com sua amostragem de indivíduos participantes na pesquisa concluiu que o entendimento da modelagem orientada a objetos é mais fácil para os novatos.

Como os próprios autores referenciam, um dos fatos dos trabalhos que avaliam a facilidade de uso e tempo para resolução do problema concluírem que a modelagem entidade-relacionamento estendida é melhor que a modelagem orientada a objetos deve-se ao fato de que a primeira é muito mais difundida e conhecida há mais tempo do que a última.

De Lucia et al (DE LUCIA, 08) avalia em seu trabalho o suporte a diagramas entidade-relacionamento e diagramas de classe utilizando-se UML durante as atividades de manutenção de modelos de dados. Os experimentos envolviam estudantes de graduação e pós-graduação, que tiveram que realizar tarefas de manutenção em modelos de dados representados por entidade-relacionamento e diagramas de classe UML. Os resultados mostraram que não há diferença significativa entre os resultados obtidos. Além disso, os autores puderam realizar algumas considerações com relação à experiência e habilidade após os experimentos, dentre elas a de que a experiência não influenciara no nível de acurácia das atividades de manutenção, enquanto os indivíduos com alto nível de

habilidade conseguiram resultados significativamente melhores do que os indivíduos com baixa habilidade quando diagramas de classe UML eram utilizados.

No Capítulo a seguir é descrita a proposta deste trabalho.

3 MÉTRICAS DE AVALIAÇÃO

Cada um dos relacionamentos de herança, agregação/composição e associação pode ser mapeado em tabelas no modelo relacional, usando diferentes soluções. No Capítulo 2 foram descritas algumas destas soluções já consolidadas como padrões, ou seja, soluções distintas para o mesmo conceito são definidas e descritas na forma de diferentes padrões. Os mesmos foram descritos em três dos seus quatro elementos essenciais, ou seja, no Capítulo 2 definiu-se (1) o nome do padrão, (2) o problema e (3) a solução. Neste Capítulo são demonstradas as consequências do uso de cada solução com relação as métricas, isto é, o quarto elemento essencial da descrição de um padrão.

Algumas referências deste trabalho (FOWLER, 02), (KELLER, 97), (AMBLER, 97) fizeram análises utilizando-se métricas como desempenho, consumo de espaço em disco, flexibilidade, processamento de consultas e custo de manutenção. Estas foram selecionadas por estes autores por representar os maiores riscos de um projeto e fatores de análise naturais quando da implementação de um banco de dados. Para esta dissertação, são analisadas métricas de desempenho, levando-se em consideração fatores como modelagem, modo de armazenamento, indexação, replicação e processamento de consultas, bem como o espaço em disco consumido, com o enfoque nos tipos de dados e finalmente a flexibilidade, também retratada como custo de manutenção.

Não são considerados, nessa dissertação, fatores específicos de Sistemas de Bando de Dados (SBD) e configurações específicas de SGBDs de mercado que melhoram e aperfeiçoam o desempenho.

Antes de iniciar a análise dos padrões propriamente ditos, uma introdução sobre cada métrica é exposta a seguir.

3.1. Desempenho

O fator que mede o desempenho de um banco de dados, relacional ou não, é denominado tempo de resposta. Tempo de resposta é o tempo medido entre o início do processamento da informação e a respectiva efetivação desta, seja ela uma atividade de escrita, de atualização ou de leitura em um banco de dados. Este fator

é relativo, variando assim de sistema para sistema. Então, o tempo de resposta satisfatório para um sistema pode ser insatisfatório para outro.

Otimizar o desempenho requer cuidados múltiplos, ora por parte da readequação do modelo de dados, desnormalizando-o, se necessário, ora por parte da aplicação que acessa os dados, através da otimização das consultas, ora baseando-as em indicadores fornecidos pelo próprio SGBD tais como estatísticas de operação, adequação de índices e contenção de Entradas/Saídas, ora por armazenamento de consultas recentes. Outra forma nem sempre eficiente e mais dispendiosa, porém de mais simples implementação é a melhoria dos recursos de *hardware* através da otimização de processadores, aumento da memória física dedicada ao banco de dados, redes mais eficientes no que tange à transferência dos dados e discos magnéticos mais rápidos.

3.1.1. Desnormalização

Alguns problemas de desempenho podem ser resolvidos depois que o banco de dados estiver em produção, como recalcular índices ou definição de espaços alocáveis de memória ou otimização de consultas. No entanto, outros problemas de desempenho podem ser resultado de um mau projeto de banco de dados, só podendo ser resolvidos pela alteração de sua estrutura e projeto.

Ao projetar e implementar um banco de dados, é preciso identificar as tabelas grandes (na casa de milhões ou bilhões de registros) no banco de dados e os processos mais complexos que o banco de dados executará para poder fazer as consultas e a modelagem de forma otimizada. Além disso, o efeito no desempenho ao se aumentar o número de usuários que podem acessar o banco de dados concorrentemente também é uma questão a ser considerada.

Alterações de projeto que melhoram desempenho podem ser feitas conforme análise prévia das solicitações que serão feitas ao banco de dados. Por exemplo, se uma tabela que contém milhares ou milhões de linhas precisar ser resumida em um relatório diário, pode-se adicionar uma ou mais colunas que contenham os dados previamente agregados a serem usados apenas para o relatório.

Uma opção que deve ser considerada é a desnormalização, ou seja, não fazer uso das formas normais de maior nível, e sim transformar um modelo da terceira forma normal para a segunda ou da segunda para a primeira. No caso de processos complexos, esta ação pode melhorar o desempenho, já que o sistema não necessitará fazer processamento adicional com a união de diversas tabelas, por exemplo, para obtenção de um resultado que, no caso de uma tabela desnormalizada consumiria menos tempo.

Robin Schumacher, Diretor da Gestão de Produto do MySQL, escreveu em 2001 que bancos de dados rápidos não eram algo bom de ter, mas sim algo necessário (SEINER, 01). Quase dez anos depois esta mensagem continua atual e é para esta necessidade de rapidez que esta dissertação também se propõe a avaliar o desempenho de um modelo implementado em um banco de dados relacional modelado a partir de padrões. Algumas vezes o projeto de um banco de dados não é levado tanto em consideração, considerando a dificuldade de sua realização e também a questão de uma demanda maior de tempo. No entanto, uma modelagem adequada é muito importante, ainda mais no que se refere a bancos de dados que exijam alto desempenho.

3.1.2. Armazenamento Lógico e Indexação

O modo de armazenamento lógico dos dados, como arranjos, árvore binária, árvore binária de busca, árvore de busca binária auto-balanceada (ou balanceada pela altura), árvore B^+, ou tabelas hash influencia diretamente no acesso aos dados. Seja uma função de inserção, alteração ou exclusão de um registro, o tempo da atividade é dependente do modo de armazenamento, sua ordenação ou não e da estrutura de índices criada ou não. Bancos de dados ordenados e bancos de dados com índices têm um tempo menor de resposta de uma consulta, porém um tempo maior quando se trata de uma inserção ou alteração, considerando que o algoritmo que mantém a base ordenada e indexada deve ser executado após cada uma destas atividades.

A indexação é usada pelo banco de dados para agilizar o acesso aos dados. Uma estratégia que se pode utilizar para a indexação é indexar a coluna de chave

primária da tabela. As colunas de chave primária são frequentemente usadas como chaves de pesquisa e em operações de ligação.

No entanto, a indexação pode não ser útil em tabelas com centenas de linhas e poucas colunas, pois elas tendem a ser armazenadas na memória cache do banco de dados. Vale ressaltar que o fator de maior interesse está relacionado a quantas linhas estarão contidas em um bloco de dados, que, em geral, tem entre 512 e 2048 bytes de tamanho. A unidade de entradas e saídas executadas pelo SGBD não é uma linha, mas sim um bloco, quando observada por uma perspectiva física.

Os índices também devem ser definidos para as consultas executadas com frequência ou para as consultas que devem recuperar dados rapidamente. Um índice deve ser definido para cada conjunto de atributos usados como critérios de pesquisa.

Os índices devem ser definidos apenas em colunas usadas como identificadores e serem formados por números simples (inteiros ou tipos de dados numéricos), ocupando assim, menos espaço (em bytes) em disco. Muita atenção deve ser tomada ao se definir índices, pois quanto mais índices houver em uma tabela, mais demorado será o processamento de inserções e atualizações, considerando que deve haver atualização dos índices nestes procedimentos.

Em alguns sistemas, o desempenho de atualizações e inserções é mais importante do que o de consultas. Um exemplo comum são os aplicativos de aquisição de dados de manufatura, nos quais os dados referentes à qualidade são obtidos em tempo real. Nesses sistemas, as consultas *on-line* são ocasionais, e a maioria dos dados é analisada periodicamente por aplicativos de relatórios em lote que fazem a análise estatística dos dados. Para os sistemas de aquisição de dados, é recomendado remover todos os índices a fim de obter a taxa máxima de transferência de dados.

Alguns exemplos de implementação de índices são em árvores B e em Hashing. O primeiro baseia-se em estruturas de dados balanceadas de índices em árvore B. São indicados quando os valores-chave dos índices são distribuídos aleatoriamente e tendem a variar muito. Seu desempenho não é bom quando os dados que estão sendo indexados já estão em uma ordem seqüencial. O Hashing, por sua vez, tem os valores-chave dos índices calculados. Esse procedimento

baseia-se na utilização do valor-chave para calcular o endereço dos dados desejados. Devido à necessidade de previsibilidade, os índices de hashing são úteis apenas para as tabelas de pesquisa médias, com poucas mudanças.

A escolha da estratégia de indexação, assim como o tempo de criação de índices, pode causar um forte impacto sobre o desempenho. As cargas de dados volumosas devem ser realizadas sem índices. O motivo para isso é que quando cada linha é adicionada, a estrutura do índice é refeita. Como linhas subsequentes mudarão a estrutura ideal do índice, o trabalho com o rebalanceamento do índice, devido à inserção de novas linhas, será em grande parte desperdiçado.

Nesta seção também vale ressaltar o estudo da complexidade de algoritmos, dado que a indexação está diretamente ligada à ordenação de uma base de dados conforme um critério. Esta ordenação segue um algoritmo que, por sua vez, tem certa complexidade.

Assim, pode-se definir a complexidade temporal de um programa ou algoritmo como o tempo que ele demora em ser executado. Denomina-se T(n) o tempo de execução em função do tamanho (n) da entrada. Deste modo, n é o tamanho da entrada no caso de uma tabela de n tuplas ser percorrida em um suposto programa, por exemplo.

Em matemática, ciência da computação e áreas afins, a notação O, também conhecida como notação Grande O (*Big Oh*), notação Landau, notação Bachmann-Landau e notação assintótica, descreve o comportamento limite de uma função quando o argumento tende para um valor particular ou infinito, geralmente em termos de funções mais simples (AVIGAD, 04). “O” é o método formal de expressar o limite superior do tempo de execução de um algoritmo.

Para se avaliar um algoritmo, identificam-se as operações dominantes, mais frequentes ou mais demoradas e determina-se o número de vezes que são executadas, e não o tempo exato de cada execução. Este resultado de quantidade de execuções é expresso com a notação de O.

A notação O avalia a complexidade do algoritmo que, por sua vez, tem seu tempo de execução obtido a partir de constantes (número de vezes que os registros

da tabela são acessados) e parcelas menos significativas para valores grandes de n. Por definição T(n) = O (f(n)), onde T(n) é de ordem f(n), se e somente se existem constantes positivas c e n_0 tal que $T(n) \leq c * f(n)$ para todo $n > n_0$ (AVIGAD, 04). Por exemplo,

- sendo $T(n) = c_k\ n^k + c_{k-1}\ n^{k-1} + ... + c_0$; então a complexidade é $O(n^k)$, pois desprezam-se os termos $c_{k-1}\ n^{k-1} + ... + c_0$.
- sendo $T(n) = \log_2 n$ então a complexidade é O(log n), considerando que a mudança de base é apenas uma mudança de constante.
- sendo T(n) = 4 então a complexidade é O(1), considerando que o número 1 é usado para indicar uma ordem constante.

3.1.3. Características do Armazenamento Físico

Para uma implementação de um banco de dados, não se pode deixar de lado outras questões, além da própria modelagem do banco de dados, pois é natural que quanto maior o banco de dados, maiores serão os requisitos de *hardware*, ou muitos poderão ser os usuários e sessões concorrentes, transferência de transações e os tipos de operações no banco de dados. Além de requisitos de armazenamento em disco, por exemplo, um *data warehouse* pode também exigir mais memória e processadores mais rápidos, de modo que mais dados possam ser colocados em cache na memória e os dados possam ser processados rapidamente.

Como citado, bancos de dados realizam operações de entrada e saída em blocos de disco, e as operações de entrada e saída nos blocos costumam ser otimizadas no *software* e no *hardware* do sistema. Deste modo, a organização física das tabelas e dos índices no banco de dados pode causar um impacto significativo sobre o desempenho do sistema.

A **Densidade das Páginas** de disco depende da extensão das mudanças esperadas para os dados ao longo do tempo. Sem entrar em detalhes, uma página menos densa aceita melhor mudanças de valores ou inclusão de dados no decorrer do tempo. Ao passo que uma página de dados mais cheia melhora o desempenho de leitura, pois mais dados são recuperados por leitura de bloco. Uma recomendação seria agrupar as tabelas de acordo com sua probabilidade de sofrer

mudanças. Três grupos são um bom ponto de partida: muito dinâmicas, pouco dinâmicas e essencialmente estáticas. As tabelas muito dinâmicas (armazenamento de dados transacionais) devem ser mapeadas para as páginas de disco com bastante espaço vazio, as tabelas pouco dinâmicas para as páginas com menos espaço vazio, e as tabelas essencialmente estáticas (armazenamento de dados mestres) precisam de muito pouco espaço vazio. Os índices para as tabelas devem ser mapeados da mesma forma.

A questão da **Localização da Página no Disco** é outra dimensão de análise. O objetivo é tentar balancear a carga de trabalho entre várias unidades e cabeças de leitura/escrita do disco para reduzir ou eliminar gargalos. Algumas diretrizes seriam nunca colocar todos os dados no mesmo disco do sistema operacional, como seus arquivos temporários ou os dispositivos de troca; tentar colocar os dados acessados simultaneamente em unidades distintas para balancear a carga de trabalho; tentar colocar os índices em uma unidade diferente daquela que contém os dados que ela indexa, para também distribuir a carga de trabalho.

Na prática, raramente há unidades suficientes para a distribuição de entradas e saídas e o balanceamento das operações de entrada e saída continua sendo um processo iterativo e experimental. A criação do protótipo do desempenho do acesso a dados durante a fase de elaboração de um banco de dados, junto com a instrumentação apropriada para monitorar operações físicas e lógicas de entrada e saída, evidencia problemas de desempenho enquanto ainda há tempo de ajustar o projeto da modelagem do banco de dados.

