UNIVERSIDADE FEDERAL DE PELOTAS
INSTITUTO DE FÍSICA E MATEMÁTICA
CURSO DE BACHARELADO EM INFORMÁTICA

Definição de uma Base de Dados para o novo sistema da Biblioteca Setorial de Ciência e Tecnologia da UFPel

Leandro Fehn Fiss

Pelotas – RS
2001

LEANDRO FEHN FISS

Definição de uma Base de Dados para o novo sistema da Biblioteca Setorial de Ciência e Tecnologia da UFPel

Projeto de Conclusão de Curso apresentado ao Curso de Bacharelado em Informática, Instituto de Física e Matemática, Universidade Federal de Pelotas.
Orientador: Prof. Paulo Ricardo Porto, Msc. /UFPel
Co-orientador: Profa. Flávia Azambuja, Msc. /UFPel

PELOTAS
2001

Monografia defendida e aprovada, em 10 de janeiro de 2001, pela banca examinadora constituída pelos professores:

Prof. Paulo Ricardo Porto, MSc. – Orientador

Profa. Flávia Azambuja, MSc.

Prof. Fábio Zschornack.

O homem só conhece o seu verdadeiro valor,
quando ele ultrapassa as maiores dificuldades.

Dedico este trabalho aos meus pais Carlos Alberto e Marta,
aos meus irmãos Gabriela e Leonardo,
e a minha companheira Rocheli,
que são a razão de eu lutar para
ser uma pessoa melhor do que eu sou hoje.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho de conclusão de curso consisti em um estudo e posterior projeto de um sistema de bibliotecas para a Biblioteca Setorial de Ciência e Tecnologia (BS/CT) da Universidade Federal de Pelotas (UFPel). Sendo este parte integrante de um projeto maior, tem como finalidade principal analisar o sistema atual e especificar uma base de dados para um software que atenda as necessidades encontradas.

Para realização deste trabalho, foi utilizado a ferramenta de desenvolvimento Dr.CASE 3.0, o que traz uma maior confiabilidade ao trabalho realizado.

Ao final deste trabalho foram obtidos os script’s em linguagem SQL, para criação e manipulação da base de dados modelada durante o projeto.

1. INTRODUÇÃO

A Universidade Federal de Pelotas vem atualizando constantemente sua estrutura, mantendo-se sempre à frente das mais novas tecnologias do mundo contemporâneo, visando conduzir a sua função de centro de referência regional da melhor forma. Para tanto, diversas áreas da Universidade já foram informatizadas e interligadas em rede. Entretanto, apesar da Biblioteca Setorial de Ciência e Tecnologia construir uma área de extremo interesse para a UFPel devido sua importância entre alunos e professores, esta ainda não obteve uma significativa automatização.

O atual sistema da Biblioteca Setorial de Ciência e Tecnologia está baseado numa plataforma que impõe uma série de restrições e dificuldade quanto ao seu uso e manutenção. O seu banco de dados está depositado num computador *mainframe* cuja única forma de acesso remoto é através de terminais modo texto, que obedecem um padrão proprietário do fabricante. Além disso, o sistema atual está sendo sub-utilizado, visto que somente alguns módulos estão ativos.

A manutenção do sistema vem sendo prejudicada pois o mesmo foi escrito em linguagens praticamente obsoletas, o que exige um pessoal técnico, não-disponível na UFPel, para executá-la.

Como a informação é extremamente preciosa dentro de uma organização, existe a necessidade de reestruturação da base de dados do sistema da Biblioteca Setorial de Ciência e Tecnologia, para que a mesma esteja disponível conforme demanda.

Este trabalho visa desenvolver uma base de dados para um software que atenda as necessidades da Biblioteca Setorial de Ciência e Tecnologia, de acordo com as mais modernas metodologias de desenvolvimento, o que trará um significativo avanço tecnológico.

Para realização deste projeto, foi feito um levantamento (capítulo 4) das atuais necessidades e dos recursos existentes na Biblioteca Setorial de Ciência e Tecnologia, buscando definir um sistema que supra as necessidades encontradas e, que possa ser implementado na sua totalidade.

No capítulo 5 desta monografia é feita uma breve introdução sobre a administração de dados, dando ênfase aos sistemas de banco de dados relacionais e ao SGBD DB2 da IBM que foi adotado durante o transcorrer do projeto.

No capítulo 6 faz-se a descrição da ferramenta CASE utilizada durante o transcorrer do projeto, mostrando como ela auxiliou no desenvolvimento do esquema de banco do dados.

No capítulo 7 é apresentado o sistema que foi projetado.

O capítulo 8 é destinado para as conclusões e posteriores considerações finais.

2. CARACTERÍSTICAS DA UNIVERSIDADE FEDERAL DE PELOTAS (UFPel)

A Universidade Federal de Pelotas (UFPel) tem-se esforçado arduamente objetivando a atualização constante de sua estrutura. Para tanto, necessita-se de uma busca contínua rumo às tecnologias do novo milênio, visando a consolidar, da melhor forma, a sua função de centro de referência regional.

Dessa forma, a UFPel nesse início de milênio, compromete-se em buscar os seguintes objetivos:

- Promover a formação de indivíduos de forma que se tornem cidadãos conscientes, profissionalmente competentes e participativos, intimamente inseridos no processo de transformação - condições indispensáveis para que possam atuar como propulsores do desenvolvimento social, econômico e tecnológico do país.
- Incentivar o desenvolvimento do espírito de indagação inerente ao jovem, oportunizando, através de ferramentas tecnológicas modernas e satisfatórias, a busca sistemática e permanente do conhecimento.
- Oportunizar à comunidade em geral, acesso a processos contínuos de formação profissional, qualificação, reprofissionalização, especialização, aperfeiçoamento e atualização tecnológica.
- Reorganizar, através de estudos de identificação do perfil de competências necessárias às atividades produtivas, as bases curriculares dos cursos técnicos de 2° grau.
- Proporcionar condições e infra-estrutura adequadas para atender demandas existentes e emergentes do ensino tecnológico vinculadas ao desenvolvimento sócio-econômico regional e nacional.
- Introduzir novas metodologias que induzam o aluno de graduação a ser empreendedor e gerador de novas oportunidades de emprego.
- Rever os modelos atuais de estruturas curriculares, introduzindo, na educação dos alunos de graduação, uma sólida cultura de informática que lhes oportunizará, através da participação nas grandes redes acadêmicas, acesso e intercâmbio de informações com outras Instituições nacionais e internacionais.

- Proporcionar, face ao contexto atual de globalização e de regionalização, estudos interdisciplinares sobre a integração latino-americana, em seus aspectos educacional, econômico, social, artístico-cultural, ecológico, político, entre outros.
- Proporcionar condições e infra-estrutura adequadas para atender às demandas acadêmicas vinculadas ao desenvolvimento sócio-econômico regional e nacional e suplementar o Estado em programas de atendimento social.
- Apontar ações que conduzam ao aprimoramento do processo de avaliação na UFPel, em função de sua vinculação com o desenvolvimento tecnológico, econômico, social e cultural do País, possibilitando a elevação constante da qualidade do ensino oferecido e de sua capacidade de formar pessoal efetivamente qualificado para o mundo do trabalho
- Promover ações continuadas de avaliação do processo educacional, em função do desenvolvimento tecnológico, sócio-econômico-cultural da região e do país, possibilitando a permanente elevação da qualidade de ensino ofertada e da capacidade de formar indivíduos efetivamente qualificados para o mercado de trabalho.

3. CARACTERÍSTICAS DA BIBLIOTECA SETORIAL DE CIÊNCIA E TECNOLOGIA (BS/CT)

A Biblioteca Setorial de Ciência e Tecnologia (BS/CT) tem como função básica, suprir os cursos de ciências exatas da Universidade Federal de Pelotas. Porém, como o seu acervo bibliográfico é muito vasto e diversificado, ela também assiste, principalmente, os cursos de Nutrição, Administração Hospitalar e Hoteleira, Produção do Vestuário, Química de Alimentos e Biologia. Secundariamente, a BS/CT atende aos cursos de Agronomia, Engenharia Agrícola, Veterinária, Medicina, Enfermagem, Odontologia, Educação Física, Artes, Arquitetura e Química, além de atender quase todos os cursos de pós-graduação, tanto a nível de especialização, mestrado e doutorado. Ademais, em virtude de convênio com a Universidade Católica de Pelotas, atende a usuários desta instituição, bem como estudantes de ensino médio do Conjunto Agrotécnico Visconde da Graça, Centro Federal de Ensino Tecnológico de Pelotas e outras de Pelotas e Capão do Leão.

Para tanto, atualmente a BS/CT conta com uma área de 239,30 metros quadrados, e o seu acervo contabiliza 12.730 obras e 21.291 exemplares de livros, 677 periódicos e 1336 folhetos. A BS/CT possui cerca de dois mil usuários cadastrados, efetuando uma média de aproximadamente cento e oitenta empréstimos diários.

A BS/CT adota o sistema aberto de empréstimos de exemplares, isto é, o próprio usuário tem acesso livre ao acervo. Desta forma, o usuário é quem procura a obra desejada nas estantes e a encaminha posteriormente ao funcionário para efetivar um empréstimo.

O critério para a aquisição de novas obras nas bibliotecas setoriais da UFPel basicamente obedece a sugestão dos professores. A Administração da UFPel é responsável pela solicitação de compra das obras indicadas pelos professores, bem como a entrega das mesmas às bibliotecas setoriais.

A BS/CT atualmente conta com cinco funcionários para atender aos usuários, bem como efetuar tarefas diversas relacionadas com o processamento dos serviços da biblioteca, sendo que um deles é o chefe da BS/CT. O mesmo é o responsável pela gestão da biblioteca, e é o funcionário mais alto na hierarquia da Biblioteca Setorial de Ciência e Tecnologia (BS/CT).

4. O SISTEMA ATUAL

Como este trabalho visa definir a base de dados de um sistema que atenda as necessidades da Biblioteca Setorial de Ciência e Tecnologia da UFPel, será necessário realizar um estudo do atual sistema, para que possam ser identificados os recursos disponibilizados e as causas que levaram-o a ser sub-utilizado, bem como da política adotado pela biblioteca, para que o trabalho possa suprir as necessidades existentes.

4.1 Descrição do SAB-II

O sistema que atualmente está sendo utilizado pela Biblioteca Setorial de Ciência e Tecnologia da UFPel denomina-se Sistema de Automação de Bibliotecas II (SAB-II), foi desenvolvido através de um convênio entre a Fundação Getúlio Vargas, a IBM e a Fundação Universidade do Rio Grande (FURG) com o objetivo de permitir o cadastro e recuperação de qualquer tipo de material informacional.

Em agosto de 1990, este sistema foi implantado em todas bibliotecas setoriais da UFPel, pois uma das funções previstas para o mesmo era a integração entre as bibliotecas, mas este recurso acabou nunca sendo utilizado.

Este sistema é executado em um *mainframe* IBM modelo 4381, localizado no Centro de Informática (CI) da UFPel, apoiado sobre o sistema operacional VM / SP. Esta máquina tem como características principais possuir 16 Mb de memória RAM, 8 Gb de espaço para armazenamento de dados em disco.

Atualmente, está em fase de testes, a migração deste sistema para uma nova máquina que possibilite uma maior performance. Este máquina seria um *mainframe* IBM / 9221 que tem como características possuir 64MB de memória principal, 32GB de espaço em disco e utilizar como plataforma o sistema operacional VM / ESA.

O sistema todo é composto de vários módulos: uma base de dados relacional IBM DB2, um software de indexação de documentos textuais, chamado STAIRS, um sistema de arquivos (VSAM), as linguagens CSP e COBOL, usadas para trabalhar com o STAIRS, além de rotinas, em *Assembly*, responsáveis pela interface entre o STAIRS e a linguagem CSP (denominado módulo EXIT). O esquema a seguir ilustra o sistema como um todo.

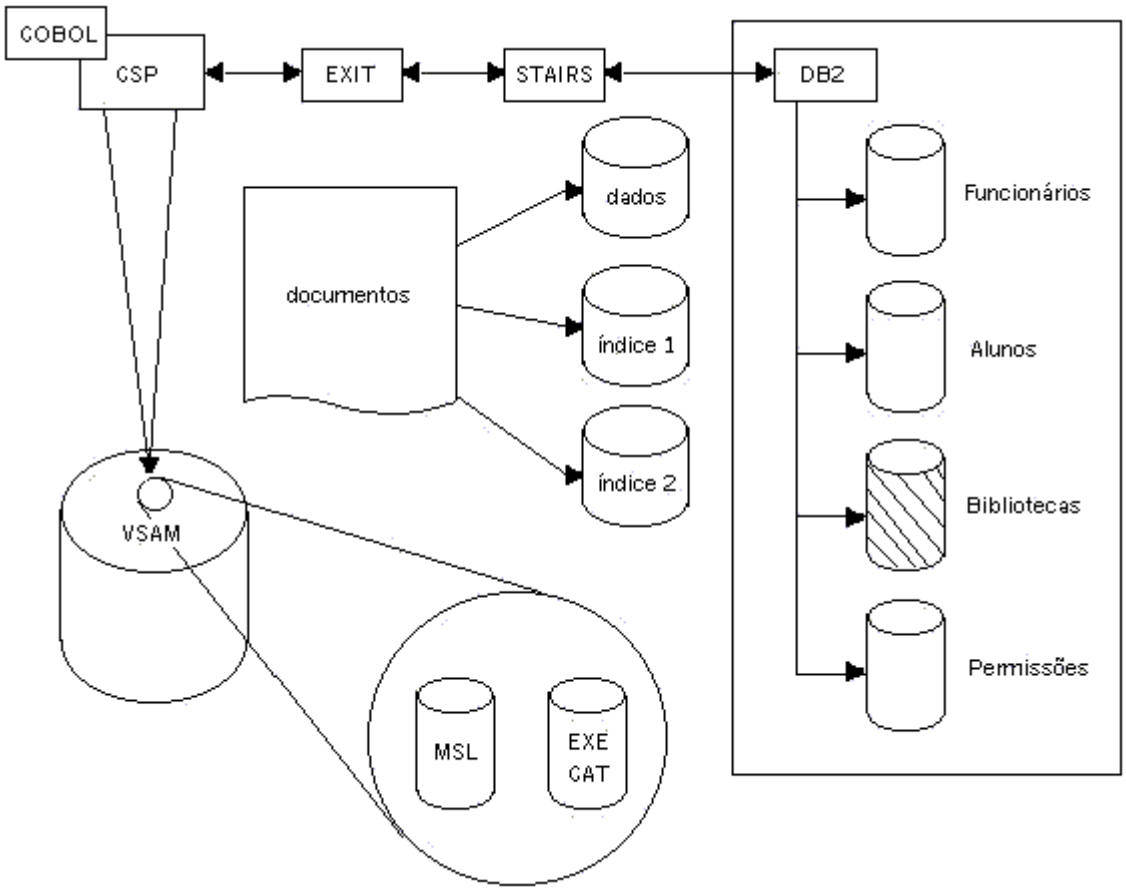

FIGURA 1 – Esquema do SAB-II

No momento em que um usuário efetua login no sistema (é estabelecida uma sessão com o SAB-II), é também criada uma máquina virtual, denominada CMS, que aloca uma área em disco e uma fatia de tempo do processador para seu próprio consumo. O acesso ao sistema é limitado por senhas, que identificam classes diferentes de usuários. Para manter a integridade em seu banco de dados o SAB-II permite que cada classe de usuário tenha acesso a um número limitado de funções (módulos), esta restrição impede que o usuário possa acessar ou alterar dados que não são pertinentes à sua classe. Os módulos e as classes de usuários serão descritas nas seções 4.2 e 4.3.

4.2 Módulos do SAB-II

O sistema SAB-II possui todas as funções necessárias para o gerenciamento de uma biblioteca, de maneira rápida e eficaz. Dentre as funções existentes no SAB-II destacam-se, cadastro de usuários, catalogação (cadastro) do acervo, pesquisa bibliográfica, circulação (transações) e emissão de relatórios de inventário, etiquetas de lombada, etiquetas de registro, relatório de devedores e emissão de carta-cobrança. Mas, atualmente estão disponíveis somente o cadastro de exemplares e a consulta ao acervo.

A seguir serão listados todos os módulos existentes no SAB-II:

- Cadastro de exemplares – módulo responsável pelo cadastro de exemplares do acervo;
- Cancelamento de reservas – módulo que efetua a cancelamento de reserva por obra;
- Consulta situação de uma obra – módulo que retorna a situação de uma obra do acervo;
- Devolução de exemplares – módulo que efetiva o encerramento de empréstimos através da devolução do exemplar;
- Emissão de relatórios – módulo que possibilita a emissão de relatórios que mantém um controle sobre as transações da biblioteca;
- Empréstimo de exemplares – módulo responsável pela efetivação de empréstimos para usuários;
- Efetivação de reservas – módulo que habilita o usuário a efetivar reserva de uma obra;
- Manipulação de coleções – módulo responsável por manter um cadastro de códigos que identifiquem para o usuário que está realizando uma consulta, se um exemplar é de consulta local, disponível para empréstimo, ou de empréstimo proibido, se este está sendo restaurado ou foi constatada a sua baixa;
- Manipulação de locais das bibliotecas – módulo que mantém um cadastro de todas as bibliotecas setoriais;
- Manipulação de obras – módulo responsável em manter um cadastro de todas as obras do acervo;
- Manipulação de parâmetros do sistema – módulo que permiti que sejam alterados parâmetros gerais de uso do sistema;
- Manipulação de recursos – módulo responsável por manter um cadastro das máquinas CMS que podem ser utilizadas pelos funcionários da biblioteca;
- Manipulação de usuários – módulo responsável por manter um cadastro de usuários do sistema;

- Manipulação de Pessoal de Processamento Técnico – módulo responsável por manter um cadastro de pessoal de processamento técnico (usuário nível “C”), com autoridade para alterar e eliminar documentos das bibliotecas para as quais os mesmos são cadastrados;
- Manipulação do pessoal – módulo responsável por manter um cadastro dos funcionários da biblioteca;
- Manipulação dos vínculos – módulo responsável por manter um cadastro de vínculos dos usuários do sistema com a instituição;
- Pesquisa bibliográfica – módulo que permite ao usuário pesquisar o acervo da biblioteca;
- Renovação de empréstimo – módulo que permite ao usuário renovar seu empréstimo, desde que não exista reserva pela obra.

A biblioteca possui dois terminais SNA que acessam o SAB-II, utilizados somente para consulta à base de dados. Tanto o cadastro de livros como o de usuários da biblioteca é feito em formulário de papel, bem como o controle de empréstimos e reservas. Os dados para o cadastro e atualização das publicações são repassados para um funcionário da universidade que atualizará a base de dados. O atendimento é feito por ordem de chegada, sem utilizar nenhum mecanismo automático para manter um controle sobre isto.

4.3 Política do SAB-II

Após serem realizadas entrevistas com a diretora da divisão de bibliotecas da UFPel, ficou definido que a política a ser adotada durante a definição do banco de dados para o software especificado deveria ser a mesma do SAB-II. Neste sistema cada usuário é enquadrado dentro de uma classe, sendo que cada um destas classes pode acessar um número restrito de módulos (descritos na seção 4.2.).

Primeiramente são identificadas duas grandes classes de usuários:

1. Funcionários da biblioteca;
2. Usuários da biblioteca.

Os usuários da classe 2 são todos aqueles que não trabalham na biblioteca (alunos, professores e servidores). Os usuários que se encaixam-se nesta classe, têm acesso as seguintes funções do sistema:

1. Pesquisa bibliográfica;

2. Efetivação de reservas;

3. Cancelamento de reservas;

4. Consulta situação de uma obra.

Todos os usuários da classe 1, são divididos em quatro sub-classes de funcionário, representadas pelas letras A, B, C, D. Esta subdivisão indica que cada funcionário tenha acesso a somente as rotinas relativas à sua função.

Estas quatros sub-classes, representados pelas letras A, B, C e D, possui hierarquia entre si, sendo que a classe "A" está no topo da hierarquia, enquanto que a classe "D" está na base da mesma. Estas quatro classes serão descritas a seguir:

- Classe "D" – são os funcionários que exercem as funções administrativas do balcão de empréstimo. O funcionário classe "D" tem direito de acesso as seguintes funções :
    1. Empréstimo de exemplares;
    2. Renovação de empréstimo;
    3. Devolução de exemplares;
    4. Reserva de obras;
    5. Cancelamento de reservas;
    6. Consulta Situação de uma obra.
- Classe "C" – são os funcionários que exercem funções inerentes a manutenção do sistema. Os funcionário desta classe possuem todos os direitos dos funcionários do nível "D" e, além disso, possuem acesso a módulos que relacionam-se com as tabelas de usuários, obras e exemplares. Os módulos acessados por um funcionário classe "C" são:
    1. Manipulação de usuários;
    2. Manipulação de obras;
    3. Manipulação de exemplares.
- Classe "B" – são os funcionários que exercem as funções de chefes das bibliotecas setoriais. Os funcionários da classe "B", possuem acesso a todos os módulos referentes às classes "C" e "D" e, além disso, possuem acesso as funções que relacionam-se com as tabelas de locais de bibliotecas, recursos computacionais, parâmetros do sistema, vínculos com a instituição e pessoal. Estes são os módulos acessados por funcionários classe "B":
    1. Manipulação de parâmetros do sistema;
    2. Manipulação de locais das bibliotecas;
    3. Manipulação de recursos;
    4. Manipulação do pessoal;
    5. Manipulação dos vínculos;

6. Manipulação de coleções;
7. Manipulação de Pessoal de Processamento técnico;
8. Emissão de relatórios.

- Classe “A” – são os funcionários que exercem a função de coordenador geral do sistema. No caso da UFPel, esse funcionário é o diretor da divisão de bibliotecas. O funcionário dessa classe, além de terem os mesmos direitos de um funcionário classe “B, são os únicos que podem cadastrar os funcionários classe “B”. A diferença entre os funcionários das classes “A” e “B” é que os funcionários classe “B” podem atuar somente sobre os dados de sua biblioteca setorial, enquanto que os funcionários classe “A” tem acesso aos dados de todas as bibliotecas setoriais.

4.4 Problemas identificados no SAB-II

Durante visitas a Biblioteca Setorial de Ciência e Tecnologia e ao Centro de Informática foi feito um levantamento sobre os atuais problemas do SAB-II, que será descrito a seguir.

Embora o sistema possua todos os módulos necessários para o gerenciamento de uma biblioteca, atualmente somente o módulo de consulta encontra-se disponível para os usuários. Os processos de cadastro de obra e de cadastro de usuário são feitos através do preenchimento de um formulário de papel. Desta maneira arcaica também são executados o controle de empréstimos e de reservas. Estas funções são extremamente indispensáveis em um sistema de biblioteca, mas, no nosso caso, os dados são registrados em fichas, de forma análoga a um sistema de arquivos de fichas de papel, o que torna uma tarefa quase impossível a recuperação dos dados para geração de relatórios

Os funcionários das bibliotecas ficam encarregados de repassarem os formulários referentes as atualizações no cadastro de obras de sua biblioteca setorial para a divisão de bibliotecas da UFPel. Estes dados serão então digitados por um funcionário autorizado, atualizando o banco de dados. Como este procedimento não é realizado regularmente, quando é realizado, os dados referentes ao acervo da UFPel não condizem com a realidade.

O módulo de consulta apresenta como característica principal o indexador STAIRS, que possibilita uma vasta gama de pesquisas. Apesar disso, devido ao sistema possuir uma interface nada amigável, a realização de consultas e o posterior refinamento das mesmas, exigem que o usuário possua um bom conhecimento na área de informática, para uso do módulo de maneira adequada. Além disso, devido a problemas na rede interna (Intranet) da UFPel, o *turn-arround* (tempo de resposta) de uma consulta é excessivamente longo.

O problema central do sistema está no fato de que a base de dados está em um formato proprietário – antigo e complicado – que não oferece a possibilidade de uso através dos aplicativos mais comuns no mercado, como no formato DB2. Isso gera problemas como a dificuldade de acesso a base de dados, complicações na inserção e remoção e, principalmente na manutenção do sistema e na descoberta de falhas.

O SAB-II foi escrito em linguagens praticamente obsoletas, CSP, COBOL, sendo que algumas rotinas forma escritas em linguagem *Assembly*, o que é o primeiro empecilho para a manutenção, visto que a UFPel não possui pessoal com conhecimento adequado para executá-la. Além disso o sistema carece de uma documentação forte e consistente, o que torna qualquer tentativa de manutenção inviável. Estes são os principais fatores que o levaram a ser sub-utilizado, já que nunca ocorreu a manutenção adequada.

Visando solucionar estes problemas, será definido um esquema de banco de dados relacional, com base nos tipos dados padrão do SGBD DB2 da IBM, já disponível na Universidade. Este novo modelo será desenvolvido com auxílio de uma ferramenta CASE, o que tornará o projeto muito mais confiável, pois se trata de uma nova metodologia de desenvolvimento de software, que facilita o trabalho dos analistas e programadores. Além disso, a criação da base de dados e a posterior manipulação da mesma, será feita através da linguagem SQL, o que facilitará a implementação do sistema, pois esta linguagem tem a capacidade de ser utilizada em linguagens hospedeiras, como COBOL, FORTRAN, C, DELPHI, etc.

5. ADMINISTRAÇÃO DE DADOS

Os dados constituem um recurso valioso dentro de uma organização e sua adequada administração contribui para que as informações possam ser mais facilmente localizadas, compreendidas e mais acuradamente utilizadas e armazenadas.

Uma das primeiras formas de armazenamento dos dados foi os sistemas de arquivos permanentes. Para permitir ao usuário a recuperação destas informações o sistema deveria apresentar um conjunto de programas de aplicações que tratavam esses arquivos. De acordo com a necessidade que se apresentava, novos programas foram incorporados a esses sistemas, com isso, novos arquivos permanentes eram criados contendo dados para atender a tais necessidades. Mas, conforme o passar do tempo, mais arquivos e programas de aplicações eram adicionados ao sistema. Sendo que, a cada nova incorporação, o volume de dados armazenados aumentava excessivamente.

A tecnologia de banco de dados surgiu no momento em que os especialistas no desenvolvimento de sistemas computacionais perceberam que, para a informatização de grandes organizações, várias questões relacionadas com o gerenciamento de dados necessitavam ser resolvidas de uma forma mais eficiente.

Um *sistema de gerenciamento banco de dados* (SGBD) consiste em uma coleção de dados inter-relacionados e em um conjunto de programas para acessá-los. Um conjunto de dados, normalmente referenciado como *banco de dados*, contém informações sobre uma empresa em particular. O principal objetivo de um SGBD é prover um ambiente que seja *adequado* e *eficiente* para recuperar e armazenar informações de um banco de dados [KOR99].

O gerenciamento destes dados envolve a definição de estruturas (arquivos) para armazenamento das informações e o fornecimento de mecanismos (programas) para manipulá-los. Estes mecanismos permitem ao usuário de um sistema de banco de dados executar uma variedade de operações sobre tais arquivos, incluindo entre outras, as seguintes:

- Acrescentar novos arquivos, vazios, ao banco de dados;
- Inserir novos dados em arquivos existentes;
- Recuperar dados de arquivos existentes;
- Atualizar dados em arquivos existentes;

- Remover arquivos existentes, vazios ou não, do banco de dados.

Além disso, a utilização de um sistema de banco de dados para o gerenciamento de dados implica em uma série de vantagens:

- **Redução da Redundância:** Num sistema gerenciamento da informação sem a utilização de banco de dados é possível encontrarmos aplicações que se utilizem cada uma de seus próprios arquivos, podendo haver dados que apareçam em mais de um arquivo, desperdiçando espaço de armazenamento. Com a utilização de um sistema de banco de dados, estes dados podem ser integrados e a redundância desnecessária pode então ser eliminada;
- **Evitar Inconsistências:** Como conseqüência da redundância de dados, é possível acontecer que uma cópia dos dados seja atualizada e a outra não, ou mesmo que sejam atualizadas com valores diferentes, o que causará o fornecimento de informações incorretas ao usuário. Se o dado apresentar-se como uma única entrada, tal problema não mais ocorrerá;
- **Compartilhamento de Dados:** As aplicações existentes poderão compartilhar os dados do banco de dados e as novas aplicações poderão ser desenvolvidas para operar sobre os mesmos dados armazenados. Isto significa que as necessidades de dados das novas aplicações podem ser satisfeita sem a criação de quaisquer dados adicionais armazenados;
- **Reforço dos Padrões:** Uma maneira de assegurar que todos os padrões aplicáveis serão observados na representação dos dados. Esta padronização é especialmente interessante para facilitar o intercâmbio de dados ou a migração entre sistemas;
- **Aplicação de Restrições de Segurança:** Este é um aspecto muito importante, já que os dados estarão centralizados e não dispersos. É possível definir canais de acesso e controles de segurança para cada tipo de acesso (consulta, atualização, etc.);
- **Manutenção da Integridade:** Através do controle centralizado do banco de dados, é possível definir controles de integridade a serem usados sempre que for empreendida qualquer operação de atualização, o que num ambiente centralizado e compartilhado é de extrema importância para que uma atualização incorreta do banco de dados não afete outros usuários.