Alocação de Espaço em Disco é a terceira dimensão relatada neste item. Para tal, é preciso estimar o número de objetos que precisam ser armazenados. O volume de espaço em disco necessário para armazenar os objetos varia de acordo com o banco de dados, mas geralmente a estimativa de tamanho é igual a **(carga fixa por tabela) + (número de linhas * (tamanho médio de linha/densidade média dos dados))** (RUP), onde a densidade média dos dados corresponde ao percentual de densidade por página. Além disso, o tamanho dos índices também deve ser considerado. Ao calcular o espaço em disco não se pode esquecer de considerar o crescimento da base de dados decorrente da inclusão de dados.

3.1.4. Disponibilidade (Replicação)

Uma opção para melhoria do desempenho de bancos de dados é a replicação dos dados para um ou mais servidores. Assim, pode-se utilizar a arquitetura de mestre/escravo para acesso aos dados. Mais comum para disponibilizar os dados enquanto se faz uma manutenção na base de dados, esta técnica é eficiente para acesso de vários usuários concorrentes a uma mesma base de dados. O SGBD faz uso do balanceamento de carga para distribuir o acesso às bases conforme sejam os usuários logados no momento.

Outra opção seria o uso de dispositivos RAID (do inglês, *Redundant Array of Independent Disks*), que nada mais são do que sistemas que incluem dois ou mais discos rígidos, ou seja, matrizes redundantes de discos independentes. Esta estrutura permite o armazenamento seguro e recuperação rápida da informação para obtenção de alta disponibilidade, resultando em menos falhas e em um tempo de parada menor. O uso destes dispositivos é totalmente não visível para usuário final, que não é capaz de distinguir se a aplicação que ele utiliza está sendo executada em um servidor único ou não. Ou seja, o fato de se acessar um ou outro servidor não acarreta mudanças ao comportamento do seu usuário.

3.1.5. Custo de Acesso a Dados

Definidas as principais consultas a serem realizadas no banco de dados, é possível alterar a modelagem do mesmo de forma que o custo de uma consulta seja menor, ou seja, o desempenho do banco de dados seja otimizado.

Para isso, faz-se uso de algoritmos para análise de custos (BATINI, 91). No geral, o desempenho de um banco de dados depende da quantidade de acessos a disco, dos tempos de CPU para a execução de consultas e do tempo para armazenar dados em tabelas resultantes e temporárias. Porém, para o caso do algoritmo e da estimativa de custo, basicamente é utilizado o número de acessos a disco.

Para a análise de custo de acesso a dados, utiliza-se a seguinte notação:

ntR (ou nt): número (médio) de tuplas na relação R;

ntbR (ou ntb): número de tuplas em um bloco de R (fator de bloco)

nbR (ou nb): número de blocos da relação R

$\lceil x \rceil$: menor inteiro maior ou igual a x

Deste modo, em uma relação R de 100 tuplas, em que cada tupla tem 512 bytes e o tamanho do bloco é de 2.048 bytes, tem-se:

nt: 100

ntb: 4 = 2.048/512 ⇔ 1 bloco contém 4 tuplas

nb: 25 = 100/4

Análise dos custos para operações de seleção (σ):

Seleção usando busca linear:

- Atributos não chave de R e não ordenados: número de acessos = nbR
- Em atributos chave de R: número médio de acesso = nbR/2

Seleção em busca binária:

Para arquivo ordenado no atributo A e a condição de seleção é uma comparação de igualdade no atributo A:

- A é atributo chave: número médio de acessos = $\lceil \log_2(nbR) \rceil$
- A não é chave: número médio de acessos = $\lceil \log_2(nbR) \rceil + \left\lceil \left(\frac{nti}{ntbR} \right) \right\rceil - 1$, onde:

nti: número de tuplas com valores iguais para A (dado em %), nti ≈ ntR/nVA

nVA: número de valores distintos de A

- A é chave e é usada em índice primário, e x é a altura do índice: número médio de acessos = x + 1. Um exemplo seria a de tabela Hash.
- Uso de índice ordenado para múltiplos registros: número médio de acesso = x + nbR/2. Um exemplo seria o ISAM (do inglês, *Indexed Sequential Access Method*)
- Uso de índice secundário com árvore B+: número médio de acessos = x + S, onde S é a cardinalidade da seleção

Considere-se uma relação Conta, com os atributos nomeAgencia, nroAgencia, nroConta e saldo. Uma consulta que deve ser analisada é a de selecionar as contas da agência cujo nome é “Centro”. Assim, temos:

Resultado = $\sigma_{\text{nomeAgencia=”Centro”}}$ (Conta)

Supondo:

ntb Conta=20 (número de tuplas por bloco)

nVA: nVnomeAgencia = 50 ⇔ ou seja, há 50 agências distintas

ntR: ntConta = 10.000

nbR: nbConta = 10.000/20 = 500 blocos

Logo:

Exemlo 1) Em uma busca linear:

número de acessos = **500**

Exemlo 2) Em uma busca onde Conta é ordenada por nomeAgencia:

nti = ntConta/nVA = 10.000/50 = 20

número de acessos = $\lceil \log_2(nbConta) \rceil + \left\lceil \left(\frac{nti}{ntbConta} \right) \right\rceil - 1 =$

$\lceil \log_2(500) \rceil + \left\lceil \left(\frac{200}{20} \right) \right\rceil - 1 =$ 9+10-1 = **18**

Exemlo 3) Em uma busca onde Conta é ordenada por nomeAgencia:

número de acessos = x + 1 = 1 + 1 = **2**

<u>Custos para operações de junção:</u>

Os algoritmos de junção envolvem sempre laços aninhados para estabelecer o produto cartesiano entre as tuplas das relações envolvidas.

Junção de laços aninhados de blocos para duas relações R e S:

número de acessos = nbR + (nbR * nbS),

para um algoritmo seguindo a lógica relatada a seguir:

```
para cada bloco de R
    para cada bloco de S
      para cada tupla tR de bR
            para cada tupla tS de bS
                  se o par (tR, tS) satisfaz a junção
                        adicionar tR*tS à resposta
                  fim-se
            fim-para
      fim-para
    fim-para
fim-para
```

Supondo as relações de Departamento e Empregado com

nbDepto = 10;

ntDepto = 50;

nbEmp = 2.000;

ntEmp = 6.000.

Tem-se:

i) Empregado $|\times|_{(Depto=sigla)}$Departamento

Não sendo o Departamento um atributo indexado:

Número de acessos: nbEmp + (nbEmp * nbDepto) = 2.000+(2.000*10) = 22.000

ii) Departamento $|\times|_{(sigla=Depto)}$ Empregado

Não sendo o Departamento um atributo indexado:

Número de acessos: nbDepto + (nbDepto * nbEmp) = 10+(10*2.000) = 20.010

Custos para operações de projeção (TT)

A operação de projeção lista todas as tuplas de uma determinada relação, retirando as duplicadas.

Custo médio = número de blocos de R = nbR

3.2. Espaço em Disco

Uma questão da modelagem que implica no uso de mais ou menos espaço em disco é o tipo de dado a ser usado, como, por exemplo, de tamanho fixo ou variável, atentando para o fato de poder receber valor nulo e requerer mais espaço em disco no caso de fixar o tamanho de um campo neste caso. É fato que em muitas situações não se dá a devida atenção aos tipos de dados de certos atributos e acaba-se usando tipos de dados super dimensionados, ocupando mais espaço em disco do que o necessário. Mesmo com a taxa de custo por megabyte diminuindo a cada dia, isso não significa que a modelagem deva usar tamanhos e tipos desnecessários, acima do requerido.

Um dos motivos do superdimensionamento está relacionado com a falta de previsibilidade do crescimento e volume de dados que serão armazenados. Nem sempre é fácil identificar qual é o melhor tipo de dados para determinada coluna e, por causa disso, o modelo acaba pecando por excesso.

Tomando como base o SGBD MySQL para análise dos tipos de dados, são mostrados no **Quadro 1** a seguir os tipos disponíveis e as quantidade de bytes que cada um ocupa em disco (MYSQL)

Tipo de Dado	Espaço em Disco
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT, INTEGER	4 bytes
BIGINT	8 bytes
FLOAT(p)	4 bytes se 0 <= p <= 24, 8 bytes se 25 <= p <= 53
FLOAT	4 bytes
DOUBLE [PRECISION], REAL	8 bytes
DECIMAL(M,D), NUMERIC(M,D)	Varia conforme versão. Para MySQL 5.5 o DECIMAL utiliza 4 bytes para cada 9 dígitos
BIT(M)	aproximadamente (M+7)/8 bytes
DATE	3 bytes
TIME	3 bytes
DATETIME	8 bytes
TIMESTAMP	4 bytes
YEAR	1 byte
CHAR(M) *	M × w bytes, 0 <= M <= 255, onde w é o número de bytes requeridos para o tamanho máximo de caracteres no conjunto de caracteres
BINARY(M) *	M bytes, 0 <= M <= 255
VARCHAR(M) *, VARBINARY(M) *	L + 1 bytes se o valor requerer 0 – 255 bytes, L + 2 bytes se os valores requererem mais de 255 bytes
TINYBLOB, TINYTEXT	L + 1 bytes, onde L < 28
BLOB, TEXT	L + 2 bytes, onde L < 216
MEDIUMBLOB, MEDIUMTEXT	L + 3 bytes, onde L < 224
LONGBLOB, LONGTEXT	L + 4 bytes, onde L < 232
ENUM('value1','value2',...)	1 ou 2 bytes, dependendo da enumeração de valores (máximo de 65.535 valores)
SET('value1','value2',...)	1, 2, 3, 4, ou 8 bytes, dependendo do número de membros no conjunto (máximo de 64 membros)

Quadro 1. Tipos de dados e espaços requeridos em disco pelo MySQL

* M representa o comprimento declarado da coluna em caracteres para tipos *string* não binários e bytes para tipos *string* binários. L representa o comprimento em bytes de um valor *string*.

É preciso avaliar os atributos numéricos conforme o intervalo de valores que eles podem assumir e com isso determinar seu tipo de dados, sem superestimar. O **Quadro 2** a seguir exibe os intervalos de valores que cada tipo de dado suporta. (MYSQL).

Tipo de Dados	Armazena-mento (Bytes)	Valor Mínimo (com sinal/sem sinal)	Valor Máximo (com sinal/sem sinal)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32.768	32.767
		0	65.535
MEDIUMINT	3	-8.388.608	8.388.607
		0	16.777.215
INT	4	-2.147.483.648	2.147.483.647
		0	4.294.967.295
BIGINT	8	-9.223.372.036.854.775.808	9.223.372.036.854.775.807
		0	18.446.744.073.709.551.615

Quadro 2. Tipos de dados numéricos e valores suportados pelo MySQL

Revendo e adequando o tipo de dado ao domínio do campo obtém-se uma redução do espaço ocupado pelas colunas numéricas, além de ganho de espaço caso estes atributos façam parte de índices. Além disso, o banco de dados ocuparia menos memória para armazenar as páginas de dados, as junções ficariam mais rápidas de serem executadas devido ao tamanho reduzido das chaves primárias, o *backup* e a replicação seriam executados mais rapidamente, o processador gastaria menos ciclos de CPU para executar cada instrução, dentre outros ganhos.

3.3. Flexibilidade e Custo de Manutenção

A flexibilidade e o custo de manutenção se traduzem em quão fácil é de se alterar o modelo de dados conforme uma necessidade específica. Por exemplo, alteração em uma hierarquia, inserção de níveis, mudanças de nós, inserção de novos atributos, alteração de cardinalidade de relacionamentos, dentre outros. Esta métrica é geralmente avaliada de forma empírica, como relatado no item 2.4 dessa dissertação.

3.4. Roteiro – Metodologia para Análise de Métricas

No Capítulo a seguir, cada padrão é avaliado unitariamente, isto é, cada um é avaliado de forma independente. Para cada padrão define-se um exemplo inicial, com um fragmento de um modelo de dados representando apenas a aplicação do

padrão, uma breve descrição do seu funcionamento e suas principais ações no banco, sendo elas consultas ou inserções/alterações. O fragmento de modelagem de banco de dados é transformado conforme o padrão propõe, e as métricas descritas neste Capítulo são avaliadas no modelo já transformado.

Os seguintes tópicos das métricas são analisados em cada aplicação de padrão:

a) Possibilidade de desnormalização para agilidade de consultas;

b) Método para melhor armazenamento lógico alinhado à estratégia de indexação;

c) Possibilidade ou necessidade de alta disponibilidade e uso de métodos de replicação;

d) Custo de Acesso a dados descrito(s) como sendo a(s) principal(is) consulta(s) realizada(s) na base de dados;

e) Possibilidade de redução do espaço em disco alterando-se (otimizando) apenas a modelagem e a utilização de tipos de dados distintos;

f) Flexibilidade e Custo de Manutenção.

4 ESTUDO DE CASO

A seguir cada padrão é avaliado conforme as métricas de desempenho detalhadas no Capítulo 3.

Para cada padrão, parte da modelagem de um banco de dados é usada como exemplo para avaliação do mesmo. Os exemplos referenciam-se a um hipotético sistema chamado ACERVUS, que tem como objetivo auxiliar nas tarefas e funções da biblioteca de uma instituição de ensino, realizando atividades como cadastro, consulta e remoção dos dados de pessoas vinculadas (alunos, professores e funcionários), dos dados de livros e seus exemplares e o gerenciamento e processamento de empréstimos de livros para os usuários da biblioteca.