A figura 2 [DAT99], demonstra os quatro componentes principais de um sistema de banco de dados: dados, hardware, software e usuários.

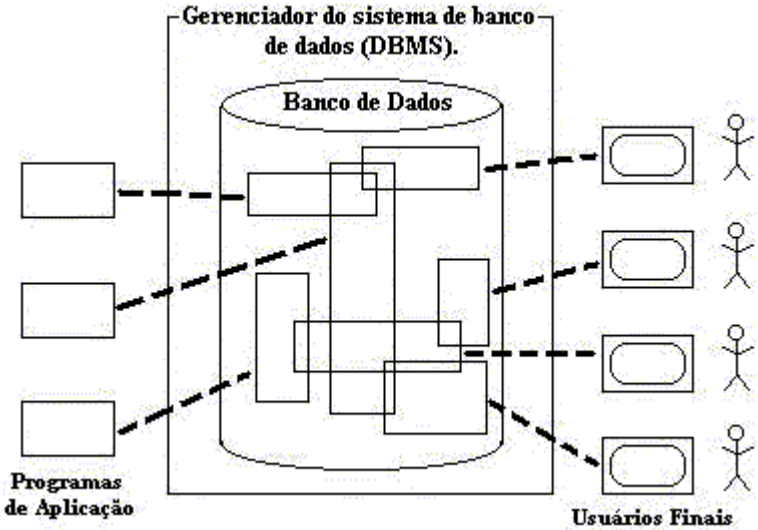

FIGURA 2 – Principais componentes de um sistema de banco de dados

Como já foi dito, o objetivo global de sistema de banco de dados é guardar a informação e torná-la disponível sob demanda. A informação em causa pode ser qualquer coisa que se considere ter relevância para o indivíduo ou organização a que o sistema deve servir. Este trabalho buscará identificar as informações que realmente são relevantes para o gerenciamento eficaz da Biblioteca Setorial de Ciência e Tecnologia, para que possa ser definido um esquema de banco de dados mais adequado a suas necessidades.

É chamado de esquema do banco de dados a representação de um banco de dados em um modelos de dados. Temos o esquema conceitual do banco de dados, o seu esquema lógico e físico e podemos vários sub-esquemas, representando as visões dos usuários. Os modelos de dados mais difundidos são o modelo entidade-relacionamento (nível conceitual) e o modelo relacional (nível lógico). Estes dois modelos foram adotados durante a especificação do esquema do banco de dados para o sistema da Biblioteca Setorial de Ciência e Tecnologia da UFPel.

5.1 Abstração de Dados

A *Abstração de dados* ocorre quando um sistema de banco de dados tem meios de proporcionar ao seu usuário uma visão abstrata dos dados que contém. Ou seja, o sistema esconde do usuário alguns detalhes sobre como os dados são armazenados, recuperados ou mantidos.

O SGBD tem uma preocupação com eficiência, que é conseguida através do uso de estruturas complexas de armazenamento de informação. Mas esta complexidade não é visível aos usuário dos bancos de dados. Os sistemas de gerenciamento de bancos de dados oferecem aos usuários vários níveis de abstração.

Neste projeto foram definidos esquemas de banco de dados nos três níveis de abstração, que serão apresentados a seguir.

5.1.1 Nível Físico

Pode-se definir o nível físico como sendo o nível mais baixo de abstração, no qual se descreve como os dados são armazenados. O nível físico depende fortemente do SGBD escolhido, pois as estruturas de armazenamento de informação que o SGBD usa são descritas neste nível.

A nível físico foram escritos script's em linguagem SQL para criação e posterior manipulação da base de dados definida, já de acordo com SGBD adotado, no caso DB2, apresentados na seção 7.3.

5.1.2 Nível Lógico

Neste nível, se descrevem os dados que serão armazenados e os relacionamentos existentes entre eles. Não existe a preocupação de como os dados serão armazenados.

O esquema de banco de dados, em nível lógico, foi definido com base no *modelo relacional*, ele será demonstrado na seção 7.2.

5.1.3 Nível Conceitual e Nível de Visão

O *nível conceitual* oferece uma visão mais comportamental do banco de dados, mais próxima do mundo real. Em outras palavras, a preocupação maior é visualizar as estruturas de dados como representações do mundo real. Neste nível foi definido um esquema de banco de com base no modelo *entidade-relacionamento*, que será descrito na seção 7.1.

O *nível de visão* pretende simplificar o acesso ao banco de dados por usuários que têm necessidade de apenas parte do banco de dados. É possível definir várias visões do banco de dados de acordo com as necessidades dos usuários.

Na figura 3, temos um esquema que mostra os níveis de abstração existentes no projeto de bancos de dados.

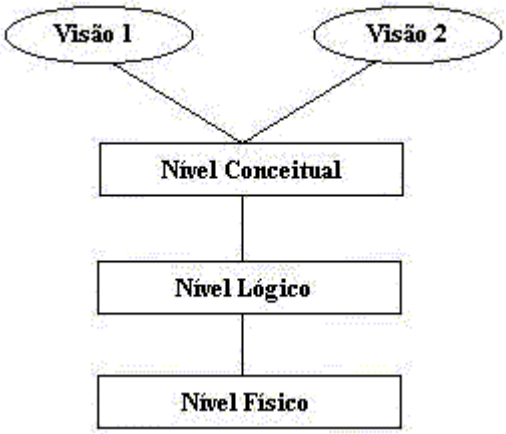

FIGURA 3 – Níveis de abstração em um projeto de banco de dados

5.2 Sistemas Relacionais

A maioria dos sistemas de banco de dados desenvolvidos nos últimos anos são *relacionais* (ao contrário dos sistemas mais antigos), quase todos os produtos banco de dados desenvolvidos a partir da década de 70 (como o DB2 da IBM) se basearam na *abordagem relacional***,** além disso, quase toda pesquisa realizada sobre banco de dados nos últimos 25 anos também baseou-se nessa abordagem. É inegável que a abordagem relacional tornou-se a tendência de mercado nos dias atuais, e que *modelo relacional* é o desenvolvimento isolado mais importante em toda história da área de banco de dados, pois consolidou-se como primeiro modelo de dados para aplicações comerciais.

Sendo assim, como a maioria das pesquisas atuais em banco de dados baseiam-se em conceitos relacionais, o trabalho proposto também seguirá essa tendência, buscando modelar uma base de dados que venha suprir as necessidades da Biblioteca Setorial de Ciência e Tecnologia da Universidade Federal de Pelotas.

Date [DAT99] definiu, resumidamente, um sistema relacional como sendo aquele, no qual:

1. Os dados são percebidos pelos usuários como tabelas (e nada mais que tabelas);
2. Os operadores a disposição do usuário (por exemplo, para a recuperação de dados) são operadores que geram novas tabelas a partir das antigas. Por exemplo, haverá um operador para extrair um subconjunto das linhas de uma determinada tabela e outro para extrair um subconjunto de colunas de uma tabela e, naturalmente um

subconjunto de linhas ou colunas de uma tabela podem, ambos ser considerados como tabelas.

O usuário de um sistema não-relacional, em contraste, vê outras estruturas de dados, ao invés de ou em adição às tabelas do sistema relacional. Por exemplo, em um sistema *hierárquico,* os dados são representados para o usuário sob a forma de um conjunto de três estruturas (hierarquias) e os operadores fornecidos para manipular tais estruturas incluem operadores capazes de percorrer caminhos hierárquicos subindo e descendo de árvores.

Os sistemas de banco de dados podem ser categorizados de acordo com as estruturas de dados e operadores que apresentam ao usuário. Os sistemas mais antigos (pré-relacionais) são divididos em três grandes categorias, sistemas *de lista invertida, hierárquicos e de rede.* Mais recentemente, a pesquisa na área de banco de dados prosseguiu dando origem ao que pode ser chamado de *sistemas pós-relacionais,* alguns baseados em extensões da abordagem relacional, outros sendo tentativas de fazer algo totalmente diferente. Os primeiros produtos resultantes desta pesquisa pós-relacional que apareceram no mercado foram os sistemas *orientados-a-objeto* (OO).

Nas seções subsequentes será feito uma analise do SGBD relacional DB2 da IBM, que foi adotado durante o transcorrer do projeto.

5.2.1 Componentes do DB2

As estruturas internas de DB2 são bastante complexas, como seria de esperar de um sistema que fornece todas as funções tipicamente encontradas num SGBD moderno. Assim, o produto contém um número muito grande de componentes internos. De um ponto de vista de alto nível, porém, pode ser considerado como tendo só quatro componentes *principais*, cada um dos quais se divide em numerosos sub-componentes. Os quatro principais são:

1. Componente de **serviços de sistema**, que suporta a operação do sistema, comunicação de operador, registro cronológico e funções similares.
2. Componente de **serviços de bloqueio**, que fornece o necessário controle para gerenciamento de acesso concorrente a dados.
3. Componente de **serviços de banco de dados**, que suporta a definição, recuperação e atualização de dados, de usuários e sistema.
4. Componente de **recurso de distribuição de dados**, que fornece suporte ao banco de dados distribuído do DB2.

Destes quatro componentes, o mais diretamente relevante para o usuário é o de *serviços de banco de dados.* Esse componente por sua vez, se divide em cinco sub-

componentes principais, que chamaremos de *Precompilador, Bind (ligação), o Supervisor de Tempo de Execução, o Gerente de Dados Armazenados e o Gerente Buffer* (Intermediário). Juntos, esses componentes suportam a preparação de programas de aplicação para execução e a subsequente execução dos mesmos.

- **Precompilador** é um preprocessador para programas de aplicação, que contém declarações SQL embutidas. Reúne tais declarações num **Módulo de Solicitação de Banco de Dados** (DBRM), substituindo-as no programa original por CALLs (chamadas) na linguagem principal ao supervisor de tempo de execução.
- O componente **Bind** compila um ou mais DBRMs relacionados, para produzir um **plano de aplicação** (isto é, um código executável para implementar as declarações SQL contidas nesses DBRMs).
- **Supervisor de Tempo de Execução:** supervisiona os programas SQL durante sua execução. Quando um tal programa solicita alguma operação do banco de dados, o controle vai primeiro ao Supervisor de Tempo de Execução, graças ao CALL inserido pelo Precompilador. O Supervisor de Tempo de Execução então passa o controle ao plano de aplicação, e este por sua vez invoca o Gerente de Dados Armazenados para que execute a função solicitada.
- **Gerente de Dados Armazenados** gerencia o banco de dados armazenado, recuperando e atualizando registros conforme solicitação dos planos de aplicação. Em outras palavras o Gerente de Dados Armazenados é o componente responsável por invocar outros componentes, se necessário, para executar funções subsidiárias tais como bloqueio, registro cronológico, classificação, etc., durante a realização de sua tarefa básica; em particular, invoca o Gerente Intermediário (ver abaixo) para executar operações I/O físicas.
- **Gerente Intermediário** (Buffer) é o componente responsável pela transferência física de páginas de dados entre disco e armazenagem principal; isto é, executa as operações I/O efetivas, como foi dito acima.

5.2.2 Fluxo de Controle no DB2

Vamos examinar as idéias da seção anterior um pouco mais profundamente. Analise a figura 4, ela mostra os passos principais, envolvidos na preparação e execução de uma aplicação DB2. Para fixar as idéias, suponhamos que o programa original *P* esteja escrito em COBOL (tomamos COBOL para definição; o processo geral, é claro, é essencialmente o mesmo para outras linguagens). Os passos que o programa *P* deve percorrer são:

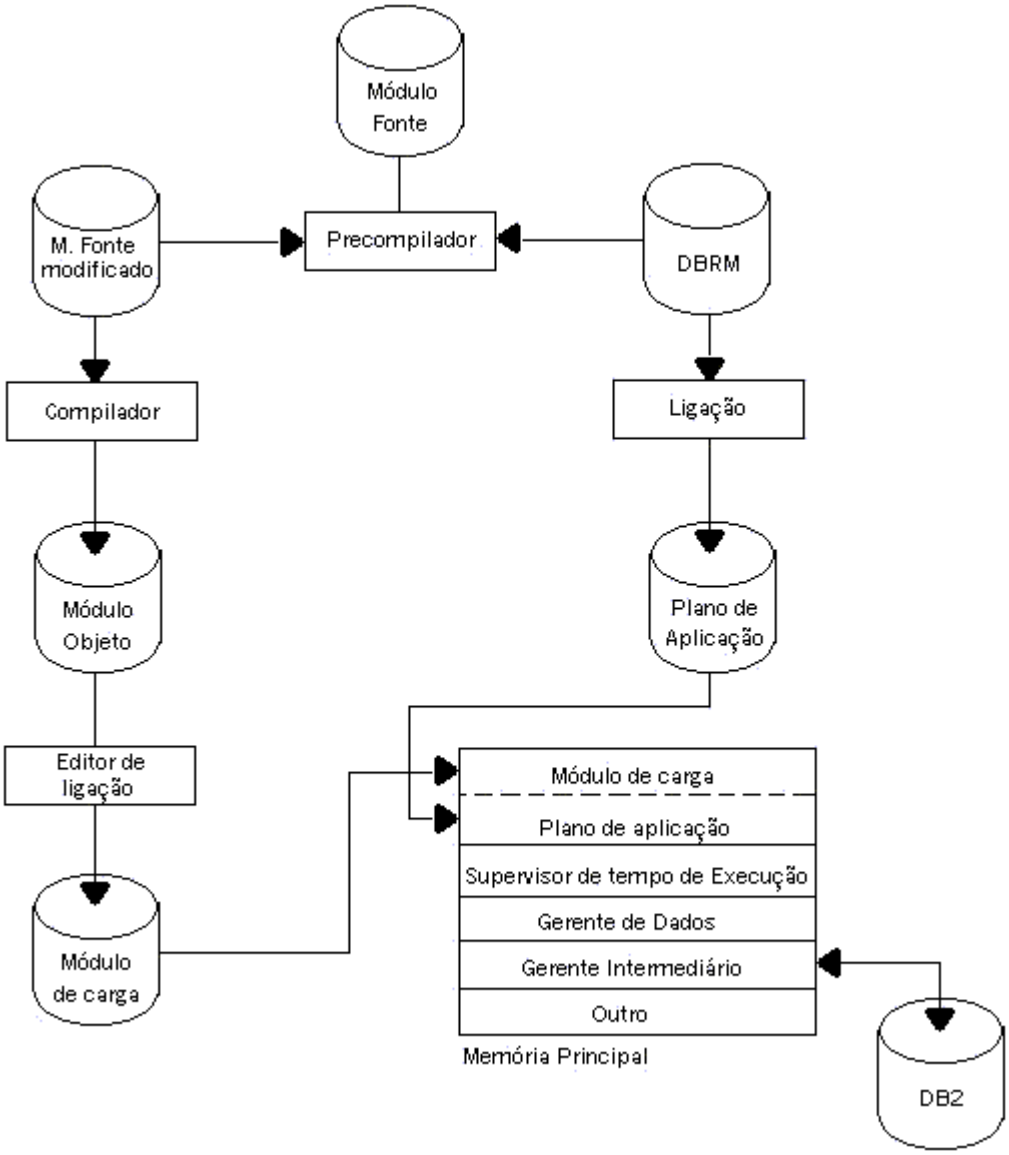

FIGURA 4 – Fluxo de controle no DB2

1. Antes de *P* poder ser compilado, pelo compilador regular COBOL, tem que ser processado pelo Precompilador DB2. Como se disse, o Precompilador é um preprocessador para linguagens de programação de aplicação DB2 (C, COBOL, DELPHI, etc.). Sua função é analisar um programa de fonte escrito em uma dessas linguagens, retirando as declarações SQL que encontrar e substituindo-as por declarações CALL da linguagem principal. Ao tempo de execução esses CALLs passarão o controle ao Supervisor de Tempo de Execução. Com as declarações SQL que encontrar, o Precompilador constrói um Módulo de Solicitação de Banco de Dados (DBRM), que, subsequente, se torna input ao componente Bind. Nota: O DBRM pode ser considerado simplesmente como uma representação interna das declarações SQL originais. Não consiste em código executável.

2. Em seguida, o programa COBOL modificado é compilado e sofre edição por ligação, do modo normal. Devemos nos referir ao output deste passo como “módulo de carga P”.
3. Agora, chegamos ao passo Bind. Como já se sugeriu, Bind na verdade é um **compilador SQL**: Converte solicitações ao banco de dados (isto é, declarações SQL) para código executável. (Ainda mais, é um compilador *que otimiza;* o output de Bind não é simplesmente só código, é código *otimizado*. Em outras palavras, em DB2 o componente de importância maior é um sub-componente de Bind. O input para Bind é um DBRM (ou, mais provavelmente, um conjunto de vários DBRMs, se o programa original envolver vários procedimentos compilados separadamente). O output de Bind – isto é, o código compilado, que, como já se disse, é chamado um plano de aplicação – é armazenado no catálogo DB2, onde pode ser achado, quando necessário, pelo Supervisor de Tempo de Execução.
4. Chegamos afinal ao tempo de execução. Como o programa original foi quebrado em duas partes (módulo de carga e plano de aplicação), esses dois pedaços devem ser, de algum modo, reunidos ao tempo de execução. É assim que isto funciona (olhe a figura 4). Primeiro, o módulo de carga *P* é carregado na memória principal; começa a executar do modo usual. Mais cedo ou mais tarde, chega ao primeiro dos CALLs inseridos pelo Precompilador. O controle passa ao Supervisor de Tempo de Execução. Este então recupera o plano de aplicação do catálogo, carrega-o na memória principal, e passa o controle a ele. O plano de aplicação, por sua vez, invoca o *Gerente de Dados Armazenados,* que executa as operações necessárias sobre os reais dados armazenados (invocando o *Gerente Intermediário*, quando for preciso) e passa os resultados de volta à aplicação executora.

A análise realizada com auxílio de [DAT99], mostra que gerenciamento dos dados no SGBD DB2 independe da linguagem de programação adotada, pois para o DB2 são relevantes apenas as instruções em linguagem SQL. O que facilitará a implementação do sistema em qualquer uma das linguagens de programação disponíveis no mercado.

6. FERRAMENTAS DE DESENVOLVIMENTO

O termo *CASE* (Computer-Aided Software Engineering), que significa Engenharia de Software Auxiliada por Computador, é utilizado para nomear as novas tecnologias de desenvolvimento de software, que facilitam os trabalho dos analistas e programadores.

Atualmente, um produto é classificado como uma ferramenta CASE quando este oferece documentação, automação e racionalização do projeto de software e de sua implementação. Dentre as classes de ferramentas CASE, podemos destacar os geradores automáticos de aplicações e as ferramentas de modelagem, projeto e implementação de bancos de dados.

Existem diversas vantagens na utilização de ferramentas CASE para o desenvolvimento de sistemas. Pode-se destacar:

- **Maior qualidade dos produtos finais:** as ferramentas CASE diminuem a probabilidade de erros, uma vez que podem ajudar no controle de consistência dos dados durante o desenvolvimento do sistema; também proporcionam maior eficácia dos produtos, ao auxiliarem as fases de Análise e Teste do produto pelo usuário;
- **Produtividade:** ao ajudar na realização de tarefas e até mesmo ao realizar algumas automaticamente, as ferramentas contribuem para uma maior agilidade no desenvolvimento de software, isto é, mais produtos em menos tempo;
- **Eliminação de trabalho monótono:** as ferramentas CASE podem realizar algumas tarefas cansativas para os desenvolvedores, tais como procurar informações e desenhar símbolos de um diagrama, as quais são mais suscetíveis ao erro;
- **Mais tempo para a tomada de decisão:** em conseqüência de as ferramentas realizarem certas atividades pelas pessoas, estas ficam liberadas para outras tarefas, geralmente mais nobres, que exigem tomada de decisão e criatividade, ao invés de tarefas repetitivas;
- **Flexibilidade para mudanças:** as ferramentas permitem que sejam mudados dados e diagramas de maneira mais rápida e fácil, o que ajuda o desenvolvedor no trabalho de tentar satisfazer o usuário;

- **Menos programação:** as ferramentas eliminam muito do trabalho de programação, deixando mais tempo para que a equipe técnica se preocupe com a Análise do Sistema, que é onde se define como solucionar o problema do usuário;
- **Melhor documentação:** por armazenarem dados e diagramas, as ferramentas também contribuem para uma melhor documentação do sistema, agilizando relatórios, busca de informações e alterações;
- **Manutenção mais fácil e ágil:** por conseqüência do item anterior, é possível ter mais informações sobre o software na hora de realizar sua manutenção (correção, atualização ou expansão).

Na modelagem do sistema foi utilizada a ferramenta Dr.CASE 3.0 que é uma ferramenta que auxilia no projeto de banco de dados, desenvolvida por Doctor Sys – Engenharia de Software – em parceria com Squadra – Tecnologia em Software. O Dr.CASE auxilia no projeto conceitual, no projeto lógico e na implementação física do Banco de Dados, além disso o Dr.CASE também dá suporte à Engenharia Reversa, recurso que permite que um banco de dados já implementado possa ter suas definições trazidas para o dicionário de dados do Dr.CASE. A partir destas definições, o Dr.CASE poderá construir diagramas E-R que descrevam o sistema correspondente.

6.1 Projeto Conceitual no Dr.CASE

Para analistas e programadores o projeto de esquema de banco de dados em nível conceitual é essencial, pois permite aos mesmos uma visão do sistema num todo, sem preocupar com detalhes. Para projeto em nível conceitual, o Dr.CASE utiliza-se do modelo entidade-relacionamento (E-R). A simbologia adotada é a sugerida por Peter Chen, com extensões.

Nesta etapa do projeto são definidos as entidades pertinentes ao sistema, e os relacionamentos entre elas, através de diagramas entidade-relacionamento (DER). No DR.CASE estes diagramas podem ser desenhados através do uso de ferramentas gráficas disponibilizadas pela ferramenta.

Os atributos e outras informações acerca das entidades podem ser informadas no mesmo ambiente de desenho do DER através de telas de diálogo.

O esquema de dados a nível conceitual, mais precisamente o diagrama E-R, do sistema especificado, foi desenvolvido com o auxílio das ferramentas gráficas disponibilizados pelo Dr.CASE.

6.2 Projeto Lógico no Dr.CASE

Apesar de alguns profissionais começarem o projeto de banco de dados por aqui, essa prática deve ser evitada a todo custo, exceto em casos onde o banco de dados é muito pequeno. O projeto lógico deve ser uma etapa de refinamento e detalhamento do projeto a partir no modelo conceitual.

Nesta etapa do projeto é realmente modelado o esquema do banco de dados. O foco deve ser concentrado sobre detalhes de como as informações serão armazenadas e, além disso, deverão ser definidas estruturas capazes de armazená-las. A trabalho a ser realizado poderá ser bem menor, se anteriormente for desenvolvido um bom projeto a nível conceitual.

Para projeto em nível lógico, o Dr.CASE trabalha com o modelo de dados relacional (baseado em tabelas).

O projeto lógico pode ser obtido a partir do conceitual, através da operação de transposição, que consiste em um conjunto de regras para mapeamento de projeto lógico partindo do conceitual. O Dr.CASE também perrmite que projeto lógico seja obtido através da leitura das estruturas de um banco de dados existente (recurso de Engenharia Reversa). Pode-se ainda optar por projetar o banco de dados diretamente em nível lógico.

O esquema de banco de dados em nível lógico, isto é, as tabelas do sistema modelado, foi obtido através das regras de transposição aplicadas pelo Dr.CASE no diagrama E-R definido a nível conceitual, como mostra a figura 5.

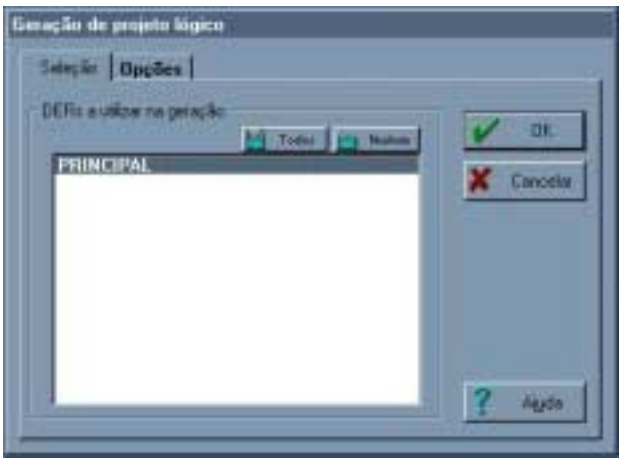

FIGURA 5 – Geração de Projeto Lógico a partir do Projeto Conceitual

É sempre possível cadastrar ou alterar tabelas, colunas, índices e chaves. Além disso, o Dr.CASE mantêm um controle de alterações das tabelas, o que evita que algumas alterações no projeto lógico sejam perdidas por uma nova operação de transposição de DER para tabelas, por exemplo.

A figura 6 mostra a tabela de empréstimos sendo editada no Dr.Case.

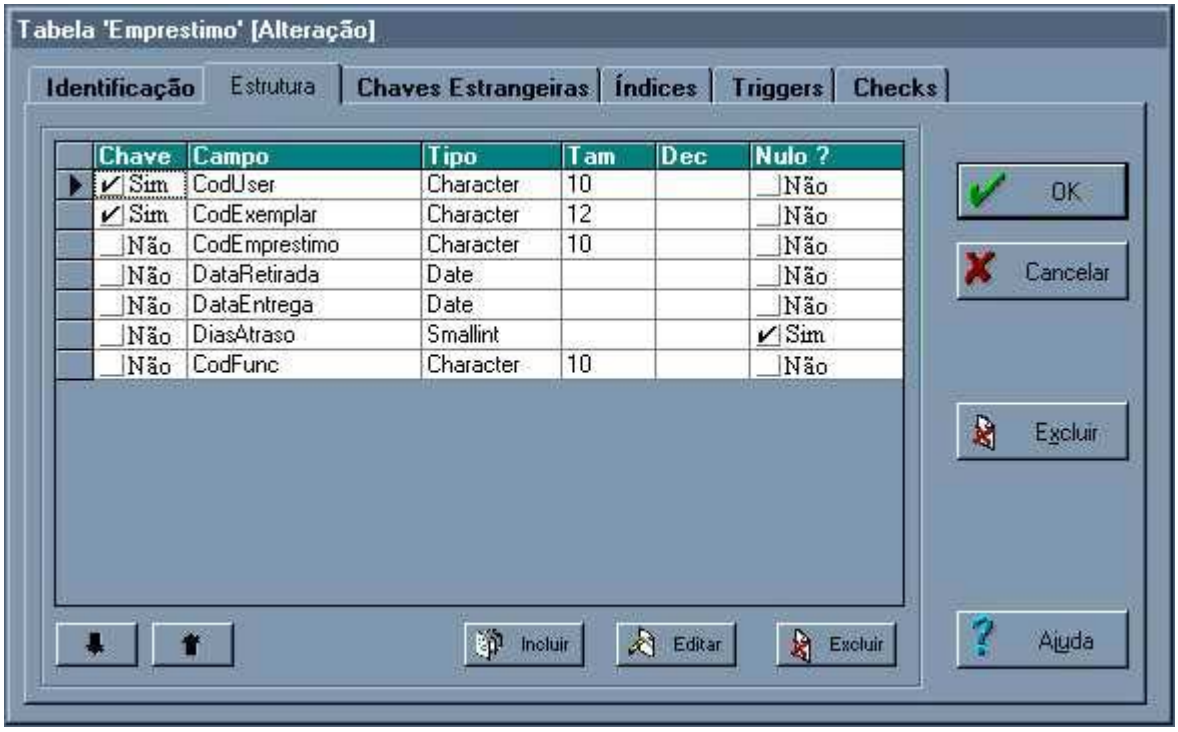

FIGURA 6 – Editando uma Tabela no Dr.CASE

6.3 Projeto Físico no Dr.CASE

Uma das principais características do Dr.CASE é capacidade de automatizar a implementação física do banco de dados. A partir do projeto lógico, pode-se gerar arquivos, tabelas, índices, visões, consultas e todas as restrições de acordo com o SGBD escolhido.