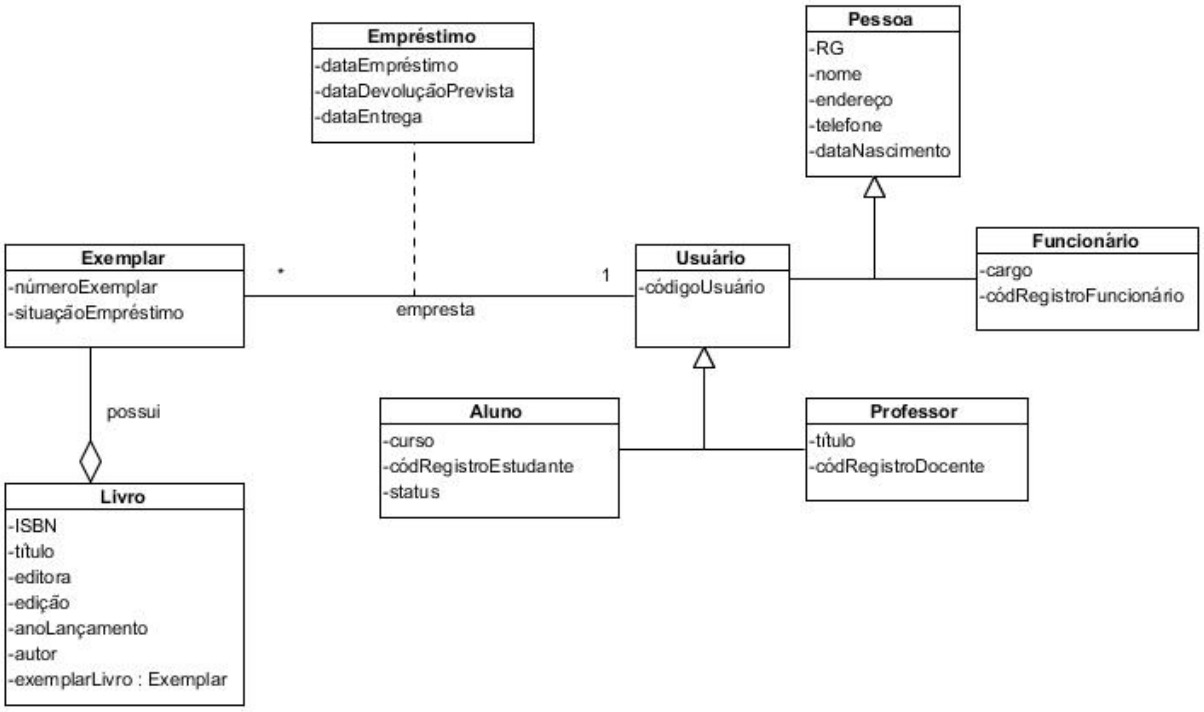

Figura 11. Diagrama de Classes do Sistema ACERVUS

Requisitos Funcionais:

RF1) O Sistema deve permitir a inclusão, alteração e remoção das pessoas vinculadas à biblioteca as quais devem ser caracterizadas por RG, nome, endereço (rua/av., número e CEP), telefone e data de nascimento (dia/mês/ano). As pessoas podem ser usuárias da biblioteca e funcionárias. Os únicos usuários armazenados

pelo sistema podem ser alunos ou professores. O Aluno possui ainda o código de registro de estudante e o Professor caracteriza-se pelo código de registro de docente e o título (nenhum, mestre, doutor, entre outros).

RF2) O sistema deve gerenciar empréstimos de livros feitos por usuários (alunos e professores), registrando o Código do Usuário, a data de empréstimo (Data e Hora), a data de devolução prevista (Data e Hora), a data de entrega (Data e Hora), o número do exemplar e o ISBN do livro referente.

RF3) Antes de autorizar um empréstimo, o sistema deve verificar se o cadastrado não excedeu o limite de livros retirados, e se sua data de liberação possui data inferior à data do sistema. Caso contrário o empréstimo não é autorizado.

RF4) Após a retirada de cada exemplar, o sistema deve calcular a data de devolução deste baseando-se na permissão do cadastrado.

RF5) Após a devolução, a situação do exemplar deve ser atualizada. O sistema deve verificar se a sua data de entrega é inferior ou igual à data de devolução registrada. No caso de ser posterior, o sistema deve atribuir ao cadastrado a data mínima (5 dias de multa para cada dia de atraso) para que este possa realizar um novo empréstimo.

RF6) Funcionários não estão autorizados a realizar empréstimo de livros. Logo, o sistema deve ser capaz de identificar se o código informado pertence a um aluno ou a um docente, antes que o empréstimo possa ser realizado.

RF7) O sistema deve permitir o cadastro de livros os quais devem possuir os seguintes atributos: código do livro (ISBN), título, editora, número da edição, ano de lançamento, e autor(es).

RF8) O sistema deve permitir a remoção de livros, através de seus códigos. Ao remover um livro do cadastro, o sistema deve se encarregar da exclusão de seus exemplares.

RF9) O sistema deve permitir o cadastro, consulta e remoção dos exemplares de um livro com os seguintes atributos: código do livro vinculado, número do exemplar e situação. É importante ressaltar que a situação denota o status em que o exemplar se encontra, no caso, disponível ou emprestado.

Consultas e Impressão de Relatórios:

C1) O sistema deve permitir a consulta de livros, através do seu Nome, Código ou Autor.

C2) O sistema deve permitir a impressão de uma listagem de livros pertencentes ao acervo da biblioteca, contendo o título, autor, código do livro e editora.

C3) O sistema deve permitir a consulta do número de retiradas de cada livro do acervo contendo o código do livro, o título e o número de exemplares retirados daquele livro.

C4) O sistema deve permitir a impressão de relatórios de empréstimos feitos, contendo o nome de quem emprestou, código do usuário, data de empréstimo, data de devolução, código do livro e título.

C5) O sistema deve permitir que sejam impressas relações de funcionários, contendo todos os dados registrados sobre eles.

C6) O sistema deve permitir aos usuários a consulta de seus dados e livros do acervo.

Requisitos Não-Funcionais:

RNF1) (Permissão) Cada aluno pode retirar 3 livros simultaneamente.

RNF2) (Permissão) Cada docente pode retirar 5 livros simultaneamente.

RNF3) (Permissão) Para um aluno, o período de empréstimo deve ser de 7 dias.

RNF4) (Permissão) Para um docente, o período de empréstimo deve ser de 10 dias.

RNF5) (Segurança) O sistema deve possuir senhas de acesso e identificação para diferentes tipos de usuário: administrador do sistema, outros funcionários, alunos e professores, garantindo o controle do acesso aos dados e funcionalidades para cada tipo de usuário.

RNF6) (Confiabilidade) O sistema deve ter a capacidade de recuperar dados perdidos, em caso de alguma falha, da última operação não salva.

RNF7) (Segurança) O sistema deve permitir a realização de backup dos arquivos de sistema.

RNF8) (Eficiência) As remoções dos dados devem ser concluídas em, no máximo, 5 segundos.

RNF9) (Eficiência) A impressão de relatórios deve ser iniciada em, no máximo, 15 segundos após a solicitação.

RNF10) (Eficiência) A resposta às consultas devem ser realizadas dentro de, no máximo, 10 segundos.

Considerações:

Tamanho do bloco: 4.096 bytes

Volumetria

- Pessoa: 75.000 registros
- Aluno: 67.500 registros (90% da base de dados)
- 80% dos Alunos têm status Ativo, ou seja, 54.000 registros
- Professor: 4.000 registros (5,3 % da base de dados)
- Usuário: 71.500 registros (soma de Alunos e Professores)
- Funcionário: 3.500 registros (4,7% da base de dados)
- Livro: 75.000 registros
- Exemplar: 150.000 registros (1 Livro tem em média 2 Exemplares)
- Empréstimo: 30.000 registros
- Usuários com empréstimo atualmente: 3.000

Tipo de dado e Quantidade de bytes dos atributos mapeados para MySQL:

Classe	Atributo	Tipo de Dado	Quantidade de bytes
Pessoa	RG	VARCHAR(14)	15
Pessoa	nome	VARCHAR(29)	30
Pessoa	rua	VARCHAR(29)	30
Pessoa	número	INT	4
Pessoa	CEP	VARCHAR(14)	15
Pessoa	telefone	VARCHAR(15)	16
Pessoa	dataNascimento	DATE	3
Usuário	códigoUsuário	INT	4
Aluno	curso	VARCHAR(39)	40
Aluno	códRegistroEstudante	INT	4
Aluno	status	VARCHAR(1)	1
Professor	título	VARCHAR(39)	40
Professor	códRegistroDocente	INT	4
Funcionário	cargo	VARCHAR(39)	40
Funcionário	códRegistroFuncionário	INT	4
Livro	ISBN	VARCHAR(24)	25
Livro	título	VARCHAR(29)	30
Livro	editora	VARCHAR(29)	30
Livro	edição	INT	4
Livro	anoLançamento	INT	4
Livro	autor	VARCHAR(29)	30
Exemplar	númeroExemplar	INT	4
Exemplar	situaçãoEmpréstimo	VARCHAR(9)	10
Empréstimo	dataEmpréstimo	DATE	3
Empréstimo	dataDevoluçãoPrevista	DATE	3
Empréstimo	dataEntrega	DATE	3

Quadro 3. Tipos de dados e quantidade de bytes dos atributos de ACERVUS

Com estes valores é possível calcular os parâmetros para cálculo de custo das consultas que será demonstrado a seguir para cada avaliação de padrão.

4.1. Padrões de mapeamento de Herança

4.1.1. Herança de Classes em Tabela Única

Para análise do padrão de herança em classe em tabela única é utilizada a parte da modelagem do sistema ACERVUS que permite o cadastro, consulta e

remoção dos dados de pessoas vinculadas (alunos, professores e funcionários). A **Figura 12** ilustra essa parte do modelo de dados do sistema em questão.

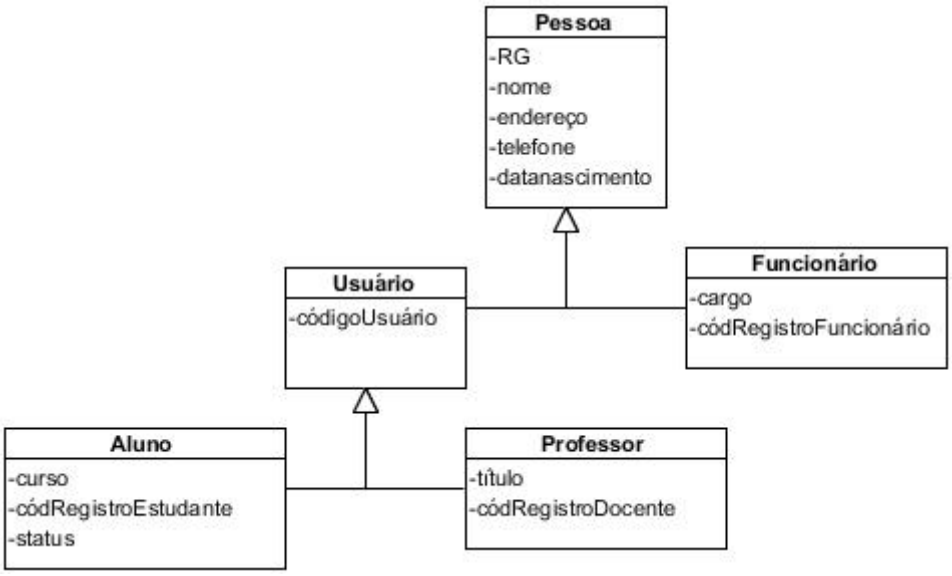

Figura 12. Diagrama de Classes que exemplifica hierarquia

Supondo as seguintes consultas mais frequentes:

1. Imprimir a relação de funcionários, contendo todos os dados registrados sobre eles.
2. Imprimir a relação de todos os alunos cadastrados na biblioteca e com status ativo. A impressão deve conter nome, código do registro do estudante e curso.

A partir do modelo ilustrado na **Figura 12**, é realizada a aplicação do padrão “Herança de Classes em Tabela Única” e é gerada a tabela ilustrada na **Figura 13**, com a união de todos os atributos de todas as classes da hierarquia da herança como colunas em uma tabela única.

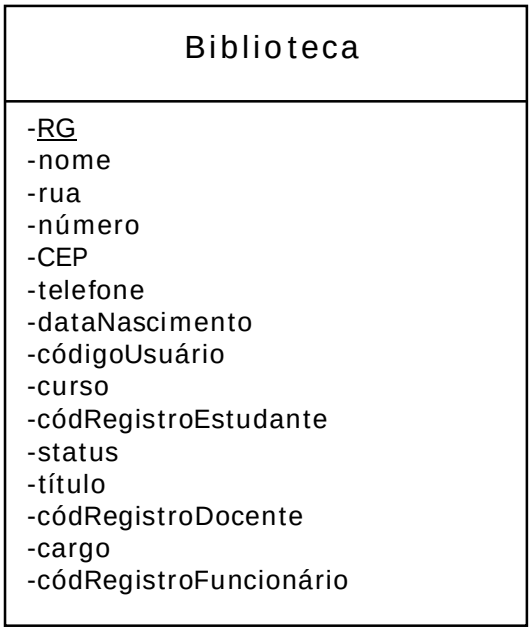

Figura 13. Tabela para o Caso Herança de Classes em Tabela Única

De acordo com a tabela apresentada e conforme as métricas definidas no Capítulo 3 realiza-se, a seguir, a análise dessa modelagem.

a) Possibilidade de desnormalização para agilidade de consultas

Não se aplica a este padrão, dado que a desnormalização combina colunas de duas ou mais tabelas distintas na mesma tabela, ligando previamente as informações. O propósito do padrão é unir todos os atributos em uma tabela e no caso da herança, não há ligações necessárias.

Pelo fato de classes em tabela única serem armazenas em apenas uma tabela, é possível obter qualquer descendente da classe base com uma operação de leitura no banco de dados, bem como a operação de gravação, que também é única. No entanto, deve-se levar em consideração que esta afirmação só é válida para tabelas que encaixem unicamente em um bloco, ou seja, que o acesso à tabela em disco seja feito em uma única vez e que este retorne a tabela como um todo.

b) Armazenamento lógico / Estratégia de indexação

Supondo o armazenamento em uma tabela sem ordenação, tem-se que a inserção de valores na tabela é de complexidade O(1), bem como a exclusão e a modificação, pois não é preciso buscar uma posição na tabela. A pesquisa é dada por O(n), sendo que n é no máximo a quantidade de registros que a tabela contém e que a pesquisa deverá ler os registros um a um até que encontre o registro pesquisado.

No caso do armazenamento ser em uma tabela ordenada pelo RG, por exemplo, tem-se que a inserção de um registro seria O(n), sendo n o deslocamento de n registros para inserção do valor atual. A exclusão e a modificação também seriam O(n) pelo mesmo motivo. A complexidade da criação de um registro, já de maneira ordenada é dada por O (n^2), conforme algoritmo a seguir.

```
Função Insere-ordenando (valor)
    para j:=2 até n faça                                          } O(n)
        chave := valor[j]                                         }
        i := j - 1                                                }
        enquanto i > 0 e valor[i] > chave faça                    } O(n)
            valor[i+1] := valor[i]                                }
            i := i – 1                                            }
        Fim-enquanto                                              }
        valor[i+1] := chave                                       }
    Fim-para                                                      } O (n²)
Fim-função
```

Por outro lado, uma criação sequencial com posterior ordenação se dá por O(n log n), ou seja, de melhor desempenho, também conforme algoritmo a seguir que descreve a ordenação de um vetor da esquerda para a direita fazendo uso de um pivô, que sempre divide a base de dados em duas partes (complexidade $log_2 n$), fazendo com que o desempenho melhore. A criação realizada antes da ordenação é O(1), feita no final de toda a base de dados, por exemplo, e não é ilustrada por meio de algoritmos devido a sua simplicidade e por não ser levada em consideração quando temos um algoritmo de ordenação que é de ordem n, e que faz com que a complexidade do ato de inserir e depois ordenar seja dada apenas pela complexidade do algoritmo que é de ordem n, e não constante.

```
Função Ordenação(valor[], esq, dir)                                          O(n log n)
i ← esq
n ← dir
pivo ← valor[(i+n)/2]
faça                                                                  O(n)
    enquanto (a[i] < pivo)                  c * log2n
        i←i+1
    enquanto (pivo < a[n])                  O(log n)
        j←j-1
    se (i <= n)
        Troca(valor[i++], valor[n--]);
enquanto (i <= n)
se (esq < n)
    Quicksort (valor, esq, n);
se (i < dir)
    Quicksort(valor, i, dir);
Fim-função
```

Em caso de armazenamento ordenado por meio de índice, seria recomendado colocar o RG como índice caso houvesse muita pesquisa direcionada ao usuário. Uma indexação por códigoUsuário neste caso também seria altamente recomendável, de modo que o ao se pesquisar os livros emprestados a um certo usuário, a pesquisa à base seria muito agilizada. No caso das consultas frequentes citadas anteriormente, não há necessidade de indexar a base de dados pois os dados são analisados periodicamente por geração de relatórios em lote.

c) Possibilidade ou necessidade de alta disponibilidade e uso de métodos de replicação

A partir das consultas principais relatadas, não é necessária alta disponibilidade dos dados, pois o uso desta parte do sistema não implica em consultas que necessitam retorno imediato. As consultas são apenas para extração de relatórios.

d) Custo e Otimização da(s) consulta(s) descrita(s) como sendo a(s) principal(is) consulta(s) realizada(s) na base de dados

Os números apresentados para os cálculos a seguir estão descritos no Quadro 3, página 50.