Na figura 7, observa-se todos os tipos de plataformas disponíveis para geração física do banco de dados. Após ser escolhida a plataforma adequada, deve-se definir quais módulos do projeto serão implementados. Para efetivação da geração física é necessário que não exista qualquer tipo de dependência entre as tabelas escolhidas e as não escolhidas. Sendo assim, é possível implementar parte do banco de dado antes mesmo do projeto ser concluído.

A figura 8 demonstra como pode ser feita, através de caixas de diálogo, a seleção de tabelas, para posterior implementação física.

FIGURA 7 – Selecionando plataforma para geração física

FIGURA 8 – Selecionando Tabelas

O Dr.CASE também automatiza a implementação física das consultas, que devem ser escritas em uma linguagem de manipulação de dados apropriada, no nosso caso SQL.

Os recursos disponibilizados pela ferramenta também automatizaram a definição do esquema de banco de dados, a nível físico. Após as tabelas terem sido refinadas durante o projeto lógico, foi feita a geração dos script's para a criação da base de dados. Além disso, também foram gerados os script's para manipulação dos dados, com base nas consultas escritas em SQL, como mostra a figura 9.

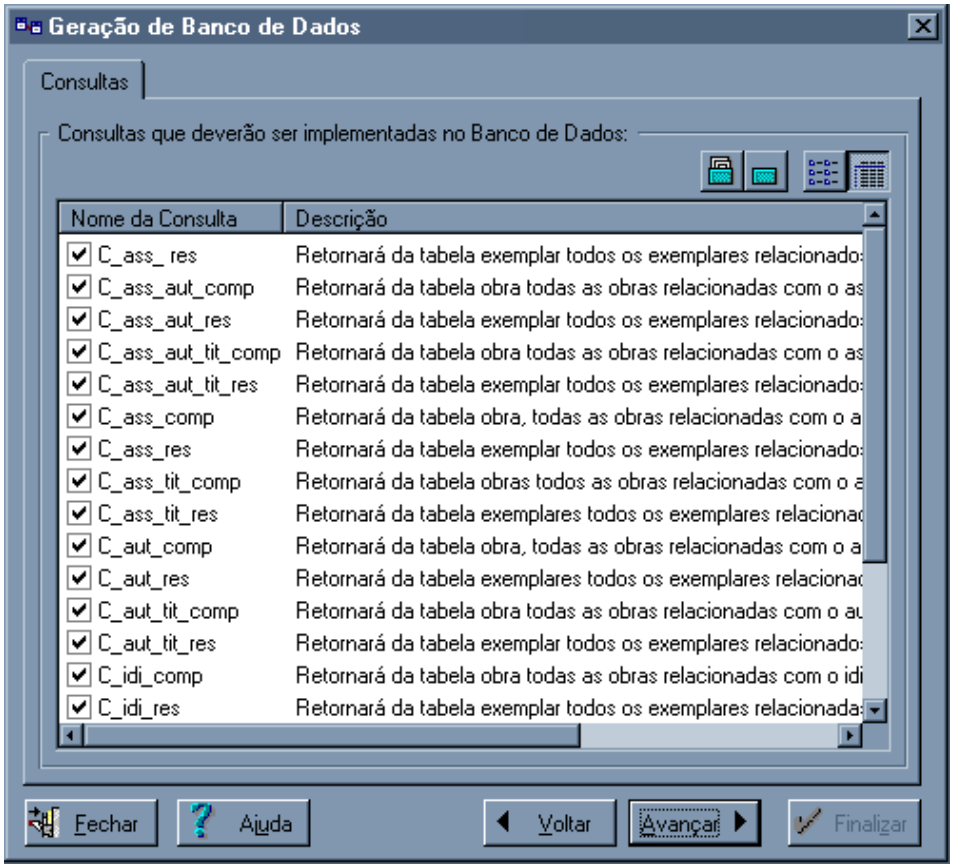

FIGURA 9 – Selecionando Consultas

6.4 Domínios no Dr.CASE

O usuário do Dr.CASE pode criar tipos personalizados para classificar os valores possíveis dos atributos e campos do sistema. Para possibilitar a implementação destes tipos em um SGBD específico, o usuário deve também fazer a associação com os tipos existentes nos SGBDs de destino.

Desta forma, tem-se flexibilidade para definir tipos de dados sem perder uma característica importante do Dr.CASE, a possibilidade de um mesmo projeto ser usado para implementar banco de dados em SGBDs diferentes.

Cabe destacar que, em um determinado projeto, o usuário pode utilizar tanto os tipos do banco de dados selecionado (DB2, por exemplo) quanto os domínios criados.

7. O SISTEMA PROPOSTO

Como solução dos problemas já apresentados, propõe-se neste capítulo uma reestruturação do esquema de bancos de dados, do atual sistema da Biblioteca Setorial de Ciência e Tecnologia da UFPel. Este sistema, como já foi dito, está baseado numa plataforma praticamente obsoleta, o que impõe uma série de restrições quanto ao seu uso e manutenção. Este sistema procurará suprir todas as necessidades da biblioteca, bem como as da UFPel, por isso foram realizadas entrevistas na intenção de descobrir os reais anseios da comunidade.

Neste capítulo será apresentado o esquema de banco de dados, nos três níveis de abstração (físico, lógico e conceitual), que foi definido com o auxílio da ferramenta case Dr.CASE 3.0, descrita no Capítulo 6, o que proporciona maior confiabilidade, maior flexibilidade para mudanças, melhor documentação, entre outras vantagens já citadas anteriormente.

O banco de dados foi definido com base nos tipos de dados padrões do SGBD DB2, o que facilitará migração dos dados do sistema atual, que também estão armazenados nessa plataforma. A criação e posterior manipulação dos dados, foi feita através da linguagem SQL, que viabiliza a implementação total do sistema, visto que a SQL tem a característica de poder ser utilizada em linguagens de programação hospedeiras (como o DELPHI).

7.1 Projeto Conceitual

Nesta fase do projeto, temos uma visão global do sistema, pois são irrelevantes para o desenvolvedor os detalhes da estrutura do banco de dados. Deve-se apenas observar quais dados são relevantes e como eles estão relacionados. Para realização desta etapa foi adotado o modelo *entidade-relacionamento*, que será descrito durante a apresentação do esquema de banco de dados conceitual.

7.1.1 Modelo entidade-relacionamento

O esquema de banco de dados, em nível conceitual, foi definido com base no modelo *entidade-relacionamento* (E-R) criado por Peter Chen em 1976, com base na teoria relacional de E.F. Codd (1970).

Este modelo baseia-se na representação do sistema por um conjunto de objetos do mundo real (entidades) e por relacionamentos entre estes objetos. O modelo E-R nada mais é do que a visão de uma dada realidade, baseia-se no relacionamento entre entidades, os quais retratam os fatos que governam nesta realidade, e que cada entidade ou relacionamento pode possuir atributos (qualificadores desta realidade).

Este modelo foi desenvolvido para otimizar o projeto do banco de dados, pois tem como característica principal a facilidade de projetar esquemas de bancos de dados. O modelo E-R tornou-se o modelo mais usado para representação de bancos de dados em nível conceitual, pois permiti que seja feita a especificação do esquema da organização, que representa toda a estrutura lógica do banco de dados.

O esquema de banco de dados em nível conceitual, é uma das partes mais importantes do projeto, pois permiti uma visão geral, sem se preocupar com os detalhes, facilitando sua compreensão.

7.1.1.1 Diagrama entidade-relacionamento

O diagrama *entidade-relacionamento,* ou simplesmente DER, é a simbologia usada para representação do esquema do banco de dados no modelo E-R. O DER é uma ferramenta muito útil na modelagem de sistemas e se destina a representar os dados do sistema e os relacionamentos entre eles. Através do DER é possível ser expressado graficamente toda a estrutura lógica de um banco de dados.

A seguir serão apresentados seus principais componentes:

- **Retângulos:** representam as entidades.
- **Elipses:** representam os atributos.
- **Losangos:** representam os relacionamentos.
- **Linhas:** que unem os relacionamentos e os atributos com as entidades.
- **Elipses duplas:** que representam atributos multivalorados.
- **Linhas duplas:** que indicam participação total de uma entidade em um relacionamento.

O diagrama E-R referente ao sistema modelado, apresentado na figura 10, será profundamente analisado durante as seções subsequentes.

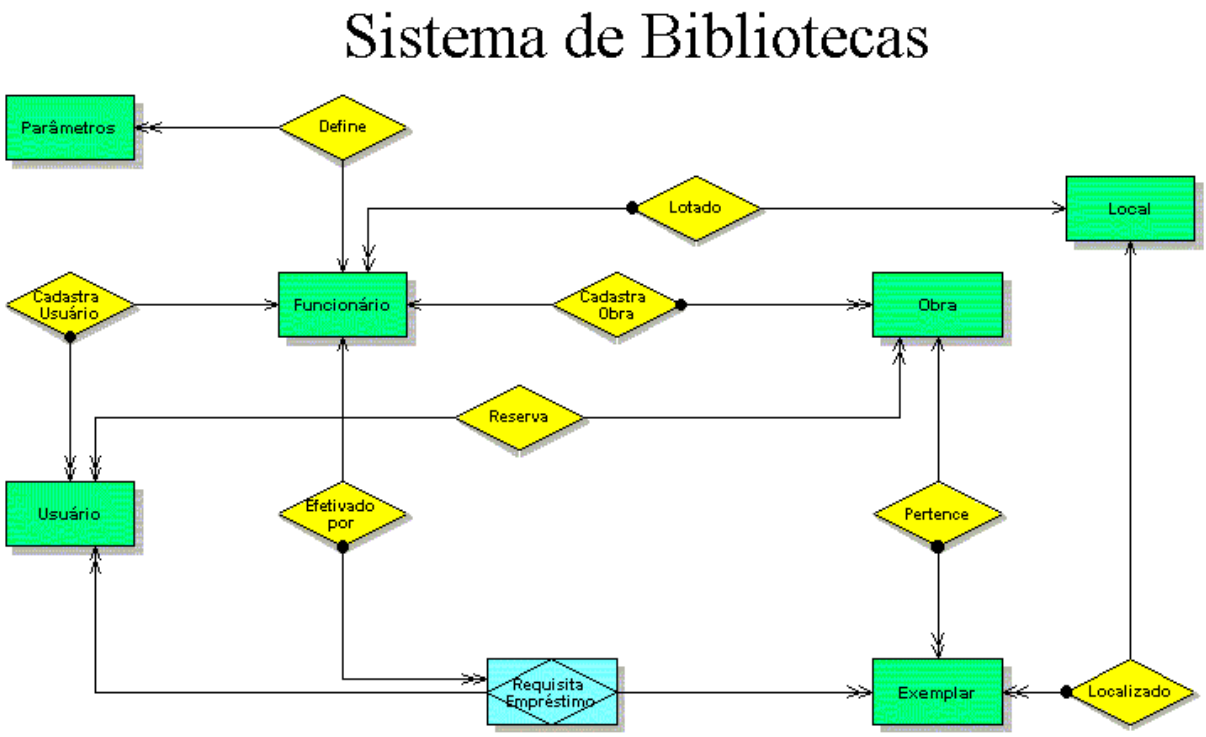

FIGURA 10 – Diagrama entidade-relacionamento do sistema modelado

7.1.1.2 Entidades

No modelo E-R, uma *entidade* pode ser definida como aquele "objeto" ou "coisa" que existe no mundo real com uma identificação própria e um significado próprio. Em outras palavras, são os objetos que descrevem o negócio. Se algum objeto, existente dentro de um negócio (realidade), proporciona interesse em manter informações (dados) sobre ele, isto o caracteriza como uma entidade [MAC96]. Por exemplo, no sistema de biblioteca que foi definido existe a entidade *usuário*, porque também há interesse nos dados referentes ao usuário, como nome, endereço, etc. Uma entidade não precisa ser obrigatoriamente um objeto físico, como usuário, ela pode ser um objeto abstrato, desde que exista interesse sobre as informações referentes ao mesmo.

A representação de uma entidade no modelo E-R se realiza através de um retângulo, com o nome desta em seu interior, como na figura 11.

FIGURA 11 – Representação de entidade no modelo E-R

As instâncias (ocorrências) de uma entidade, não são representadas no diagrama E-R, mas são semanticamente interpretados dentro do mesmo.

Uma entidade deve ser entendida como uma tabela de dados, onde cada linha desta tabela representa um instância da mesma, sendo que cada instância é descrita pelos qualificadores (atributos) da entidade.

Durante a analise, realizada para identificar as entidades pertinentes ao sistema modelado, foram definidos como entidades os seguintes objetos:

- **Exemplar:** Cadastro de todos os exemplares (acervo das bibliotecas).
- **Funcionário:** Cadastro dos funcionários do sistema.
- **Obra:** Cadastro de todas as obras.
- **Parâmetros:** Contém os parâmetros do sistema.
- **Usuário:** Cadastro de usuários (alunos, professores e servidores).
- **Local:** Cadastro das bibliotecas setoriais do sistema.

7.1.1.3 Atributos

Qualquer objeto considerado como entidade possui propriedades que são descritas por *atributos* e *valores*. A designação de um atributo para uma entidade indica que o banco de dados mantém informações similares para cada uma das instâncias (ocorrências) da mesma. Mas, para cada ocorrência dentro de uma entidade existem valores próprios para cada um de seus atributos.

Considerando o sistema desenvolvido, temos uma entidade, um objeto, sobre a qual deseja-se manter informações armazenadas, chamado funcionário e existem atributos que descrevem cada instância desta entidade (cada funcionário da biblioteca). Os atributos que descrevem o funcionário são código do funcionário, nome do funcionário, função, senha de acesso ao sistema e código da biblioteca setorial onde o mesmo trabalha. Cada instância da entidade funcionário será formada por valores nestes atributos, este conjunto de valores deve ser visualizado como uma linha de uma tabela de dados que representa a entidade no todo.

Os valores de um ou mais atributos, nas ocorrências de uma entidade, são sempre diferentes para cada instância, caracterizando que não existem objetos repetidos dentro de uma entidade. Este conjunto de atributos cujos valores nunca se repetem tem a função de atuar como identificador único das instâncias de uma entidade. Dentro da abordagem relacional de banco de dados, denomina-se *chave primária* de uma tabela o conjunto de um ou mais atributos concatenados que identifica uma única ocorrência dentro de uma tabela. Por exemplo, o atributo *Código do Usuário* identifica univocamente um usuário.

No caso da entidade funcionário, o atributo que identifica cada uma das ocorrências é o *Código do Funcionário*.

Os valores dos atributos que descrevem as entidades constituem uma fatia significativa dos dados que serão armazenados no banco de dados. Os atributos das entidades são definidos durante a construção do modelo E-R, e podem ser caracterizados pelos seguintes tipos.

- Atributos **simples** ou **composto:** Os atributos simples são aqueles que não divididos em partes. Os atributos compostos são aqueles divididos em mais de um atributo. Os atributos compostos possibilitam que atributos correlacionados fiquem agrupados, tornando o modelo mais claro. Um de exemplo atributo composto seria, ao invés de utilizar-se um campo para o nome completo de uma pessoa, utilizar um atributo para nome, outro para nome intermediário e outro para sobrenome.
- Atributos **monovalorados** ou **multivalorados:** os atributos monovalorados são aqueles que podem assumir um valor único para cada ocorrência de uma entidade. Já os atributos multivalorados são aqueles que podem assumir mais de um valor para cada instância de uma entidade. Se houver necessidade é possível estabelecer limites inferiores e superiores que indicam quantos valores pode assumir um atributo multivalorado.
- Atributos **nulos:** um atribulo nulo ocorre quando as instâncias de uma entidade podem assumir um valor desconhecido para o mesmo. O valor desconhecido indica que valor do atributo não está disponível, por não existir ou por ter ser sido omitido.
- Atributo **derivado:** o atributo derivado é aquele cujo valor pode ser derivado de outros atributos ou entidades a ele relacionados.

Os atributos definidos durante o projeto conceitual poderão ser melhor visualizados na definição da estrutura das tabelas do sistema (seção 7.2).

7.1.1.4 Relacionamentos

Entender perfeitamente o que são relacionamentos dentro do modelo E-R, facilita muito no projeto de banco de dados, por isso esta seção se dedica ao mesmos.

Um relacionamento poderia ser definido como o fato (acontecimento) que liga dois objetos existentes no mundo real. Considerando sistemas relacionais, pode-se estender este conceito como sendo relacionamento o fato que efetua a junção de duas ou mais tabelas de dados.

Os relacionamentos são divididos nos seguintes tipos:

- **Relacionamentos Binários:** representa a associação entre duas entidade, é graficamente representado por um losango nomeado com linhas para ambas as entidades envolvidas. Como um relacionamento identifica uma função exercida pelas

duas entidade envolvidas, é aconselhável que o mesmo seja nomeado com um substantivo plural, pois o uso de verbos é um limitador para a criatividade para a determinação de nomes. No momento que um relacionamento é definido, também deve-se definir o grau (ou cardinalidade) do mesmo, esse aspecto será explanado posteriormente.

- **Relacionamentos Reflexivos:** é aquele onde há associação entre ocorrências de uma mesma entidade. Neste caso específico, é extremamente recomendável que fique indicado explicitamente qual a função exercida por cada lado da associação, para evitar interpretações dúbias. Por exemplo, uma relação de chefia entre dois funcionários, é extremamente recomendado que sejam explicitados os papéis do superior e do subordinado.
- **Relacionamentos Ternários:** ocorre quando três entidades são relacionadas. Este tipo de relacionamento deve utilizado somente quando é realmente necessário associar, simultaneamente, um par de entidades com uma terceira. Por exemplo, supondo três entidades *aluno, professor* e *disciplina,* sabe-se que um aluno pode estar matriculado em uma ou mais disciplinas, e que um disciplina é cursada por no mínimo um aluno, sendo que a cada disciplina é ministrada por um professor. Neste caso, é possível enxergar claramente o relacionamento entre as três entidades, quando isto não ocorre é extremamente recomendável o uso de agregação.

No DER do sistema, foram definidos somente relacionamentos binários entre as entidades, eles são apresentados na tabela 1.

TABELA 1 – Relacionamentos do sistema

Título	Entidades Envolvidas	Descrição
Cadastra Obra	Funcionário Obra	Indica que o cadastro de obra é feito por um funcionário habilitado
Cadastra Usuário	Funcionário Usuário	Indica que todo usuário é cadastrado por um funcionário.
Define	Funcionário Parâmetros	Indica que os parâmetros podem ser alterados por um funcionário habilitados.
Efetivado por	Funcionário Requisita Empréstimo (Agregação)	Indica que todos os empréstimos deve ser efetivados por um funcionário habilitado.
Pertence	Exemplar Obra	Indica que cada obra possui um ou mais exemplares, enquanto que cada exemplar pertence somente à uma obra.
Requisita Empréstimo	Usuário Exemplar	Indica que os usuários podem requisitar o empréstimo de um ou mais exemplares, este relacionamento é representado logicamente pela tabela empréstimos.

Reserva	Usuário Obra	Indica que os usuários do sistema podem efetivar a reserva de uma ou mais de obras, este relacionamento é representado logicamente pela tabela reserva.
Lotado	Funcionário Local	Indica que todo o funcionário está lotado em uma biblioteca setorial.
Localizado	Exemplar Local	Indica que todo exemplar está localizado em uma biblioteca setorial.

7.1.1.5 Grau do Relacionamento

Quando temos um relacionamento entre duas entidades, o número de ocorrências de uma entidade que está associado com as ocorrências da outra entidade, determina o Grau do relacionamento, ou Cardinalidade do mesmo. A cardinalidade dos relacionamentos será discutida a seguir.

7.1.1.6 Relacionamento de Um-para-Muitos (1:N)

Este tipo de relacionamento é o mais comum no mundo real, sendo o mesmo denominado de relacionamento básico entre entidades.

Considerando duas entidades hipotéticas A e B, existe um relacionamento de cardinalidade Um-para-Muitos entre elas, se e somente se, cada elemento da entidade A relaciona-se com muitos (um ou mais) elementos da entidade B, enquanto cada elemento da entidade B relaciona-se com apenas um elemento da entidade A.

No diagrama E-R do sistema modelado existem os seguintes relacionamentos deste tipo:

TABELA 2 – Relacionamentos Um-para-Muitos

Título	Entidades Envolvidas	Entidade (1)	Entidade (N)
Cadastra Obra	Funcionário Obra	Funcionário	Obra
Cadastra Usuário	Funcionário Usuário	Funcionário	Usuário
Define	Funcionário Parâmetros	Funcionário	Parâmetros
Efetivado por	Funcionário Requisita Empréstimo (Agregação)	Funcionário	Requisita Empréstimo (Agregação)
Localizado	Exemplar Local	Local	Exemplar
Lotado	Funcionário Local	Local	Funcionário
Pertence	Exemplar Obra	Obra	Exemplar

Estes relacionamentos são efetivados, a nível lógico, através da chamada *chave estrangeira*, que é mapeada na tabela referente à entidade que possui muitos elementos envolvidos no relacionamento.

7.1.1.7 Relacionamento de Muitos-para-Muitos (N:M)

Neste relacionamento, cada elemento de ambas as entidades relaciona-se com um ou mais elementos da outra entidade.

Relacionamentos deste tipo ocorrem se, entre duas entidades hipotéticas A e B, cada elemento da entidade A está relacionado com um ou mais elementos da entidade B e vice-versa.

Este tipo de relacionamento possui uma característica que lhe é extremamente peculiar, ele pode possuir atributos. Isto que dizer que este relacionamento pode possuir dados que são inerentes somente ao fato e não as entidades.

A nível lógico, este relacionamento é efetivado através de uma tabela que contém os atributos que compõem a *chave primária* das entidades envolvidas e, os atributos pertinentes ao mesmo.

Na tabela 3, temos a listagem de todos os relacionamentos deste tipo.

TABELA 3 – Relacionamentos Muitos-para-Muitos

Título	Entidades Envolvidas	Atributos
Requisita Empréstimo	Usuário Exemplar	Código do Empréstimo Data da Retirada Data de Entrega Dias de Atraso
Reserva	Usuário Obra	Código da Reserva Data da reserva Atendida Data de Atendimento

7.1.1.8 Relacionamentos Incondicionais

Relacionamentos incondicionais são aqueles onde cada elemento de uma entidade está obrigatoriamente relacionado com pelo menos um elemento da outra entidade. Esta característica é chamada de totalidade, sendo representa no diagrama E-R através de uma esfera preta desenhada na(s) extremidade(s) do relacionamento.

Na tabela 4 pode-se observar os relacionamentos incondicionais do sistema modelado.

TABELA 4 – Relacionamentos Incondicionais

Título	Entidades Envolvidas	Totalidade
Cadastra Obra	Funcionário Obra	Toda obra é obrigatoriamente cadastrada por um funcionário.

Cadastra Usuário	Funcionário Usuário	Todo usuário é obrigatoriamente cadastrado por um funcionário.
Efetivado por	Funcionário Requisita Empréstimo (Agregação)	Todo empréstimo é obrigatoriamente efetivado por um funcionário.
Localizado	Exemplar Local	Toda exemplar está obrigatoriamente localizado em uma biblioteca setorial.
Lotado	Funcionário Local	Todo o funcionário está obrigatoriamente lotado em uma biblioteca setorial.
Pertence	Exemplar Obra	Todo exemplar está obrigatoriamente relacionado à uma obra.

7.1.1.9 Relacionamentos Condicionais

São efetivamente aqueles relacionamentos em que nem todos os elementos de uma entidade A estão relacionados com elementos de uma entidade B. Relacionamentos deste tipo possuem a característica chamada de opcionalidade.

Na tabela 5 estão descritos os relacionamentos que possuem esta característica.

TABELA 5 – Relacionamentos Condicionais

Título	Entidades Envolvidas
Define	Funcionário Parâmetros
Requisita Empréstimo	Usuário Exemplar
Reserva	Usuário Obra

7.1.2 Agregação

Agregação é uma abstração através da qual os relacionamentos são tratados como entidades, podendo assim se relacionar entre si mesmo ou com outras. Em diversos casos, para se manter consistente o DER, é necessário lançar mão deste artifício.

No DER do sistema o relacionamento “Requisita Empréstimo” foi efetivado como uma agregação, pois também foi definido que todo empréstimo necessita ser efetivado por um funcionário. Sendo representada por um losango dentro de um retângulo.

7.2 Projeto Lógico

Esta etapa é dedicada, dentro de um projeto de banco de dados, ao refinamento dos dados obtidos durante o projeto conceitual. São definidas as estruturas das tabelas que representaram entidades, relacionamentos e seus respectivos atributos. Para definição do

esquema de banco de dados lógico, foi utilizado o modelo *relacional*, que será apresentado a seguir.

7.2.1 Modelo Relacional

O *modelo relacional* é a base de toda a moderna tecnologia que encontramos atualmente na área de banco de dados. Este modelo foi formalmente definido no ano de 1970 por E. F. Codd, que realizou sua pesquisa nos Laboratórios da IBM localizados em San José – Califórnia. O projeto inicialmente chamava-se de *System R* e definia como os dados seriam organizados e linguagens formais para sua manipulação. A primeira versão da linguagem SQL (*Structured Query Language)* foi definida com base nessas linguagens formais. A linguagem SQL consolidou-se como um padrão quando se fala em gerenciamento de dados em SGBDs relacionais.

Dentro deste modelo entidades e relacionamentos são representados por relações, que são equivalentes ao conceito matemático de conjunto, isto é, um agrupamento de elementos sem repetição. Uma relação sempre é vista como uma tabela bidimensional, onde cada linha representa uma ocorrência (tupla) dentro da relação, e as colunas representam os campos (atributos) pertinentes a cada ocorrência.

Neste modelo os relacionamentos do tipo Muitos-para-Muitos são estabelecidos através de tabelas que contém os campos que compõem a *chave primária* das entidades envolvidas, nessa tabela também serão armazenados os dados referentes ao relacionamento, caso existam.

Os relacionamentos do tipo Um-para-muitos são modelados através da *chave externa*, que é mapeada na tabela que representa a entidade que possui muitos elementos envolvidos no relacionamento.

7.2.1.1 Chave Primária

A *chave primária* (Primary Key – PK) é um conjunto de um ou mais campos que identificam univocamente uma tupla dentro de uma tabela. Além disso, a chave primária é a principal forma de acesso as tuplas de uma relação.

7.2.1.2 Chave Estrangeira

A *chave estrangeira* (Foreign Key – FK) pode ser definida como o conjunto de um ou mais campos de uma tabela que fazem referência à chave primária de outra tabela para fins de manutenção de integridade e consistência de dados do sistema. Temos como exemplo de

chave estrangeira o campo *CodObra* da tabela exemplar, pois indica que cada exemplar refere-se a uma obra identificada pela sua chave primária (CodObra).

7.2.3 Definição das estruturas das tabelas

Nesta seção serão definidas as tabelas que representam, logicamente, as entidades e os relacionamentos descritos no projeto conceitual.

7.2.3.1 Tabela USUÁRIO

A tabela USUÁRIO armazena os dados dos usuários cadastrados no sistema. Os usuários podem ser do tipo aluno, professor ou servidor e, segundo a política vigente na Biblioteca Setorial de Ciência e Tecnologia, possuem os mesmos direitos quanto ao uso da mesma (empréstimos, reservas). Sua estrutura é demonstrada na tabela 6.

TABELA 6 – Tabela USUÁRIO

Campo	Tipo	Observação	Descrição
CodUser	*Character (10)*	*PK, Not Null*	Indica o código que identifica o usuário no sistema. Esse código é o número da matrícula (aluno), ou o número do SIAPE (professor e servidor).
NomeUser	*Character (50)*	*Not Null*	Indica o nome do usuário.
Vinculo	*Character (1)*	*Not Null*	Indica o tipo de vinculo do usuário para com a biblioteca. A- aluno, P- professor, S- servidor.
RG	*Character (10)*	*Not Null*	Indica o número da carteira de identidade.
CodCurso	*Character (8)*	*Not Null*	Indica o código do curso do aluno, ou da lotação no caso de professor ou servidor.
Endereço	*Character (50)*	*Not Null*	Indica o endereço do usuário.
Bairro	*Character (20)*	*Not Null*	Indica o bairro onde o usuário reside.
Cidade	*Character (20)*	*Not Null*	Indica a cidade onde o usuário reside.
CEP	*Character (10)*	*Not Null*	Indica o CEP referente ao endereço do usuário.
UF	*Character (2)*	*Not Null*	Indica o estado onde o usuário reside.
Email	*Character (20)*		Indica o endereço de correio eletrônico do usuário.
Fone	*Character (7)*		Indica o número de telefone do usuário.
DDD	*Character (3)*		Indica o código de discagem direta a distância referente ao telefone do usuário.
SenhaUser	*Character (8)*	*Not Null*	Indica a senha do usuário, requerida no momento em que o usuário efetivar reserva.
DiasMulta	*Smallint*		Indica o número de dias de multa que o usuário ainda tem pendente.