Tabela Pessoa	Tamanho do bloco: **tb** = 4.096 bytes Tamanho da tupla: **tamt** = 15 + 30 + 30 + 4 + 15 + 16 + 3 + 4 + 40 + 4 + 1 + 40 + 4 + 40 + 4 = 250 bytes Número médio de tuplas: **nt** = 75.000 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/250⌉ = 17 Número de blocos (nt / ftb): **nb** = ⌈75.000/17⌉ = 4.412

Consultas mais frequentes:

1. Imprimir a relação de funcionários, contendo todos os dados registrados sobre eles.

ESQUEMA DE NAVEGAÇÃO:

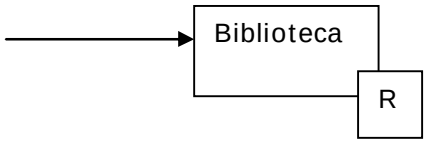

TABELA DE VOLUME DE ACESSO DA OPERAÇÃO:

Operação	Frequência	Conceito	R (Retrieval), U (Update), I (Insert) ou D (Delete)	Média de ocorrências acessadas
01	1 vez ao dia	Biblioteca	R	1

$R1 = \sigma_{codRegistroFuncionarios}$ (Biblioteca)

$R2 = \pi_{RG, nome, endereço, telefone, dataNascimento, cargo, codRegistroFuncionarios}$ (R1)

CUSTOS:

O custo médio para uma operação de projeção é o custo é o acesso aos blocos de toda a tabela. Como demonstrado anteriormente, esse custo é de **4.412** acessos.

2. Imprimir a relação de todos os alunos cadastrados na biblioteca e com status ativo. A impressão deve conter nome, código do registro do estudante e curso.

ESQUEMA DE NAVEGAÇÃO:

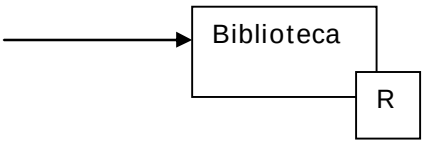

TABELA DE VOLUME DE ACESSO DA OPERAÇÃO:

Operação	Frequência	Conceito	R (Retrieval), U (Update), I (Insert) ou D (Delete)	Média de ocorrências acessadas
02	1 vez ao dia	Biblioteca	R	1

$R1 = \sigma_{codRegistroEstudante}$ (Biblioteca)

$R2 = \sigma_{status= \text{“A”}}$ (R1)

$R3 = \pi_{nome, curso, codRegistroEstudante}$ (R2)

CUSTOS:

R1: O custo médio para uma operação de seleção é o custo é o acesso aos blocos de toda a tabela R1, que é de **4.412** acessos.

R2: Supondo que 90% da base de dados sejam de estudantes, tem-se 75.000 * 0,9 = 67.500 tuplas de alunos.

R2	Número médio de tuplas: **nt** = 75.000 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/250⌉ = 17 Número de blocos (nt / ftb): **nb** = ⌈67.500/17⌉ = 3.971

Com isso, o custo da consulta R2 é de **3.971** acessos.

R3:

Custo médio = nbR, que é a quantidade de blocos para o total de registros que a busca retornou.

67.500 tuplas são referentes a alunos. Se 80% destes alunos estão ativos, significa que R2 retornou um total de 54.000 tuplas.

R3	Número médio de tuplas: **nt** = 54.000 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/250⌉ = 17 Número de blocos (nt / ftb): **nb** = ⌈54.000 /17⌉ =3.177

Com isso, o custo da consulta é de **3.177** acessos.

Portanto, tem-se que o custo total da consulta é Custo de R1 + Custo de R2 + Custo de R3 = **4.412** + **3.971** + **3.177** = **11.560** acessos

e) Possibilidade de redução do espaço em disco alterando-se (otimizando) apenas a modelagem e a utilização de tipos de dados distintos

Quanto mais níveis hierárquicos, maior é o consumo em disco, pois os atributos de todos os objetos serão mapeados, e atributos que não fazem parte de um objeto terão espaço em disco reservado e será preenchido com nulo quando não houver relação com o valor a que se refere à tupla.

Uma opção para redução de espaço em disco é a alteração do tipo de dado. No entanto, deve-se avaliar como o SGBD armazena cada tipo para poder fazer a melhor escolha. No caso deste exemplo de Biblioteca em que se tem alguns valores do tipo inteiro (INT), pode-se avaliar se o seu uso se adapta aos formatos TINYINT, SMALLINT ou MEDIUMINT, que ocupam, no MySQL, um espaço de armazenamento de 1, 2 e 3 bytes respectivamente – menor que os 4 bytes de ocupação do tipo INT. Uma melhoria que poderia ser implementada seria a troca do tipo de dado do atributo “número” para SMALLINT, que suporta valores positivos até 65.535. Outra melhoria seria a troca dos tipos dos atributos “códigoUsuário”,

“códRegistroEstudante”, “códRegistroDocente” e “códRegistroFuncionário” para MEDIUMINT, que suportaria até 16.777.215 valores.

f) Flexibilidade e Custo de Manutenção

Alterações na estrutura da hierarquia ou movimento de atributos entre classes da hierarquia não necessitam alterações no banco de dados. Uma inserção de níveis hierárquicos acarreta apenas a inserção, na tabela, dos atributos diferentes dos presentes em toda estrutura hierárquica.

4.1.2. Herança de Classes em Tabelas para cada Classe

Utilizando o mesmo Sistema ACERVUS descrito no item 4.1.1, a partir do modelo ilustrado na **Figura 12**, a aplicação do padrão “Herança de Classes em Tabelas para cada Classe” são montadas as tabelas ilustradas na **Figura 14**, com a criação de tabelas separadas para cada classe da hierarquia da herança. Os atributos são as colunas das tabelas.

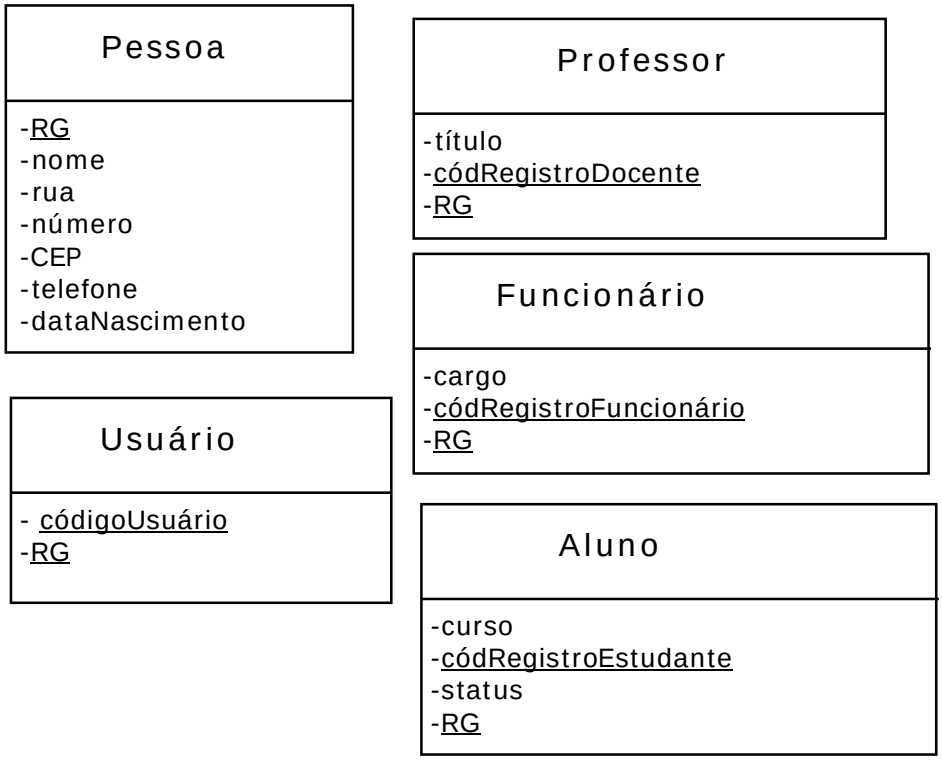

Figura 14. Tabelas para o Caso Herança de Classes em Tabela para cada Classe

No que se refere ao tamanho dos atributos, valem as mesmas considerações do item 4.1.1.

De acordo com as tabelas apresentadas e conforme as métricas definidas no Capítulo 3 realiza-se, a seguir, a análise dessa forma de modelagem.

a) Possibilidade de desnormalização para agilidade de consultas

A abordagem deste padrão não é determinante direto do desempenho. A leitura de uma classe derivada tem tantas chamadas ao banco quanto for o nível dela na hierarquia. Caso haja a classe base e em segundo momento a classe derivada, haverá 2 chamadas ao banco, uma à classe base e outra fazendo a associação entre a classe base e a derivada. Assim, a escrita e leitura dos dados de um objeto têm um custo definido pela profundidade da árvore hierárquica.

A desnormalização é uma opção válida para aplicação deste padrão, dado que com ela é possível combinar colunas de duas ou mais tabelas distintas na mesma tabela, ligando previamente as informações. Sendo assim, poder-se-ia unir todos os atributos em uma tabela e no caso da herança, não haveria ligações necessárias. Com isso, tem-se como resultado uma modelagem tal como a da **Figura 13**, da aplicação do padrão Herança de Classe em Tabela Única.

b) Armazenamento lógico / Estratégia de indexação

A análise da forma de armazenamento e indexação é a mesma contida no item 4.1.1.

c) Possibilidade ou necessidade de alta disponibilidade e uso de métodos de replicação

A partir das consultas principais relatadas, não é necessário alta disponibilidade pois o uso desta parte do sistema não implica em consultas que necessitam retorno imediato. As consultas são apenas para extração de relatórios.

d) Custo e Otimização da(s) consulta(s) descrita(s) como sendo a(s) principal(is) consulta(s) realizada(s) na base de dados

Os números apresentados para os cálculos a seguir estão descritos no Quadro 3, página 50.

Tabela Pessoa	Tamanho da tupla: **tamt** = 15 + 30 + 30 + 4 + 15 + 16 + 3 = 113 bytes Número médio de tuplas: **nt** = 75.000 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/113⌉ = 37 Número de blocos (nt / ftb): **nb** = ⌈75.000/37⌉ = 2.028
Tabela Funcionário	Tamanho da tupla: **tamt** = 40 + 4 + 15 = 59 bytes Número médio de tuplas: **nt** = 3.500 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/59⌉ = 70 Número de blocos (nt / ftb): **nb** = ⌈3.500/70⌉ = 50
Tabela Aluno	Tamanho da tupla: **tamt** = 40 + 4 + 1 + 15 = 60 bytes Número médio de tuplas: **nt** = 67.500 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/60⌉ = 69 Número de blocos (nt / ftb): **nb** = ⌈67.500/69⌉ = 979

Consultas mais frequentes:

1. Imprimir a relação de funcionários, contendo todos os dados registrados sobre eles.

ESQUEMA DE NAVEGAÇÃO:

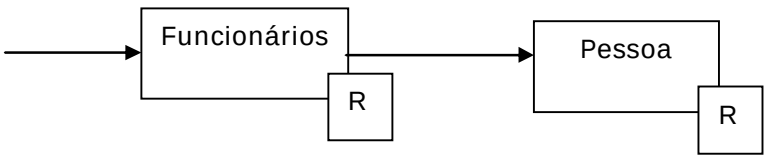

TABELA DE VOLUME DE ACESSO DA OPERAÇÃO:

Operação	Frequência	Conceito	R (Retrieval), U (Update), I (Insert) ou D (Delete)	Média de ocorrências acessadas
01	1 vez ao dia	Funcionário	R	1

R1 = Funcionário $|\times|$ Pessoa

R2 = $\Pi_{\text{RG, nome, endereço, telefone, dataNascimento, cargo, codRegistroFuncionarios}}$ (R1)

CUSTOS:

R1: nbFuncionario + nbFuncionario * nbPessoa

50 + 50 * 2.028 = **101.450** acessos

R2: Supondo que o retorno seja de 3.500 funcionários, isto é, a quantidade de funcionários cadastrados, tem-se para R2:

R2	Tamanho da tupla: **tamt** = todos os atributos de Funcionário somados aos de Pessoa = 59 + 113 = 172 bytes Número médio de tuplas: **nt** = 3.500 Fator tamanho de bloco (tb / tamt): **ftb** = $\lceil 4.096/172 \rceil = 24$ Número de blocos (nt / ftb): **nb** = $\lceil 3.500/24 \rceil = 146$

Assim, o custo é o acesso a todos os blocos, que é de **146** acessos.

Custo total da consulta: Custo de R1 + Custo de R2 = **101.450 + 146 = 101.596** acessos

2. Imprimir a relação de todos os alunos cadastrados na biblioteca e com status ativo. A impressão deve conter nome, código do registro do estudante e curso.