Situacao	*Character (8)*	*Not Null*	Indica a situação do usuário perante o sistema, podendo ser: H- usuário habilitado a usufruir dos serviços da biblioteca; A- usuário afastado da biblioteca; D- usuário em débito com a biblioteca.
CodFunc	*Character (10)*	*FK, Not Null*	Indica o código do funcionário que cadastrou o usuário no sistema.

7.2.3.2 Tabela OBRA

Esta tabela contém os dados referentes as obras castradas no sistema. Sua estrutura é demonstrada na tabela 3. Deve-se observar a presença do campo multivalorado *Idioma.* Sua estrutura é apresentada na tabela 7.

TABELA 7 – Tabela OBRA

Campo	Tipo	Observação	Descrição
CodObra	*Character (10)*	*PK, Not Null*	Indica o código que identifica a obra no sistema.
Titulo	*Character (50)*	*Not Null*	Indica o título principal da obra.
Autor	*Character (50)*	*Not Null*	Indica o autor principal da obra. Podendo referir-se à nome pessoal, entidade coletiva ou evento.
NumPadrao	*Character (15)*	*Not Null*	Indica o número padrão para a identificação da obra, varia conforme o tipo da mesma, no caso de livro ISBN, de periódico ISSN, etc.
Edicao	*Smallint*	*Not Null*	Indica o número de edição da obra.
Imprenta	*Character (100)*		Indica as notas tipográficas da obra.
Ano1	*Smallint*	*Not Null*	Indica o ano em que a obra foi publicada.
Ano2	*Smallint*		Indica o ano em que foi encerrada a publicação.
NumCham	*Character (30)*	*Not Null*	Indica o número de chamada de uma obra.
Tipo	*Character (30)*	*Not Null*	Indica o tipo da obra (ex: livro, periódico, etc.).
QtdeExemplares	*Smallint*	*Not Null*	Indica a quantidade de exemplares que uma obra possui.
NumPag	*Integer*	*Not Null*	Indica o número de páginas de uma obra. Se a obra for do tipo coleção, indica o número do volume referido.
Serie	*Character (50)*		Indica o título da série a que a obra pertence.
Notas	*Character (100)*		Indica notas gerais, a respeito da obra, que não foram incluídas em nenhum dos outros campos
SecAutor	*Character (100)*		Indica o nome dos co-autores da obra, bem como nome de colaboradores, tradutores, etc.

SecTitulo	*Character (50)*		Indica outros títulos que referenciam a obra. Ex: título original, no caso de obra traduzida.
Thesaurus	*Character (100)*	*Not Null*	Indica o nome dos Thesaurus que estão sendo utilizados para identificar os assuntos referentes a cada obra. No caso da BS/CT o Thesaurus utilizado é o Spines.
Assunto	*Character (100)*	*Not Null*	Indica os descritores do Thesaurus Spines a que a obra se relaciona. Em outras palavras, nada mais é do que uma lista de assuntos que podem ser relacionados à obra.
Classificacao	*Character (4)*	*Not Null*	Indica o tipo de classificação usada (CDU, CDD, outras).
NumRegistro	*Character (8)*	*Not Null*	Indica o número de registro patrimonial da obra.
Livre	*Character (100)*		Indica termos livres relacionados à obra, não relacionados no Thesaurus Spines.
DataInclusao	*Date*	*Not Null*	Indica a data de inclusão da obra no acervo da biblioteca.
Idioma_1	*Character (3)*	*Not Null*	Indica um dos idiomas em que obra foi publicada.
Idioma_2	*Character (3)*		Indica um dos idiomas em que obra foi publicada.
Idioma_3	*Character (3)*		Indica um dos idiomas em que obra foi publicada.
CodFunc	*Character (10)*	*FK, Not Null*	Indica o código do funcionário que cadastrou a obra no sistema.

7.2.3.3 Tabela EXEMPLAR

Esta tabela armazena os dados dos exemplares cadastrados no sistema. Como todo exemplar é referente à uma obra, existe a chave externa *CodObra*, que indica a qual obra o exemplar refere-se. Além disso, também existe a chave externa *CodLocal*, que identifica a biblioteca onde o exemplar está armazenado.

Sua estrutura é demonstrada na tabela 8.

TABELA 8 – Tabela EXEMPLAR

Campo	Tipo	Observação	Descrição
CodExemplar	*Character (12)*	*PK, Not Null*	Indica o código que identifica o exemplar no sistema.
Status	*Character (1)*	*Not Null*	Indica a situação do exemplar no momento do empréstimo. Podendo ser: B- baixa, D- disponível , E- emprestado, L- consulta local, F- consulta local mas momentaneamente fora da biblioteca, P- empréstimo proibido, R- reservado.

Volume	*Smallint*		Indica o número do volume ao qual o exemplar pertence, no caso de uma coleção.
DataCadastro	*Date*	*Not Null*	Indica a data em que o exemplar foi cadastrado.
DataBaixa	*Date*		Indica a data da baixa do exemplar, ou seja, indicará a data de exclusão do mesmo, no caso desta ocorrer, pois um exemplar só pode ser excluído do sistema se a obra a qual ele se refere for também excluída.
CodObra	*Character (10)*	*FK, Not Null*	Indica o código da obra a qual o exemplar refere-se.
CodLocal	*Character (1)*	*FK, Not Null*	Indica o local, isto é, o código da biblioteca setorial onde o exemplar se encontra.

7.2.3.4 Tabela EMPRÉSTIMO

Este tabela contém os dados referentes a todos os empréstimos que ainda não foram finalizados. Após ser concluído, os dados referentes ao empréstimo serão armazenados na tabela histórico para posterior geração de relatórios. Como todo empréstimo é efetivado por um funcionário pode-se observar a existência da chave estrangeira *CodFunc*. Sua estrutura é listada na tabela 9.

TABELA 9 – Tabela EMPRÉSTIMO

Campo	Tipo	Observação	Descrição
CodUser	*Character (10)*	*PK, Not Null*	Indica o código do usuário que realizou o empréstimo.
CodExemplar	*Character (12)*	*PK, Not Null*	Indica o código do exemplar que foi emprestado.
CodEmprestimo	*Character (10)*	*Not Null*	Indica o código do empréstimo.
DataRetirada	*Date*	*Not Null*	Indica a data em que foi efetivado o empréstimo.
DataEntrega	*Date*	*Not Null*	Indica a data prevista para a entrega do material retirado.
DiasAtraso	*Smallint*		Indica os dias de atraso, caso existam.
CodFunc	*Character (10)*	*FK, Not Null*	Indica o código do usuário que efetivou o empréstimo.

7.2.3.5 Tabela FUNCIONÁRIO

A Tabela FUNCIONÁRIO contém os dados referentes aos funcionários da biblioteca. Sua estrutura é descrita na tabela 10.

TABELA 10 – Tabela FUNCIONÁRIO

Campo	Tipo	Observação	Descrição
CodFunc	*Character (10)*	*PK, Not Null*	Indica o código do funcionário no sistema, que é o seu respectivo número do SIAPE.
NomeFunc	*Character (50)*	*Not Null*	Indica o nome do funcionário.
Funcao	*Character (1)*	*Not Null*	Indica a função que os funcionários exercem na biblioteca. Isto é, sua classe A,B,C ou D.

SenhaFunc	*Character (8)*	*Not Null*	Indica a senha do funcionário para acesso ao sistema.
CodLocal	*Character (1)*	*FK, Not Null*	Indica o código da biblioteca setorial onde funcionário trabalha.

7.2.3.6 Tabela RESERVA

A Tabela RESERVA armazena os dados referentes as reservas. A estrutura da mesma é descrita na tabela 11.

TABELA 11 – Tabela RESERVA

Campo	Tipo	Observação	Descrição
CodUser	*Character (10)*	*PK, Not Null*	Indica o código do usuário que efetivou a reserva.
CodObra	*Character (10)*	*PK, Not Null*	Indica o código da obra que foi reservada.
CodReserva	*Character (10)*	*Not Null*	Indica o código da reserva.
DataReserva	*TimeStamp*	*Not Null*	Indica a data da reserva, incluindo dia, mês, ano, hora, minuto e segundo da mesma.
Atendida	*Character (1)*		Indica se a reserva já foi atendida
DataAtendimento	*TimeStamp*		Indica a data em que a reserva foi atendida.

7.2.3.7 Tabela PARÂMETROS

Nesta tabela serão definidos os parâmetros do sistema, os parâmetros podem ser alterados por um funcionário autorizado, isto caracteriza a presença da chave estrangeira *CodFunc*. Temos sua estrutura na tabela 12.

TABELA 12 – Tabela PARÂMETROS

Campo	Tipo	Observação	Descrição
NomeInst	*Character (50)*	*PK, Not Null*	Indica o nome da instituição que utiliza o sistema.
MaxFila	*Smallint*	*Not Null*	Indica o número máximo de usuários na fila de reserva por obra.
HorasDisp	*Smallint*	*Not Null*	Indica o número máximo de horas que o usuário tem um exemplar disponível após sua reserva ser atendida.
AtualFila	*TimeStamp*		Indica a hora que expira a reserva mais antiga da fila de reservas, isto é, indica quando a fila de reserva deve ser atualizada.
MaxExemp	*Smallint*	*Not Null*	Indica o número máximo de exemplares que o usuário pode ter emprestados simultaneamente.
CodFunc	*Character (10)*	*FK, Not Null*	Indica o código do funcionário que alterou os parâmetros.

7.2.3.8 Tabela LOCAL

Nesta tabela são armazenados os dados referentes as bibliotecas setoriais cadastradas no sistema. Na tabela 13 é demonstrada a sua estrutura.

TABELA 13 – Tabela LOCAL

Campo	Tipo	Observação	Descrição
CodLocal	*Character (1)*	*PK, Not Null*	Indica o código da biblioteca setorial.
NomeLocal	*Character (50)*	*Not Null*	Indica o nome da biblioteca setorial.

7.2.3.9 Tabelas OBRA_EXCLUÍDA, EXEMPLAR_EXCLUÍDO e HISTÓRICO

Esta tabelas só a existem a nível lógico, pois foram definidas apenas para armazenar os dados que foram excluídos das tabelas OBRA, EXEMPLAR e EMPRÉSTIMO sistema, com o intuito de permitir uma posterior geração de relatórios. A estrutura destas tabelas é praticamente idêntica a das tabelas mães, exceto pelo fato de possuírem um campo a mais destinado a armazenar a data de exclusão. Suas estruturas serão demonstradas a seguir.

TABELA 14 – Tabela OBRA_EXCLUÍDA

Campo	Tipo	Observação	Descrição
CodObraEx	*Character (10)*	*PK, Not Null*	Indica o código que identificava a obra no sistema.
TituloEx	*Character (50)*	*Not Null*	Indica o título principal da obra.
AutorEx	*Character (50)*	*Not Null*	Indica o autor principal da obra. Podendo referir-se à nome pessoal, entidade coletiva ou evento.
NumPadraoEx	*Character (15)*	*Not Null*	Indica o número padrão para a identificação da obra, varia conforme o tipo da mesma, no caso de livro ISBN, de periódico ISSN, etc.
EdicaoEx	*Smallint*	*Not Null*	Indica o número de edição da obra.
ImprentaEx	*Character (100)*		Indica as notas tipográficas da obra.
Ano1Ex	*Smallint*	*Not Null*	Indica o ano em que a obra foi publicada.
Ano2Ex	*Smallint*		Indica o ano em que foi encerrada a publicação.
NumChamEx	*Character (30)*	*Not Null*	Indica o número de chamada de uma obra.
TipoEx	*Character (30)*	*Not Null*	Indica o tipo da obra (ex: livro, periódico, etc.).
QtdeExemplaresEx	*Smallint*	*Not Null*	Indica a quantidade de exemplares que uma obra possui.
NumPagEx	*Integer*	*Not Null*	Indica o número de páginas de uma obra. Se a obra for do tipo coleção, indica o número do volume referido.
SerieEx	*Character (50)*		Indica o título da série a que a obra pertence.
NotasEx	*Character (100)*		Indica notas gerais, a respeito da obra, que não foram incluídas em nenhum dos outros campos
SecAutorEx	*Character (100)*		Indica o nome dos co-autores da obra, bem como nome de colaboradores, tradutores, etc.

SecTituloEx	*Character (50)*		Indica outros títulos que referenciam a obra. Ex: título original, no caso de obra traduzida.
ThesaurusEx	*Character (100)*	*Not Null*	Indica o nome dos Thesaurus que estão sendo utilizados para identificar os assuntos referentes a cada obra. No caso da BS/CT o Thesaurus utilizado é o Spines.
AssuntoEx	*Character (100)*	*Not Null*	Indica os descritores do Thesaurus Spines a que a obra se relaciona. Em outras palavras, nada mais é do que uma lista de assuntos que podem ser relacionados à obra.
ClassificacaoEx	*Character (4)*	*Not Null*	Indica o tipo de classificação usada (CDU, CDD, outras).
NumRegistroEx	*Character (8)*	*Not Null*	Indica o número de registro patrimonial da obra.
LivreEx	*Character (100)*		Indica Termos livres relacionados à obra, não relacionados no Thesaurus Spines.
DataInclusaoEx	*Date*	*Not Null*	Indica a data de inclusão da obra no acervo da biblioteca.
IdiomaEx_1	*Character (3)*	*Not Null*	Indica um dos idiomas em que obra foi publicada.
IdiomaEx_2	*Character (3)*		Indica um dos idiomas em que obra foi publicada.
IdiomaEx_3	*Character (3)*		Indica um dos idiomas em que obra foi publicada.
CodFuncEx	*Character (10)*	*FK, Not Null*	Indica o código do funcionário que cadastrou a obra no sistema.
DataExObra	*Date*	*Not Null*	Indica a data de exclusão da obra.

TABELA 15 – Tabela EXEMPLAR_EXCLUÍDO

Campo	Tipo	Observação	Descrição
CodExemplarEx	*Character (12)*	*PK, Not Null*	Indica o código que identificava o exemplar no sistema.
StatusEx	*Character (1)*	*Not Null*	Indica a situação do exemplar no momento do empréstimo. No caso B (baixa).
VolumeEx	*Smallint*		Indica o número do volume ao qual o exemplar pertence, no caso de uma coleção.
DataCadastroEx	*Date*	*Not Null*	Indica a data em que o exemplar foi cadastrado.
DataBaixaEx	*Date*	*Not Null*	Indica a data da baixa do exemplar, ou seja, indicará a data de exclusão do mesmo, no caso desta ocorrer, pois um exemplar só pode ser excluído do sistema se a obra a qual ele se refere for também excluída.