ESQUEMA DE NAVEGAÇÃO:

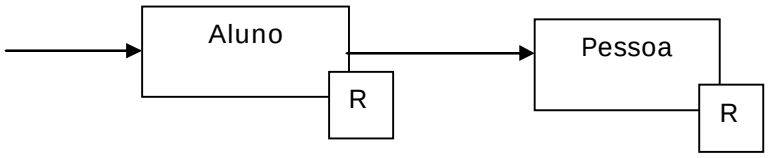

TABELA DE VOLUME DE ACESSO DA OPERAÇÃO:

Operação	Frequência	Conceito	R (Retrieval), U (Update), I (Insert) ou D (Delete)	Média de ocorrências acessadas
01	1 vez ao dia	Aluno	R	1

R1 = Aluno $|\times|$ Pessoa

R2 = $\sigma_{\text{status= "A"}}$ (R1)

R3 = $\pi_{\text{nome, curso, codRegistroEstudante}}$ (R2)

CUSTOS:

R1: nbAluno + (nbAluno * nbPessoa) = 979 + (979 * 2.028) = **1.986.391** acessos

R2: Supondo que o retorno seja de 67.500 alunos, isto é, a quantidade total de alunos cadastrados, tem-se para R2:

R2	Tamanho da tupla: **tamt** = todos os atributos de Aluno somados aos de Pessoa = 60 + 113 = 173 bytes Número médio de tuplas: **nt** = 67.500 Fator tamanho de bloco (tb / tamt): **ftb** = $\lceil 4.096/173 \rceil = 24$ Número de blocos (nt / ftb): **nb** = $\lceil 67.500/24 \rceil = 2.813$

Com isso, o custo da consulta R2 é de **2.813** acessos.

R3:

Custo médio = nbR, que é a quantidade de blocos para o total de registros que a busca retornou.

67.500 tuplas são referentes a alunos. Se 80% destes alunos estão ativos, significa que R2 retornou um total de 54.000 tuplas.

R3	Tamanho da tupla: **tamt** = todos os atributos de Aluno somados aos de Pessoa = 60 + 113 = 173 bytes Número médio de tuplas: **nt** = 54.000 Fator tamanho de bloco (tb / tamt): **ftb** = $\lceil 4.096/173 \rceil = 24$ Número de blocos (nt / ftb): **nb** = $\lceil 54.000/24 \rceil = 2.250$

Com isso, o custo da consulta é de **2.250** acessos.

Portanto, tem-se que o custo total da consulta é Custo de R1 + Custo de R2 + Custo de R3 = **1.986.391** + **2.813 + 2.250 = 1.991.454** acessos.

e) Possibilidade de redução do espaço em disco alterando-se (otimizando) apenas a modelagem e a utilização de tipos de dados distintos

O consumo em disco decorrente da aplicação deste padrão não é alto, considerando que atributos apenas são preenchidos quando realmente pertencem a um objeto. Não há atributos que possam ficar ociosos por não terem relação direta com aquele ramo da herança ao qual um determinado objeto pertença.

Uma possibilidade é a avaliação da quantidade de caracteres que estão sendo alocados em cada atributo, como Titulo, Cargo, Curso. Cada caractere a mais alocado e não usado é desperdiçado, logo pode-se considerar a redução dos mesmos. Por exemplo, sendo os títulos “Doutor” ou “Mestre”, é possível reduzir o atributo Titulo para tamanho 6 ao invés de 39. O mesmo pode ocorrer para Cargo. Uma análise de possíveis valores para este atributo, isto é, o seu domínio, pode-se mostrar que as opções válidas são “Professor Associado”, “Professor Adjunto”, “Secretária”, “Auxiliar Administrativo”, “Bibliotecário”. Assim, o maior valor possível tem 22 caracteres e, portanto, pode-se alterar o atributo Cargo para VARCHAR(22).

f) Flexibilidade e Custo de Manutenção

Alterações na estrutura da hierarquia ou movimento de atributos entre classes da hierarquia não provocam alterações no banco de dados. A inserção de níveis hierárquicos acarretará apenas a criação de uma nova tabela com seus atributos.

4.1.3. Herança de Classes em Tabelas para cada Classe Concreta

Utilizando o mesmo Sistema ACERVUS descrito no item 4.1.1, a partir do modelo ilustrado na Figura 12, a aplicação do padrão “Herança de Classes em Tabelas para cada Classe Concreta” são montadas as tabelas ilustradas na **Figura 15**, com a criação de tabelas separadas para cada classe da hierarquia da herança. Os atributos são as colunas das tabelas.

Figura 15. Tabelas para o Caso Herança de Classes em Tabelas para cada Classe Concreta

No que se refere ao tamanho dos atributos, valem as mesmas considerações do item 4.1.1.

De acordo com a tabela apresentada e conforme as métricas definidas no Capítulo 3, realiza-se, a seguir, a análise dessa forma de modelagem.

a) Possibilidade de desnormalização para agilidade de consultas

Apenas uma operação para escrita ou leitura de um objeto no banco de dados é requerida nesta abordagem. Já para a execução de leituras polimórficas, como o caso de uma listagem de todos os RGs e nomes de pessoas cadastradas no sistema ACERVUS, todas as tabelas da hierarquia devem ser lidas. Ou seja, é necessário acessar todas as tabelas quando a consulta exige retornar um objeto da original

Classe Base do modelo OO. Isto é, uma consulta deste tipo deve recuperar todos os dados relacionados que estão presentes em todas as tabelas, tornando-se este tipo de consulta das mais onerosas.

A desnormalização é uma opção válida para aplicar-se a este padrão, dado que com ela é possível combinar colunas de duas ou mais tabelas distintas na mesma tabela, ligando previamente as informações. No entanto, esta desnormalização deve ser avaliada com base na frequência da execução de uma consulta que teria melhor resultado caso o modelo fosse desnormalizado. Se determinada consulta é rara ou não tem impacto a sua demora, o modelo pode ser mais bem aproveitado da maneira como originalmente fora desenhado.

Sendo a consulta algo frequente e crítico, poder-se-ia unir todos os atributos em uma tabela e no caso da herança, não haveria ligações necessárias. Com isso, temos como resultado uma modelagem tal como a da **Figura 13**, da aplicação do padrão “Herança de Classes em Tabela Única”.

b) Armazenamento lógico / Estratégia de indexação

A análise da forma de armazenamento e indexação é a mesma contida no item 4.1.1.

c) Possibilidade ou necessidade de alta disponibilidade e uso de métodos de replicação

A partir das consultas principais relatadas, não é necessária alta disponibilidade pois o uso desta parte do sistema não implica em consultas que necessitam retorno imediato. As consultas são apenas para extração de relatórios.

d) Custo e Otimização da(s) consulta(s) descrita(s) como sendo a(s) principal(is) consulta(s) realizada(s) na base de dados

Os números apresentados para os cálculos a seguir estão descritos no Quadro 3, página 50.

Tabela Funcionário	Tamanho da tupla: **tamt** = 15 + 30 + 30 + 4 + 15 + 16 + 3 + 40 + 4 = 157 Número médio de tuplas: **nt** = 3.500 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/157⌉ = 27 Número de blocos (nt / ftb): **nb** = ⌈3.500/27⌉ = 130
Tabela Aluno	Tamanho da tupla: **tamt** = 15 + 30 + 30 + 4 + 15 + 16 + 3 + 4 + 40 + 4 + 1 = 162 Número médio de tuplas: **nt** = 67.500 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/162⌉ = 26 Número de blocos (nt / ftb): **nb** = ⌈67.500/26⌉ = 2.597

1. Imprimir a relação de funcionários, contendo todos os dados registrados sobre eles.

ESQUEMA DE NAVEGAÇÃO:

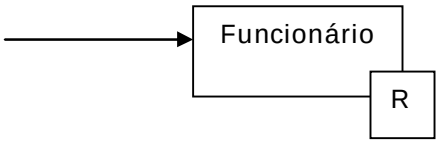

TABELA DE VOLUME DE ACESSO DA OPERAÇÃO:

Operação	Frequência	Conceito	R (Retrieval), U (Update), I (Insert) ou D (Delete)	Média de ocorrências acessadas
01	1 vez ao dia	Funcionário	R	1

R1 = $\Pi_{\text{RG, nome, endereço, telefone, dataNascimento, cargo, codRegistroFuncionarios}}$ (Funcionário)

CUSTOS:

O custo é o acesso a todos os blocos da tabela Funcionário, calculado no início desta sessão e que é de **135** acessos.

2. Imprimir a relação de todos os alunos cadastrados na biblioteca e com status ativo. A impressão deve conter nome, código do registro do estudante e curso.

ESQUEMA DE NAVEGAÇÃO:

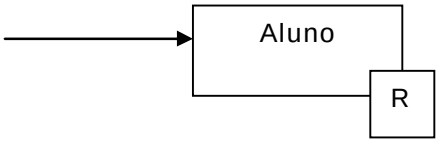

TABELA DE VOLUME DE ACESSO DA OPERAÇÃO:

Operação	Frequência	Conceito	R (Retrieval), U (Update), I (Insert) ou D (Delete)	Média de ocorrências acessadas
01	1 vez ao dia	Alulno	R	1

R1 = $\mathbf{TT}_{\text{nome, curso, codRegistroEstudante}}$ (Aluno)

CUSTOS:

O custo é o acesso a todos os blocos, que é de **2.597** acessos.

e) Possibilidade de redução do espaço em disco alterando-se (otimizando) apenas a modelagem e a utilização de tipos de dados distintos

O consumo em disco decorrente da aplicação deste padrão não é alto, considerando que atributos apenas são preenchidos quando realmente pertencem a um objeto. Não há atributos que possam ficar ociosos por não terem relação direta com aquele ramo da herança ao qual um determinado objeto pertença.

Uma possibilidade é a avaliação da quantidade de caracteres que estão sendo alocados em cada atributo, como Titulo, Cargo, Curso. Cada caractere a mais alocado e não usado é desperdiçado, logo pode-se avaliar a redução dos mesmo. Por exemplo, sendo os títulos “Doutor” ou “Mestre”, é possível reduzir o atributo Titulo para tamanho 6 ao invés de 39. O mesmo pode ocorrer para Cargo. Uma análise de possíveis valores para este atributo, isto é, o seu domínio, pode mostrar

que as opções válidas são “Professor Associado”, “Professor Adjunto”, “Secretária”, “Auxiliar Administrativo”, “Bibliotecário”. Assim, o maior valor possível tem 22 caracteres e, portanto, podemos manter o atributo Cargo como sendo VARCHAR(22).

f) Flexibilidade e Custo de Manutenção

Alterações na estrutura da hierarquia ou movimento de atributos entre classes da hierarquia não necessitam alterações no banco de dados. A inserção de níveis hierárquicos acarretará apenas a criação de uma nova tabela com seus atributos.

4.2. Padrões de mapeamento de Agregação

4.2.1. Agregação em Tabela Única

Considerando outra parte da modelagem do Sistema ACERVUS, descrito no item 4.1.1, seja a **Figura 16**, que contém mais especificamente a parte referente aos dados de livros e seus exemplares.

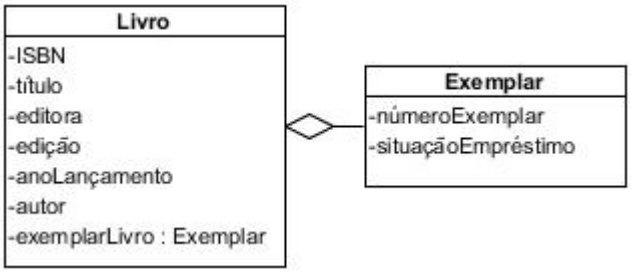

Figura 16. Diagrama de Classes para cadastro de Livros e Exemplares

E supondo as seguintes consultas mais frequentes:

1. Retirar um exemplar. Isto ocorre 500 vezes ao dia.
2. Consultar livros, através do seu Nome, Código ISBN ou Autor. Isto ocorre 1000 vezes ao dia.

A partir do modelo ilustrado na **Figura 16**, a aplicação do padrão “Agregação em Tabela Única” é realizada e ilustrada na **Figura 17**, com a união de todos os atributos de todas as classes envolvidas na agregação como colunas em uma tabela única.

Figura 17. Tabela para o Caso Agregação em Tabela Única

No que se refere ao tamanho dos atributos, valem as mesmas considerações do item 4.1.1.

De acordo com a tabela apresentada e conforme as métricas definidas no Capítulo 3, realiza-se, a seguir, a análise dessa forma de modelagem.

a) Possibilidade de desnormalização para agilidade de consultas

A análise da desnormalização é a mesma contida no item 4.1.1.

É importante ressaltar que esta solução é ideal em termos de desempenho, pois apenas uma tabela precisa ser acessada para recuperar um objeto de agregação com todos os seus objetos agregados. Por outro lado, os campos de atributos de objetos agregados tendem a aumentar o número de páginas recuperadas com cada acesso ao banco de dados, resultando em uma possível perda de largura de banda de entrada e saída.

b) Armazenamento lógico / Estratégia de indexação

A análise da forma de armazenamento e indexação é a mesma contida no item 4.1.1.

c) Possibilidade ou necessidade de alta disponibilidade e uso de métodos de replicação

A partir das consultas principais relatadas, pode haver necessidade de alta disponibilidade, principalmente imaginando um crescimento no número de usuários ou livros adquiridos pela biblioteca. O uso do sistema para consultas de extração de relatórios não necessita uma alta disponibilidade, mas as operações de retirada e devolução de livros que requerem acesso rápido ao sistema podem ficar lentas conforme a quantidade de dados inseridos no sistema, sem falar em questões de usuários concorrentes ou limitações de rede.

d) Custo e Otimização da(s) consulta(s) descrita(s) como sendo a(s) principal(is) consulta(s) realizada(s) na base de dados

Os números apresentados para os cálculos a seguir estão descritos no Quadro 3, página 50.

Tabela Livro	Tamanho da tupla: **tamt** = 25 + 30 + 30 + 4 + 4 + 30 + 4 + 10 + 4 + 10 + 4 + 10 = 165 bytes Número médio de tuplas: **nt** = 75.000 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/165⌉ = 25 Número de blocos (nt / ftb): **nb** = ⌈75.000/25⌉ = 3.000

1. Retirar um exemplar. Isto ocorre 500 vezes ao dia.

ESQUEMA DE NAVEGAÇÃO:

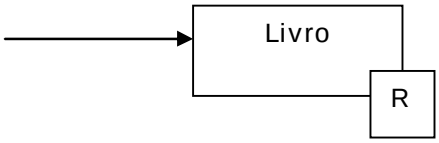

TABELA DE VOLUME DE ACESSO DA OPERAÇÃO:

Operação	Frequência	Conceito	R (Retrieval), U (Update), I (Insert) ou D (Delete)	Média de ocorrências acessadas
01	500 vezes ao dia	Livro	U	500

CUSTOS:

O custo para inserção é, no pior caso, de um acesso a disco, já que esta será feita de forma sequencial.

Como isto ocorre 500 vezes ao dia, o custo total é de 1 * 500 = **500** acessos.

2. Consultar livros, através do seu Nome, Código ISBN ou Autor. Isto ocorre 1.000 vezes ao dia.

ESQUEMA DE NAVEGAÇÃO:

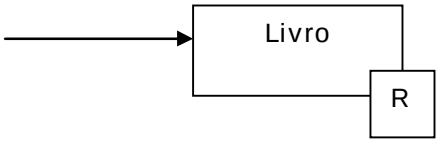

TABELA DE VOLUME DE ACESSO DA OPERAÇÃO:

Operação	Frequência	Conceito	R (Retrieval), U (Update), I (Insert) ou D (Delete)	Média de ocorrências acessadas
02	1.000 vezes ao dia	Livro	R	3.000

$R1 = \sigma_{ISBN=001}$ (Livro)

$R2 = \pi_{CodLivro, Titulo, Autor, Editora, NumeroExemplares}$ (R1)

CUSTOS:

R1: ISBN é chave, e supondo que ele seja usado em um índice primário, que tenha altura x = 3:

x + 1 = 3 + 1 = 4

Contando os acessos por dia tem-se 4* 1.000 = **4.000**

R2: Como o retorno esperado é de uma tupla, temos para o R2 o custo como sendo o número de blocos de Livro, ou seja, 3.000.

Como isso ocorre 1.000 vezes ao dia, o custo é de 3.000 * 1.000 = **3.000.000** acessos

Portanto, tem-se que o custo total da consulta é Custo de R1 + Custo de R2 = **4.000 + 3.000.000** = **3.004.000** acessos

e) Possibilidade de redução do espaço em disco alterando-se (otimizando) apenas a modelagem e a utilização de tipos de dados distintos

Quanto mais exemplares o livro tiver, maior é o consumo em disco, pois haverá dois atributos novos na tabela para cada exemplar, e atributos que não fazem parte de um objeto terão espaço em disco reservado que será preenchido com valor nulo quando não houver relação com o valor a que se refere à tupla. Por exemplo, se o livro que mais tiver exemplares na biblioteca tiver 10 exemplares, a tabela do banco de dados deve ser criada com 10 conjuntos de atributos que descrevem o exemplar (numeroExemplar_xx e situacaoEmprestimo_xx). Se outro livro tiver apenas 1 exemplar na biblioteca, ele estará cadastrado no sistema também com 10 conjuntos de atributos mas com apenas um preenchido e os demais nove estarão preenchidos com valor nulo.

Uma opção na mudança na modelagem é a exclusão dos atributos que contemplam o número do exemplar (numeroExemplar_xx), pois como a modelagem é feita toda em uma única tabela, tem-se que o valor armazenado no atributo de situação de empréstimo já informaria a qual exemplar se refere, de acordo com o número contido no próprio nome do atributo. Assim, situacaoExemplar_01 se referencia ao primeiro exemplar, situacaoExemplar_02 se referencia ao segundo exemplar, e assim por diante.

Uma segunda opção para redução de espaço em disco é trocar o tipo do atributo de situação de empréstimo para um tipo que ocupe menos espaço, como o caso de um booleano ou um caractere de 1 posição, gravando ‘E’ para emprestado, ‘D’ para disponível e ‘A’ para status de atraso de devolução, por exemplo.

f) Flexibilidade e Custo de Manutenção

Haverá inserção de novos atributos quando houver a disponibilização de mais um exemplar do livro que tiver a maior quantidade de exemplares e que está com todos os atributos do banco de dados preenchidos. A manutenção neste caso é simples e se dá com a criação dos atributos que descrevem o exemplar (numeroExemplar_xx e situacaoEmprestimo_xx) na tabela. Novos atributos relacionados ao livro também são de simples inclusão, em uma única tabela.

4.2.2. Agregação em Chave Estrangeira

Considerando outra parte da modelagem do Sistema ACERVUS, descrito no item 4.1.1, seja a **Figura 16**, que contém mais especificamente a parte referente aos dados de livros e seus exemplares e é utilizada para exemplificar este padrão.

E supondo as seguintes consultas frequentes:

1. Retirar um exemplar. Isto ocorre 500 vezes ao dia.
2. Consultar livros, através do seu Nome, Código ISBN ou Autor. Isto ocorre 1.000 vezes ao dia.

A partir do modelo ilustrado na **Figura 16** a aplicação do padrão Agregação em Chave Estrangeira é realizada e ilustrada na **Figura 18**, com a criação de tabelas separadas para cada classe envolvida na agregação. Os atributos são as colunas das tabelas.

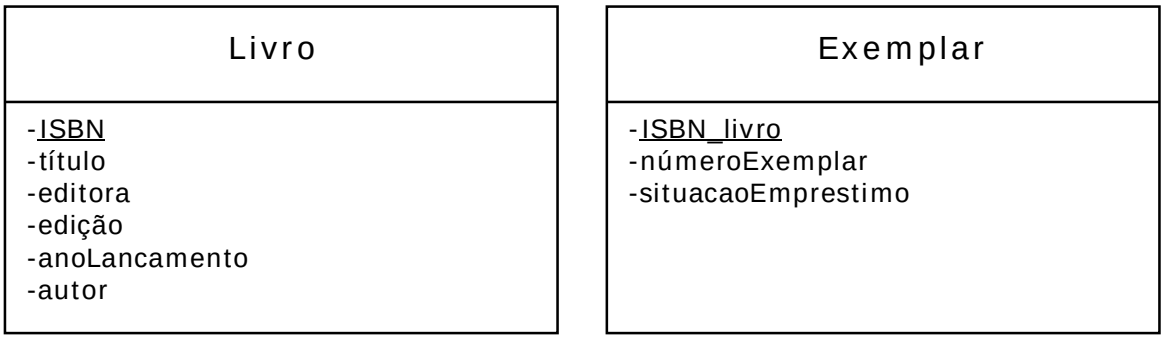

Figura 18. Tabela para o Caso Agregação com Chave Estrangeira

No que se refere ao tamanho dos atributos, valem as mesmas considerações do item 4.1.1.

A análise das métricas é feita a seguir.

a) Possibilidade de desnormalização para agilidade de consultas

O resultado da aplicação do padrão Agregação em Chave Estrangeira precisa de uma operação de junção ou, pelo menos, dois acessos de dados ao ponto que a modelagem definida pelo padrão Agregação em Tabela Única precisa de uma única operação de banco de dados. Se acessar objetos agregados é um caso estatisticamente raro no projeto, isso é aceitável. Se os objetos agregados são sempre recuperados, juntamente com o objeto de agregação, é necessária uma melhor avaliação para estas questões de desempenho.

Deste modo, a desnormalização é uma opção válida para aplicar-se a este padrão, dado que é possível combinar colunas das duas tabelas em uma tabela, ligando previamente as informações. Sendo assim, poder-se-ia unir todos os atributos em uma tabela e no caso da agregação, não haveria ligações necessárias. Com isso, temos como resultado um modelagem tal como a da **Figura 17**, da aplicação do padrão “Agregação em Tabela Única”.

b) Armazenamento lógico / Estratégia de indexação

A análise da forma de armazenamento e indexação é a mesma contida no item 4.1.1.

c) Possibilidade ou necessidade de alta disponibilidade e uso de métodos de replicação

A partir das consultas principais relatadas, pode haver necessidade de alta disponibilidade, principalmente imaginando um crescimento no número de usuários ou livros adquiridos pela biblioteca. O uso do sistema para consultas de extração de relatórios não necessita uma alta disponibilidade, mas as operações de retirada e devolução de livros que requerem acesso rápido ao sistema podem ficar lentas conforme a quantidade de dado inserido no sistema, sem falar em questões de usuários concorrentes ou limitações de rede.

d) Custo e Otimização da(s) consulta(s) descrita(s) como sendo a(s) principal(is) consulta(s) realizada(s) na base de dados

Os números apresentados para os cálculos a seguir estão descritos no Quadro 3, página 50.

Tabela Livro	Tamanho da tupla: **tamt** = 25 + 30 + 30 + 4 + 4 + 30 = 123 bytes Número médio de tuplas: **nt** = 75.000 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/123⌉ = 34 Número de blocos (nt / ftb): **nb** = ⌈75.000/34⌉ = 2.206
Tabela Exemplar	Tamanho da tupla: **tamt** = 25 + 4 + 10 = 39 bytes Número médio de tuplas: **nt** = 150.000 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/39⌉ = 106 Número de blocos (nt / ftb): **nb** = ⌈150.000/106⌉ = 1.416

1. Retirar um exemplar. Isto ocorre 500 vezes ao dia.

ESQUEMA DE NAVEGAÇÃO:

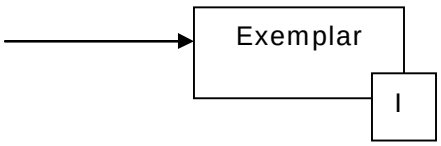

TABELA DE VOLUME DE ACESSO DA OPERAÇÃO:

Operação	Frequência	Conceito	R (Retrieval), U(Update), I(Insert) ou D(Delete)	Média de ocorrências acessadas
01	500 vezes ao dia	Exemplar	U	500

ÁLGEBRA RELACIONAL:

$R \leftarrow R \cup$ **(**ISBN, numeroExemplar, situacaoExemplar)

CUSTOS:

O custo para inserção é, no pior caso, de um acesso a disco, já que esta será feita de forma sequencial.

Como isto ocorre 500 vezes ao dia, o custo total é de 1 * 500 = **500** acessos.

2. Consultar livros, através do seu Nome, Código ISBN ou Autor. Isto ocorre 1000 vezes ao dia.

ESQUEMA DE NAVEGAÇÃO:

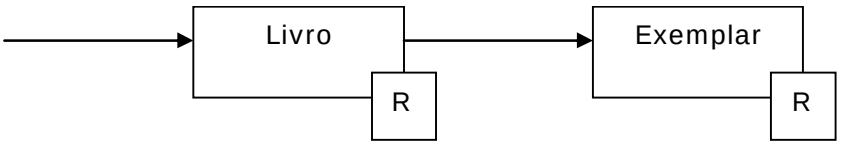

TABELA DE VOLUME DE ACESSO DA OPERAÇÃO:

Operação	Frequência	Conceito	R (Retrieval), U(Update), I(Insert) ou D(Delete)	Média de ocorrências acessadas
02	1.000 vezes ao dia	Livro Exemplar	R R	1.000 2.000

ÁLGEBRA RELACIONAL:

$R1 = \sigma_{Codlivro=001}$ (Livro)

$R2 = R1 \ _{(CodLivro=CodLivro)} |\times|$ Exemplar

$R3 = \pi_{CodLivro, Titulo, Autor, Editora, NumeroExemplares}$ (R2)

CUSTOS:

R1:

CodLivro é chave, e supondo que ele seja usado em um índice primário, que tenha altura x = 3:

x + 1 = 3 + 1 = 4

4* 1.000 = **4.000**

R2:

Como o retorno esperado é de uma tupla, tem-se para o R1:

R2	Tamanho da tupla: **tamt** = todos os atributos de Livro somados ao de Exemplar = 123 + 39 = 162 bytes Número médio de tuplas: **nt** = 1 Fator tamanho de bloco (tb / tamt): **ftb** = $\lceil 4.096/163 \rceil = 26$ Número de blocos (nt / ftb): **nb** = $\lceil 1/26 \rceil = 1$

Assim, tem-se o custo de R2 como sendo:

nbR1 + (nbR1 * nbExemplar) = 1 + (1 * 1.416) = 1.417 acessos

1.417 * 1.000 = **1.417.000**

R3:

Como um livro possui, em média, 2 exemplares, supõe-se que o retorno seja de 2 tuplas e tem-se para R3:

R3	Tamanho da tupla: **tamt** = todos os atributos de R2 (Livro) somados ao de Exemplar = 123 + 39 = 162 bytes Número médio de tuplas: **nt** = 2 Fator tamanho de bloco (tb / tamt): **ftb** = $\lceil 4.096/163 \rceil = 26$ Número de blocos (nt / ftb): **nb** = $\lceil 2/26 \rceil = 1$

Assim, o custo é o acesso ao bloco em que está o resultado, que é de 1 acesso.

1 * 1.000 = **1.000**

Portanto, tem-se que o custo total da consulta é Custo de R1 + Custo de R2 + Custo de R3 = **1.417.000 + 1.000 = 1.418.000** acessos

e) Possibilidade de redução do espaço em disco alterando-se (otimizando) apenas a modelagem e a utilização de tipos de dados distintos

Uma opção para redução de espaço em disco é trocar o tipo do atributo de situação de empréstimo para um tipo que ocupe menos espaço, como o caso de um booleano ou um caractere de 1 posição, gravando 'E' para emprestado, 'D' para disponível e 'A' para status de atraso de devolução, por exemplo.

f) Flexibilidade e Custo de Manutenção

Haverá inserção de novos atributos quando novos atributos relacionados ao livro forem adotados. Estes são de simples inclusão, ocorrendo apenas em uma única tabela. O mesmo acontece para o caso de haver novos atributos relacionados ao exemplar.

4.3. Padrões de mapeamento de Associação

4.3.1. Associação 1:n em Chave Estrangeira

Considerando outra parte da modelagem do Sistema ACERVUS, descrito no item 4.1.1, seja a **Figura 19** que contém mais especificamente a parte referente aos Usuários que pegam Exemplares e que será utilizada para exemplificar este padrão.

E supondo as seguintes consultas frequentes:

1. Impressão de relatórios de empréstimos feitos, contendo o nome de quem pegou emprestado, código do usuário, ISBN do livro e título.

A partir do modelo ilustrado na **Figura 19**, a aplicação do padrão "Associação 1:n em Chave Estrangeira" é realizada e ilustrada na **Figura 20**, com a criação de

tabelas separadas para cada classe da hierarquia da herança. Os atributos são as colunas das tabelas.

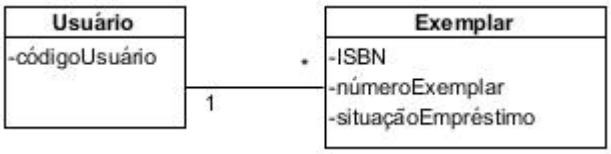

Figura 19. Diagrama de Classes para associação de Usuários com Exemplares

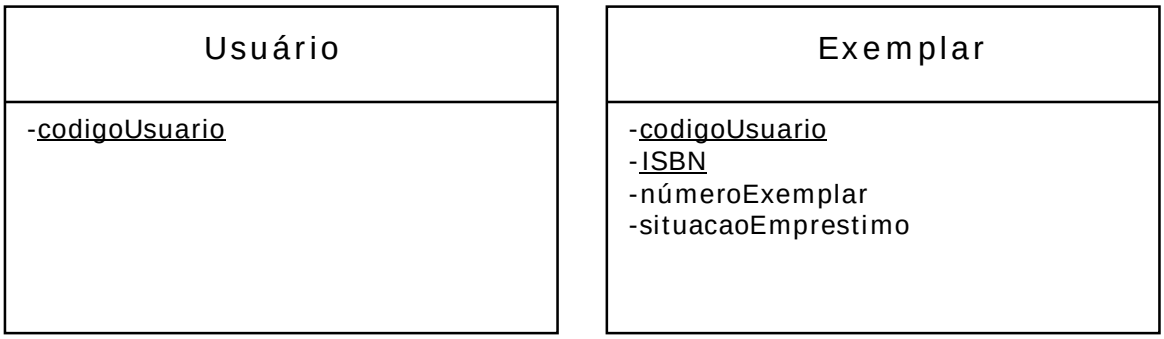

Figura 20. Tabela para o Caso Associação 1:n com Chave Estrangeira

No que se refere ao tamanho dos atributos, valem as mesmas considerações do item 4.1.1.

A análise das métricas é feita a seguir.

a) Possibilidade de desnormalização para agilidade de consultas

A desnormalização é uma opção válida para aplicar-se a este padrão, dado que com ela é possível combinar colunas de duas ou mais tabelas distintas na mesma tabela, ligando previamente as informações. Sendo assim, poder-se-ia unir todos os atributos em uma tabela e no caso da associação, não haveria ligações necessárias. Com isso, tem-se como resultado uma modelagem em que há apenas uma tabela, de Exemplar, adicionando à original o atributo de código do usuário. Assim, empréstimos seriam gravados no sistema na tabela de Exemplar.

b) Armazenamento lógico / Estratégia de indexação

A análise da forma de armazenamento e indexação é a mesma contida no item 4.1.1.

c) Possibilidade ou necessidade de alta disponibilidade e uso de métodos de replicação

A partir das consultas principais relatadas, não é necessário alta disponibilidade pois o uso desta parte do sistema não implica em consultas que necessitam retorno imediato. As consultas são apenas para extração de relatórios.

d) Custo e Otimização da(s) consulta(s) descrita(s) como sendo a(s) principal(is) consulta(s) realizada(s) na base de dados

Os números apresentados para os cálculos a seguir estão descritos no Quadro 3, página 50.

Tabela Exemplar	Tamanho da tupla: **tamt** = 4+ 25 + 4 + 10 = 43 bytes Número médio de tuplas: **nt** = 150.000 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/43⌉ = 96 Número de blocos (nt / ftb): **nb** = ⌈150.000/96⌉ = 1.563

1. Impressão de relatórios de empréstimos feitos, contendo o nome de quem pegou emprestado, código do usuário, ISBN do livro e título.

ESQUEMA DE NAVEGAÇÃO:

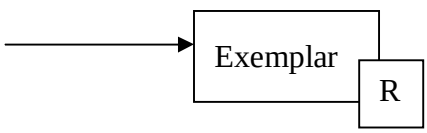

TABELA DE VOLUME DE ACESSO DA OPERAÇÃO:

Operação	Frequência	Conceito	R (Retrieval), U(Update), I(Insert) ou D(Delete)	Média de ocorrências acessadas
01	1 vez ao dia	Exemplar	R	1

ÁLGEBRA RELACIONAL:

$R1 = \Pi_{nome, ISBN}$ (Exemplar)

CUSTOS:

O custo é o acesso aos blocos de toda tabela, ou seja, de **1.563** acessos.

e) Possibilidade de redução do espaço em disco alterando-se (otimizando) apenas a modelagem e a utilização de tipos de dados distintos

Uma opção para redução de espaço em disco é trocar o tipo do atributo de situação de empréstimo para um tipo que ocupe menos espaço, como o caso de um booleano ou um caractere de 1 posição, gravando ‘E’ para emprestado, ‘D’ para disponível e ‘A’ para status de atraso de devolução, por exemplo.

f) Flexibilidade e Custo de Manutenção

Haverá inserção de novos atributos quando novos atributos relacionados ao usuário ou ao exemplar forem adotados. Estes são de simples inclusão, ocorrendo apenas em uma única tabela.

Caso a cardinalidade mude, haverá a inserção de mais uma entidade no modelo, fazendo com que o padrão “Associação m:n em Tabela” seja aplicado.

4.3.2. Associação m:n em Tabela

Considerando outra parte da modelagem do Sistema ACERVUS, descrito no item 4.1.1, seja a **Figura 19** que contém mais especificamente a parte referente aos Usuários que pegam Exemplares e que será utilizada para exemplificar este padrão.

E supondo as seguintes consultas frequentes:

1. Impressão de relatórios de empréstimos feitos, contendo o nome de quem pegou emprestado, código do usuário, ISBN do livro e título.

A partir do modelo ilustrado na **Figura 19**, a aplicação do padrão “Associação m:n em Tabelas” é realizada e ilustrada na **Figura 21**, com a criação de tabelas separadas para cada classe da hierarquia da herança. Os atributos são as colunas das tabelas.

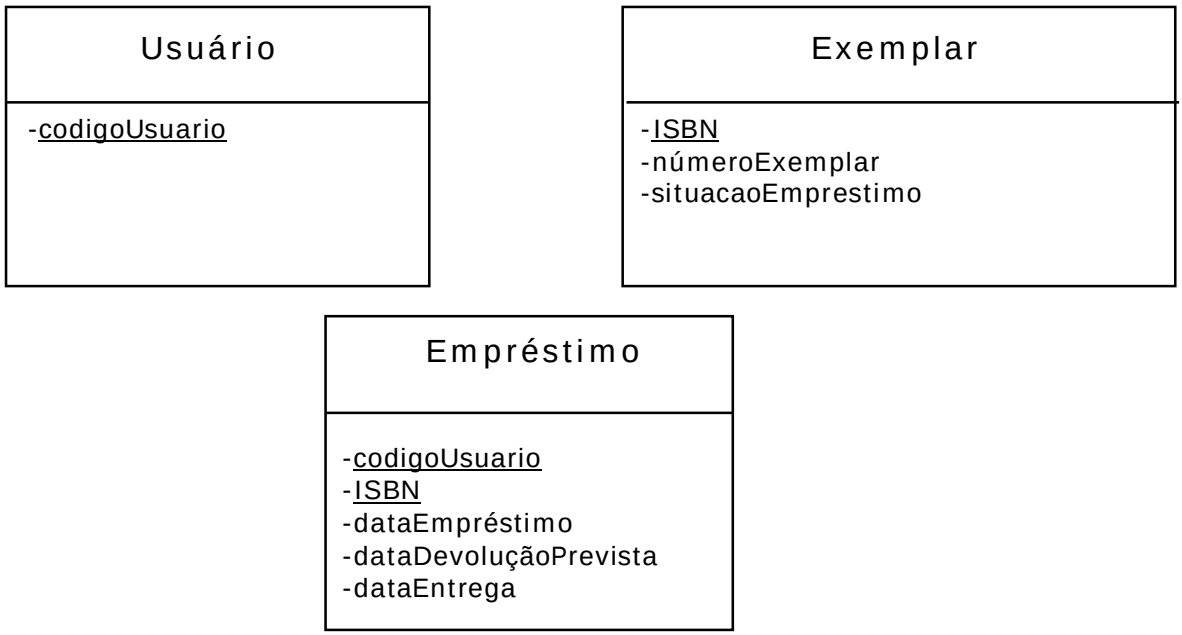

Figura 21. Tabela para o Caso Associação m:n em Tabelas

No que se refere ao tamanho dos atributos, valem as mesmas considerações do item 4.1.1.

A análise das métricas é feita a seguir.

a) Possibilidade de desnormalização para agilidade de consultas

A desnormalização é uma opção válida para aplicar-se a este padrão, dado que com ela é possível combinar colunas de duas ou mais tabelas distintas na mesma tabela, ligando previamente as informações. Sendo assim, poder-se-ia unir todos os atributos em uma tabela e no caso da associação, não haveria ligações necessárias. Com isso, temos como resultado uma modelagem em que há apenas uma tabela, de Exemplar, adicionando à original os atributos de código do usuário quantas vezes necessário, isto é, quantas vezes o exemplar de livro existir. Assim, empréstimos seriam gravados no sistema na tabela de Exemplar.

b) Armazenamento lógico / Estratégia de indexação

A análise da forma de armazenamento e indexação é a mesma contida no item 4.1.1.

c) Possibilidade ou necessidade de alta disponibilidade e uso de métodos de replicação

A partir das consultas principais relatadas, não é necessário alta disponibilidade pois o uso desta parte do sistema não implica em consultas que necessitam retorno imediato. As consultas são apenas para extração de relatórios.

d) Custo e Otimização da(s) consulta(s) descrita(s) como sendo a(s) principal(is) consulta(s) realizada(s) na base de dados

Os números apresentados para os cálculos a seguir estão descritos no Quadro 3, página 50.

Tabela Usuário	Tamanho da tupla: **tamt** = 4 bytes Número médio de tuplas: **nt** = 71.500 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/4⌉ = 1.024 Número de blocos (nt / ftb): **nb** = ⌈71.500/1.024⌉ = 70

Tabela Exemplar	Tamanho da tupla: **tamt** = 25 + 4 + 10 = 39 bytes Número médio de tuplas: **nt** = 150.000 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/39⌉ = 106 Número de blocos (nt / ftb): **nb** = ⌈150.000/106⌉ = 1.416
Tabela Empréstimo	Tamanho da tupla: **tamt** = 4 + 4 + 3 + 3 + 3 = 17 bytes Número médio de tuplas: **nt** = 30.000 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/17⌉ = 241 Número de blocos (nt / ftb): **nb** = ⌈30.000/241⌉ = 125

1. Impressão de relatórios de empréstimos feitos, contendo o nome de quem pegou emprestado, código do usuário, ISBN do livro e título.

ESQUEMA DE NAVEGAÇÃO:

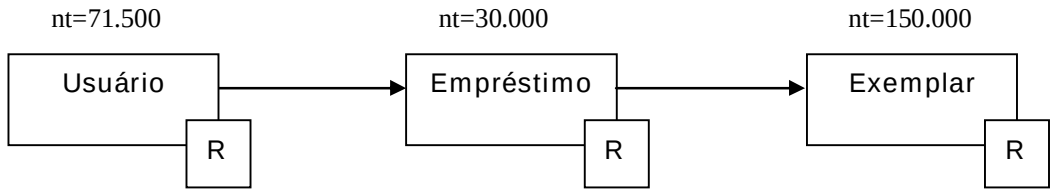

TABELA DE VOLUME DE ACESSO DA OPERAÇÃO:

Operação	Frequência	Conceito	R (Retrieval), U(Update), I(Insert) ou D(Delete)	Média de ocorrências acessadas
01	1 vez por semana	Usuário	R	1
		Empréstimo	R	1
		Livro	R	1

ÁLGEBRA RELACIONAL:

R1 = Empréstimo $_{(CodUsuario=CodUsuario)}$ |X| Usuario

R2 = R1 $_{(CodLivro=CodLivro)}$ |X| Exemplar

R3 = $\Pi_{Nome, CodigoUsuario, DataEmprestimo, DataPrevistaDevolucao, Titulo, CodLivro, NumeroExemplar}$ (R2)

CUSTOS:

R1:

Supondo que o retorno seja de 90.000 tuplas, isto é, as 30.000 existentes em Empréstimo cruzadas com 3.000 de Usuários, que são as pessoas com empréstimo no momento, têm-se para o R1:

R1	Tamanho da tupla: **tamt** = todos os atributos de Empréstimo somados aos de Usuário = 17 + 4 = 21 bytes Número médio de tuplas: **nt** = 90.000 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/21⌉ = 196 Número de blocos (nt / ftb): **nb** = ⌈90.000/196⌉ = 460

nbEmpréstimo + (nbEmpréstimo * nbUsuário)

125 + (125 * 70)

8.875 acessos

R2:

Supondo que o retorno seja de 30.000 tuplas, isto é, as 30.000 existentes em Empréstimo, que é exatamente o que a consulta pede, temos para o R2:

R2	Tamanho da tupla: **tamt** = todos os atributos de R1 (Empréstimo + Usuário) somados aos de Exemplar = 21 + 39 = 60 bytes Número médio de tuplas: **nt** = 30.000 Fator tamanho de bloco (tb / tamt): **ftb** = ⌈4.096/60⌉ = 69 Número de blocos (nt / ftb): **nb** = ⌈30.000/69⌉ = 435

nbR1 + (nbR1 * nbExemplar)

460 + (460 * 1.416)

651.820 acessos

R3:

O custo é o acesso a todos os blocos de R2, que é de **435** acessos.

Portanto, tem-se que o custo total da consulta é Custo de R1 + Custo de R2 + Custo de R3 = **8.875 + 651.820 + 435 = 661.130** acessos

e) Possibilidade de redução do espaço em disco alterando-se (otimizando) apenas a modelagem e a utilização de tipos de dados distintos

Uma opção para redução de espaço em disco é trocar o tipo do atributo de situação de empréstimo para um tipo que ocupe menos espaço, como o caso de um booleano ou um caractere de 1 posição, gravando ‘E’ para emprestado, ‘D’ para disponível e ‘A’ para status de atraso de devolução, por exemplo.

f) Flexibilidade e Custo de Manutenção

Haverá inserção de novos atributos quando novos atributos relacionados ao usuário ou ao exemplar forem adotados. Estes são de simples inclusão, ocorrendo apenas em uma única tabela. Atributos do relacionamento são adicionados na tabela de Empréstimo, como o caso de data do empréstimo e data de devolução.

4.4. Comparação de Padrões

A seguir há três quadros de comparação das métricas avaliadas no Capítulo 4 para cada tipo de padrão, isto é, padrões de mapeamento de Herança, de Agregação e de Associação.

Para as tabelas de comparação das avaliações de padrões foi adotada a seguinte métrica para os custos de consultas:

- ✓ até 10.000 acessos – custo muito baixo;
- ✓ até 50.000 acessos – custo baixo;
- ✓ até 250.000 acessos – custo médio;
- ✓ até 1.000.000 acessos – custo alto;
- ✓ acima de 1.000.000 acessos – custo muito alto.

a) Herança

Padrão / Métrica	Herança de Classes em Tabela Única	Herança de Classes em Tabelas para cada Classe	Herança de Classes em Tabelas para cada Classe Concreta
Desnormalização	Não se aplica.	É uma opção, considerando que a leitura de uma classe derivada tem tantas chamadas ao banco quantos forem o nível dela na hierarquia.	É uma opção caso uma consulta principal requeira obter um objeto da Classe Base, pois seria necessário acessar todas as tabelas para isso.
Indexação	Recomendável caso haja consultas online direcionadas ao retorno de dados do usuário. Não há necessidade para geração de relatórios.	Idem padrão Herança de Classes em Tabela Única.	Idem padrão Herança de Classes em Tabela Única.
Replicação	Não há necessidade.	Não há necessidade.	Não há necessidade.
Custo de consulta	C1: muito baixo (4.412 acessos) C2: baixo (11.560 acessos)	C1: médio (101.596 acessos) C2: muito alto (1.991.454 acessos)	C1: muito baixo (135 acessos) C2: muito baixo (2.597 acessos)
Espaço em disco	Maior nível hierárquico, maior consumo. Opção de alterar tipo de dados.	Não é alto, pois atributos são preenchidos apenas quando realmente pertencem a um objeto. Opção de alterar tipo de dados.	Não é alto, pois atributos são preenchidos apenas quando realmente pertencem a um objeto. Opção de alterar tipo de dados.
Manutenção	Inserção de níveis hierárquicos demandam inserção de atributos na tabela.	Inserção de níveis hierárquicos demandam criação de uma nova tabela com seus atributos.	Inserção de níveis hierárquicos demandam criação de uma nova tabela com seus atributos.

Quadro 4. Comparativo de Padrões de Herança

Especificamente neste estudo de caso a aplicação das seis métricas de avaliação aos padrões de Herança descritos e analisados nessa dissertação, tem-se uma maior discrepância dos valores avaliados no Custo das Consultas. Quando são comparadas as métricas de Desnormalização, Replicação e Manutenção, há pouca ou nenhuma variação entre cada padrão. Porém quando se analisa o Espaço em Disco já se nota uma diferença do padrão de Herança de Classes em Tabela Única em comparação aos de Classes em Tabelas para cada Classe e de Classes em Tabelas para cada Classe Concreta. Mas a diferença mais significativa é a com relação ao custo das consultas frequentes. Esta métrica determina que o pior padrão a ser implementado é o segundo, que mapeia a Herança de Classes em Tabelas para cada Classe e tem a quantidade de acessos 700 vezes maior que o melhor caso avaliado, que é o de mapear a Herança de Classes em Tabelas para cada Classe Concreta.

b) Agregação

Padrão / Métrica	Agregação em Tabela Única	Agregação em Chave Estrangeira
Desnormalização	Não se aplica.	É uma opção, pois a agregação precisa de uma operação de junção ou, pelo menos dois acessos de dados.
Indexação	Recomendável caso haja consultas online direcionadas ao retorno de dados do usuário. Não há necessidade para geração de relatórios.	Idem padrão Agregação em Tabela Única.
Replicação	Pode haver necessidade para operações em tempo real e podem ficar lentas conforme a quantidade de dados inseridos no sistema.	Idem padrão Agregação em Tabela Única.
Custo de consulta	C1: muito baixo (500 acessos) C2: muito alto (3.004.000 acessos)	C1: muito baixo (500 acessos) C2: muito alto (1.418.000 acessos)
Espaço em disco	Quanto mais agregados o agregador, maior é o consumo em disco, pois haverá atributos novos na tabela para cada exemplar, e atributos que não fazem parte de um objeto terão espaço em disco reservado e serão preenchidos com valor nulo quando não houver relação com o valor a que se refere à tupla. Opção de mudança na modelagem e de troca de tipo de dados dos atributos.	Opção de troca de tipo de dados dos atributos.
Manutenção	Mudança quando se inserem novos atributos.	Mudança quando inserem novos atributos.

Quadro 5. Comparativo de Padrões de Agregação

Similar à análise das comparações dos padrões de mapeamento de herança no caso dos padrões de agregação, quando são comparadas as métricas de Desnormalização, Replicação e Manutenção, tem-se pouca ou nenhuma alteração. Porém quando se analisa o Espaço em Disco vê-se que o padrão Agregação em Tabela Única requer mais espaço em disco do que o padrão que agrega, com uso de Chaves Estrangeiras. Isso se deve à alocação de espaço para atributos conforme o maior caso que houver de agregação, ou seja, quanto mais agregados uma tabela tiver, mais atributos são criados para a tabela em questão, o que demanda muito espaço de armazenamento. A diferença com relação aos custos de consultas é divergente, sendo o padrão de chaves estrangeira aproximadamente 50% mais rápido. No entanto, vale lembrar que as consultas dadas não são consideradas válidas para um ambiente produtivo, elas apenas valem como guia para a próxima discussão sobre o assunto.

c) Associação

Padrão / Métrica	Associação 1:n em Chave Estrangeira	Associação m:n em Tabela
Desnormalização	É uma opção, pois ao unir todos os atributos em uma tabela não haverá necessidade de junção.	É uma opção, pois ao unir todos os atributos em uma tabela não haverá necessidade de junção.
Indexação	Recomendável caso haja consultas online direcionadas ao retorno de dados do usuário. Não há necessidade para geração de relatórios.	Idem padrão Associação 1:n em Chave Estrangeira.
Replicação	Para o exemplo dado e consultas relatadas, não há necessidade.	Para o exemplo dado e consultas relatadas, não há necessidade.
Custo de consulta	C1: muito baixo (1.563 acessos)	C1: alto (661.130 acessos)
Espaço em disco	Opção de troca de tipo de dados dos atributos.	Opção de troca de tipo de dados dos atributos.
Manutenção	Mudança quando insere novos atributos. Caso a cardinalidade mude, haverá a inserção de mais uma entidade no modelo, fazendo com que o padrão “Associação m:n em Tabela” seja aplicado.	Mudança quando insere novos atributos.

Quadro 6. Comparativo de Padrões de Associação

Comparando-se as métricas de Desnormalização, Replicação, Manutenção, e até mesmo a de Espaço em Disco sobre o resultado da aplicação dos padrões Associação 1:n em Chave Estrangeira e Associação m:n em Tabela, nota-se que os resultados são muito parecidos. A diferença com relação aos custos de consultas é alta, sendo o segundo padrão aproximadamente 400 vezes mais custoso que o primeiro.

Como se vê, o custo da consulta é o que mais varia com relação ao desempenho do modelo de banco de dados proposto. Assim, quando modelar um novo banco de dados se faz necessário avaliar qual o uso que este banco terá para que as consultas frequentes possam ser levantadas e a modelagem do BD seja então criada de modo a otimizar o desempenho destas consultas principais.

5 CONCLUSÕES

Neste Capítulo são fundamentados os resultados e uma correspondência entre as conclusões e os objetivos propostos é realizada.

5.1. Contribuições

A primeira contribuição desta dissertação consiste na organização e sistematização de passos a serem seguidos para avaliar a modelagem de um banco de dados relacional modelado a partir de um requisito ou de um diagrama de classes. Esta dissertação oferece uma visão prática da aplicação da teoria de padrões de mapeamento e da avaliação dos mesmos.

Outra contribuição importante foi a proposição das métricas a serem avaliadas para cada um dos padrões, tomando-se como principal premissa o fato destas métricas representarem os maiores riscos de sucesso de um projeto assim como, independentemente do embasamento teórico, elas já serem fatores de análise naturais de análise do administrador de banco de dados quando da implementação de um banco de dados propriamente dito.

Dentre todas as métricas avaliadas para todos os padrões, concluiu-se que o processamento de consultas é o que mais diverge, sendo ele o principal quesito a ser avaliado quando se deseja implementar um banco de dados relacional com alto desempenho. Enquanto as outras métricas avaliadas são mais subjetivas, a métrica de custo de acesso a dados é quantitativa e pode variar muito conforme as consultas principais elencadas pelo operador do sistema e cujas análises devem ser feitas pelo analista responsável pela modelagem do banco de dados. Esta análise visa a otimização da modelagem, bem como da consulta a ser feita no banco de dados para obtenção do resultado que o operador espera.

Não é possível dizer, no entanto, que a otimização de um código SQL para realização de uma consulta ao banco de dados, por exemplo, é o principal responsável pelo melhor ou pior desempenho. Esta dissertação não trata de questões de *hardware* e métodos de armazenamento físico dos dados, que podem ser críticos para o acesso ao banco de dados. O projeto físico de um banco de dados é algo que deve ser avaliado com muito critério para que o desempenho seja

tratado da forma como seja necessário. Isso envolve, por exemplo, o uso de um BD distribuído, com a utilização de particionamento horizontal ou vertical, em que no primeiro caso, os registros são separados e no segundo há uma quebra por colunas.

As avaliações das métricas descritas nesta dissertação podem ser aplicadas a qualquer Banco de Dados, indistintamente, pois elas se baseiam exclusivamente na modelagem e na implementação de um banco de dados e não se detêm em questões específicas de gerenciamento de banco de dados ou otimização que determinados SGBDs possam realizar quando são executadas consultas a sua base de dados.

5.2. Trabalhos Futuros

Esta dissertação considerou sete diferentes métricas de análises de padrões: possibilidade de desnormalização, método para armazenamento lógico alinhado à estratégia de indexação, armazenamento físico, necessidade de alta disponibilidade e uso de métodos de replicação, custo e otimização de consultas, espaço em disco e, por fim, flexibilidade e custo de manutenção.

Cada um desses itens pode ser fragmentado e detalhado com mais riqueza de informações. Assim, trabalhos futuros a esta dissertação poder-se-iam focar individualmente em cada item, como por exemplo:

- Detalhar os diferentes tipos de armazenamento lógico que não foram descritos nesta dissertação e o funcionamento de um sistema e seu banco de dados com operações de inserção, atualização e exclusão em arranjos, árvore binária, árvore binária de busca, árvore de busca binária auto-balanceada, árvore B^+ e tabela hash.
- Fixando-se um modo de armazenamento lógico, os algoritmos de ordenação também influenciam diretamente no desempenho das quatro funções básicas do banco de dados: criar, ler, atualizar, excluir (CRUD, do inglês "*Create, Read, Update and Delete*"). Uma opção seria aplicar ou comparar os métodos descritos nos *benchmarks* de algoritmos de ordenação (SORT, 10) a um modelo criado através da aplicação dos padrões de um mesmo tipo de relacionamento.

- Automatização da análise das métricas de avaliação aplicada a cada padrão de modelagem.
- Avaliação de métodos de armazenamento físico que possam ser utilizados para melhorar a utilização, manutenção e desempenho do banco de dados com a aplicação dos padrões. Por exemplo, o uso de particionamento horizontal de uma tabela única de histórico, separando suas tuplas por intervalos de tempo, em que os mais recentes e mais utilizados estariam juntos em uma tabela reduzida. Ou então, o uso de particionamento vertical, em que campos do tipo texto longo ou BLOB ficariam igualmente separados do restante da tabela.

REFERÊNCIAS

AGUIRRE-URRETA, M., MARAKAS, G. Comparing Conceptual Modeling Techniques: A Critical Review of the EER vs. OO Empirical Literature.The DATA BASE for Advances in Information Systems. Volume 39, Number 2, May 2008.

ALEXANDER, ET AL. A Pattern Language. Oxford, 1977.

AMBLER, S. Mapping Objects to Relational Databases: O/R Mapping In Detail. 1997. Disponível em <http://www.agiledata.org/essays/mappingObjects.html>. Acesso em outubro de 2008.

AMBLER, Scott. The Fundamentals of Mapping Objects to Relational Databases. 2003. Disponível em <http://www.agiledata.org>. Acesso em março de 2008.

AVIGAD, J., DONNELLY, K.Formalizing O notation in Isabelle/HOL. Second international joint conference, IJCAR 2004.

BATINI, C; CERI, S; NAVATHE, S.B. Conceptual database design: an Entity-relationship approach, Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, 1991

BOCK, D., RYAN, T. Accuracy in modeling with extended entity relationship and object oriented data models. Journal of Database Management. Hershey: Fall 1993. Vol. 4, Iss. 4; pg. 30, 10 pgs.

BROWN, Kyle, et al. Enterprise Java Programming with IBM Websphere. Addison Wesley, second edition, December 2003.

CHAN, H., TEO, H., ZENG, X. An Evaluation of Novice End-User Computing Performance: Data Modeling, Query Writing, and Comprehension. Journal of the American Society for Information Science and Technology. 2005. Vol. 56, No. 8, pp. 843-853.

DE LUCIA, A.; GRAVINO, C.; OLIVETO, R.; TORTORA, G. Assessing the Support of ER and UML Class Diagrams during Maintenance Activities on Data Models. Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference on Volume. 2008. pp. 173 – 182.

FOWLER, Martin, et al. Patterns of Enterprise Application Architecture. Addison Wesley, 1.3 edition, November 2002.

GAMMA, Erich, et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley Professional, 1994

KELLER, Wolfgang. Mapping Objects to Tables: A Pattern Language. Washington University, Department of Computer Science - EuroPLoP 1997, 1997.

LEE, H., CHOI, B. A Comparative Study of Conceptual Data Modeling Techniques. Journal of Database Management. 1998. Vol. 9, No. 2, pp. 26- 35.

LIAO, C., PALVIA, P. The Impact of Data Models and Task Complexity on End-User Performance: An Experimental Investigation. International Journal of Human-Computer Studies, 2000. Vol. 52, pp. 831-845.

MYSQL – MySQL 5.5 Reference Manual. Disponível em <http://dev.mysql.com/doc/refman/5.5/en/>. Acesso em novembro de 2010.

Object Management Group, "OMG Unified Modeling Language Specification (Draft)", 1999. Disponível em < http://www.rational.com/uml/resources/documentation/OMG-UML-1.3.-Alpha2.pdf>. Acesso em setembro de 2008.

PALVIA, P.C., LIAO, C., TO, P. The impact of conceptual data models on end-user performance. Journal of Database Management, 3(4):4–15, 1992.

RUP - Rational Unified Process. Rational Software Corp. Disponível em <http://www.wthreex.com/rup/portugues/index.htm>. Acesso em abril de 2010.

SEINER, R.S. The Data Administration Newsletter – TDAN.com. April 2001. Disponível em <http://www.tdan.com>. Acesso em março de 2010.

SHOVAL, P., FRUMERMANN, I. OO and EER conceptual schemas: A comparison of user comprehension. Journal of Database Management, 5(4):28–38, 1994.

SHOVAL, P., SHIRAN, S. Entity-relationship and object-oriented data modeling experimental comparison of design quality Data & Knowledge Engineering 21. 1997. 297-315.

SINHA, A., VESSEY, I. An Empirical Investigation of Entity-Based and Object-Oriented Data Modeling: A Development Lifecycle Approach, Proceedings of the International Conference on Information Systems. 1999.

SORT Benchmark Home Page. Disponível em <http://sortbenchmark.org>. Acesso em agosto de 2010.

YUGOPUSPITO P., ARAKI K., Evolution of Relational Database to Object-Relational Database in Abstract Level. Proceedings of the InternationalWorkshop on Principles of Software Evolution, July 1999, pp 103-107.

REFERÊNCIAS COMPLEMENTARES

BERTOLAZZI, P. SCANNAPIECO, M. Introducing Data Quality in a Cooperative Context. Proc. 6th International Conference on Information Quality (ICIQ). 2001.

ELMASRI, R. NAVATHE, S. B. Fundamentals of Database Systems, Third Edition. Addison-Wesley, 2000. 838p.

FURLAN, J. D.; Modelagem de Objetos Através da UML, Makron Books, 1998.

GARCIA-MOLINA, H. Implementação de Sistemas de Bancos de Dados. Rio de Janeiro: Campus. 2001.

HEUSER, C. A. Projeto de Banco de Dados. Sagra Luzzatto, 2001. 254p.

ROTHERY, Brian. ISO 9000. São Paulo, Makron Books,1993.p.13

RUMBAUGH, J., JACOBSON, I., BOOCH, G. The Unified Modeling Language Reference Manual, Second Edition. Addison Wesley, 2004 . 752p.

RUMBAUGH, J.; OMT Insights: Perspective on Modeling from the Journal of Object-Oriented Programming, SIGS Books, 1996.

SILBERSCHATZ, A., KORTH, H.F, SUDARSHAN, S. Database System Concepts, Fourth Edition, McGraw Hill, 2001.