CodObra	*Character (10)*	*FK, Not Null*	Indica o código da obra a qual o exemplar referia-se.
CodLocal	*Character (1)*	*FK, Not Null*	Indica o local, isto é, o código da biblioteca setorial onde o exemplar se encontrava.
DataExExemp	*Date*	*Not Null*	Indica a data em que o exemplar foi excluído do acervo.

TABELA 16 – Tabela HISTÓRICO

Campo	Tipo	Observação	Descrição
CodUserH	*Character (10)*	*PK, Not Null*	Indica o código do usuário que realizou o empréstimo.
CodExemplarH	*Character (12)*	*PK, Not Null*	Indica o código do exemplar que foi emprestado.
CodEmprestimoH	*Character (10)*	*Not Null*	Indica o código do empréstimo.
DataRetiradaH	*Date*	*Not Null*	Indica a data em que foi efetivado o empréstimo.
DataEntregaH	*Date*	*Not Null*	Indica a data prevista para a entrega do material retirado.
DiasAtrasoH	*Smallint*		Indica os dias de atraso, caso existam.
CodFuncH	*Character (10)*	*FK, Not Null*	Indica o código do usuário que efetivou o empréstimo.
DataExH	*Date*	*Not Null*	Indica a data em que o empréstimo foi finalizado.

7.3 Projeto Físico

Após ter sido definida a estrutura da base de dados, durante o projeto lógico, partiu-se para a definição do esquema de banco de dados físico, que foi feita através de script's em linguagem SQL, responsáveis pela criação e posterior manipulação da base de dados.

7.3.1 A Linguagem SQL

O nome SQL significa *Structured Query Language* (Linguagem Estruturada de Pesquisa). Esta linguagem, de grande utilização, baseou-se no modelo relacional de Codd (1970). Sua primeira versão recebeu o nome SEQUEL *(Structured English Query Language)* sendo definida por D.D. Chamberlin, entre outros, em 1974, nos laboratórios de pesquisa da IBM (Califórnia). Entre 1976 e 1977, a linguagem SEQUEL foi revisada e ampliada, e teve seu nome alterado para SQL, devido a questões jurídicas.

A idéia original da SQL só previa seu uso de forma interativa. Após sofrer alguns acréscimos, ela passou também a ter a capacidade de ser utilizada em linguagens hospedeiras, como COBOL, FORTRAN, C, DELPHI, etc.

A Linguagem SQL possui numerosas aplicações, podendo ser utilizada para executar as seguintes funções de um SGBD:

- **Definição de dados (DDL):** permite ao usuário a definição da estrutura e organização dos dados armazenados;
- **Manipulação de dados (DML):** permite ao usuário ou a um programa de aplicação, a inclusão, remoção, seleção ou atualização de dados previamente armazenados no banco;
- **Controle de acesso:** proteger os dados contra acessos e alterações não autorizadas;
- **Compartilhamento de dados:** coordenar o compartilhamento dos dados, por usuários concorrentes, sem contudo interferir na ação de cada um deles;
- **Integridade dos dados:** auxiliar no processo da definição da integridade do dados, protegendo-os contra corrupções, inconsistências e eventuais falhas do sistema.

No ano de 1982, o *American National Standard Institute* (ANSI) efetivou a SQL como a linguagem padrão oficial para uso em ambientes relacionais. Com a padronização seu uso tornou-se ainda mais difundido, estes dois fatores proporcionam ao analista as seguintes vantagens:

- **Independência do fabricante:** A SQL é oferecida em praticamente todos os SGBD's, e os que não têm estão se encaminhando para isso. Devido a este fato os desenvolvedores não precisam se preocupar com qual SGBD será utilizado como plataforma;
- **Portabilidade entre computadores:** A SQL pode ser utilizada desde um computador pessoal, passando por uma estação de trabalho, chegando até um mainframe;
- **Redução dos custos de treinamento:** Se baseando no item anterior, as aplicações podem se movimentar de um ambiente para outro sem que seja necessária uma reciclagem da equipe de desenvolvimento;
- **Inglês estruturado de alto nível:** A SQL é formada por um conjunto bem simples de sentenças em inglês, oferendo um rápido e fácil entendimento;
- **Consulta interativa:** A SQL provê um acesso rápido aos dados, fornecendo respostas ao usuário, a questões complexas, em minutos ou segundos;
- **Múltiplas visões de dados:** A SQL permite ao criador do banco de dados, levar diferentes visões dos dados a diferentes usuários;

- **Definição dinâmica dos dados:** Através da SQL, pode-se alterar, expandir ou incluir, dinamicamente, as estruturas dos dados armazenados, com a máxima flexibilidade;

A linguagem SQL também foi utilizada durante o transcorrer deste trabalho, no momento da criação da base de dados e suas respectivas consultas, pois permite uma perfeita adaptação ao SGBD adotado (DB2) e, além disso, viabiliza a total implementação do sistema, pois pode ser utilizada como hospedeira na maioria das linguagens de programação existentes no mercado.

7.3.2 Definição dos dados

A seguir temos a listagem do código fonte SQL para criação da base de dados, definida no projeto lógico.

```
--  Criação do Banco de Dados.
CREATE DATABASE SB;

--  Conexão ao banco de dados.
CONNECT TO SB;

--  Exclusão da tabela Emprestimo (Emprestimo)
DROP TABLE Emprestimo;

--  Exclusão da tabela Reserva (Reserva)
DROP TABLE Reserva;

--  Exclusão da tabela Local (Local)
DROP TABLE Local;

--  Exclusão da tabela ExempExcluido (ExempExcluido)
DROP TABLE ExempExcluido;

--  Exclusão da tabela ObraExcluida (ObraExcluida)
DROP TABLE ObraExcluida;

--  Exclusão da tabela Historico (Historico)
DROP TABLE Historico;

--  Exclusão da tabela Parametros (Parametros)
DROP TABLE Parametros;

--  Exclusão da tabela Funcionario (Funcionario)
DROP TABLE Funcionario;

--  Exclusão da tabela Exemplar (Exemplar)
DROP TABLE Exemplar;

--  Exclusão da tabela Obra (Obra)
DROP TABLE Obra;

--  Exclusão da tabela Usuario (Usuario)
DROP TABLE Usuario;

--  Criação da tabela Usuario (Usuario)
CREATE TABLE Usuario (
```

```
        CodUser                         CHAR(10)   NOT NULL,
        NomeUser                        CHAR(50)   NOT NULL,
        Vinculo                         CHAR(10)   NOT NULL,
        RG                              CHAR(10)   NOT NULL,
        CodCurso                        CHAR(8)    NOT NULL,
        Endereço                        CHAR(50)   NOT NULL,
        email                           CHAR(20),
        Bairro                          CHAR(20)   NOT NULL,
        Cidade                          CHAR(20)   NOT NULL,
        CEP                             INTEGER    NOT NULL,
        UF                              CHAR(2)    NOT NULL,
        DDD                             CHAR(3),
        Fone                            CHAR(7),
        SenhaUser                       CHAR(8)    NOT NULL,
        DiasMulta                       SMALLINT,
        Situacao                        CHAR(1)    NOT NULL,
        CodFunc                         CHAR(10)   NOT NULL);

--  Criação da tabela Obra (Obra)
CREATE TABLE Obra (
        CodObra                         CHAR(10)   NOT NULL,
        Titulo                          CHAR(50)   NOT NULL,
        Autor                           CHAR(50)   NOT NULL,
        NumPadrao                       CHAR(15)   NOT NULL,
        Edicao                          SMALLINT   NOT NULL,
        Imprenta                        CHAR(100),
        Ano1                            SMALLINT   NOT NULL,
        Ano2                            SMALLINT,
        NumCham                         CHAR(30)   NOT NULL,
        Tipo                            SMALLINT   NOT NULL,
        QtdeExemplares                  SMALLINT   NOT NULL,
        NumPag                          INTEGER    NOT NULL,
        Serie                           CHAR(50),
        Notas                           CHAR(100),
        SecAutor                        CHAR(100),
        SecTitulo                       CHAR(50),
        Thesaurus_                      CHAR(100)  NOT NULL,
        Assunto                         CHAR(100)  NOT NULL,
        Classificacao                   CHAR(4)    NOT NULL,
        NumRegistro                     CHAR(8)    NOT NULL,
        Livre                           CHAR(100),
        DataInclusao                    DATE       NOT NULL,
        CodFunc                         CHAR(10)   NOT NULL,
        Idioma_1                        CHAR(3)    NOT NULL,
        Idioma_2                        CHAR(3),
        Idioma_3                        CHAR(3));

--  Criação da tabela Exemplar (Exemplar)
CREATE TABLE Exemplar (
        CodExemplar                     CHAR(12)   NOT NULL,
        Status                          CHAR(1)    NOT NULL,
        Volume                          SMALLINT,
        DataCadastro                    DATE       NOT NULL,
        DataBaixa                       DATE,
        CodObra                         CHAR(10)   NOT NULL,
        CodLocal                        CHAR(1)    NOT NULL);

--  Criação da tabela Funcionario (Funcionario)
CREATE TABLE Funcionario (
        CodFunc                         CHAR(10)   NOT NULL,
        NomeFunc                        CHAR(50)   NOT NULL,
```

```
        Funcao                          CHAR(1)    NOT NULL,
        SenhaFunc                       CHAR(8)    NOT NULL,
        CodLocal                        CHAR(1)    NOT NULL);

--  Criação da tabela Parametros (Parametros)
CREATE TABLE Parametros (
        NomeInst                        CHAR(50)   NOT NULL,
        MaxFila                         SMALLINT   NOT NULL,
        HorasDisp                       SMALLINT   NOT NULL,
        AtualFila                       TIMESTAMP,
        MaxExemp                        SMALLINT   NOT NULL,
        CodFunc                         CHAR(10));

--  Criação da tabela Historico (Historico)
CREATE TABLE Historico (
        CodUserH                        CHAR(10)   NOT NULL,
        CodExempH                       CHAR(12)   NOT NULL,
        NomeUserH                       CHAR(50)   NOT NULL,
        CodEmpH                         CHAR(10)   NOT NULL,
        DataRetH                        DATE       NOT NULL,
        DataEntregaH                    DATE       NOT NULL,
        DiasAtrasoH                     SMALLINT,
        CodFuncH                        CHAR(10)   NOT NULL,
        DataExH                         DATE       NOT NULL);

--  Criação da tabela ObraExcluida (ObraExcluida)
CREATE TABLE ObraExcluida (
        CodObraEx                       CHAR(10)   NOT NULL,
        TituloEx                        CHAR(50)   NOT NULL,
        AutorEx                         CHAR(50)   NOT NULL,
        NumPadraoEx                     CHAR(15)   NOT NULL,
        EdicaoEx                        SMALLINT   NOT NULL,
        ImprentaEx                      CHAR(50)   NOT NULL,
        Ano1Ex                          SMALLINT   NOT NULL,
        Ano2Ex                          SMALLINT,
        NumChamEx                       CHAR(30)   NOT NULL,
        TipoEx                          CHAR(20)   NOT NULL,
        QtdeExemplaresEx                SMALLINT   NOT NULL,
        NumPagEx                        INTEGER    NOT NULL,
        SerieEx                         CHAR(50),
        NotasEx                         CHAR(50),
        SecAutorEx                      CHAR(100),
        SecTituloEx                     CHAR(50),
        ThesaCatEx                      CHAR(100)  NOT NULL,
        AssuntoEx                       CHAR(100)  NOT NULL,
        ClassificacaoEx                 CHAR(4)    NOT NULL,
        NumRegistroEx                   CHAR(8)    NOT NULL,
        LivreEx                         CHAR(100),
        LocalEx                         CHAR(1)    NOT NULL,
        DataInclusaoEx                  DATE       NOT NULL,
        CodFuncEx                       CHAR(10)   NOT NULL,
        NomeFuncEx                      CHAR(50),
        DataExObra                      DATE       NOT NULL,
        IdiomaEx_1                      CHAR(3)    NOT NULL,
        IdiomaEx_2                      CHAR(3),
        IdiomaEx_3                      CHAR(3));

--  Criação da tabela ExempExcluido (ExempExcluido)
CREATE TABLE ExempExcluido (
        CodExemplarEx                   CHAR(12)   NOT NULL,
        StatusEx                        CHAR(1)    NOT NULL,
```

```
        VolumeEx                        SMALLINT,
        DataCadastroEx                  DATE       NOT NULL,
        DataBaixaEx                     DATE       NOT NULL,
        CodObraEx                       CHAR(10)   NOT NULL,
        CodLocalEx                      CHAR(1)    NOT NULL,
        DataExExemp                     DATE       NOT NULL);

--  Criação da tabela Local (Local)
CREATE TABLE Local (
        CodLocal                        CHAR(1)    NOT NULL,
        NomeLocal                       CHAR(30)   NOT NULL);

--  Criação da tabela Reserva (Reserva)
CREATE TABLE Reserva (
        CodUser                         CHAR(10)   NOT NULL,
        CodObra                         CHAR(10)   NOT NULL,
        CodReserva                      INTEGER    NOT NULL,
        DataReserva                     TIMESTAMP  NOT NULL,
        Atendida                        CHAR(1)    NOT NULL,
        DataAtendimento                 TIMESTAMP);

--  Criação da tabela Emprestimo (Emprestimo)
CREATE TABLE Emprestimo (
        CodUser                         CHAR(10)   NOT NULL,
        CodExemplar                     CHAR(12)   NOT NULL,
        CodEmprestimo                   CHAR(10)   NOT NULL,
        DataRetirada                    DATE       NOT NULL,
        DataEntrega                     DATE       NOT NULL,
        DiasAtraso                      SMALLINT,
        CodFunc                         CHAR(10)   NOT NULL);

--  Criação de chave primária PK_Usuario (PrimaryKey) da tabela Usuario
(Usuario)
ALTER TABLE Usuario ADD CONSTRAINT PK_Usuario PRIMARY KEY(CodUser);

--  Criação de chave primária PK_Obra (PrimaryKey) da tabela Obra (Obra)
ALTER TABLE Obra ADD CONSTRAINT PK_Obra PRIMARY KEY(CodObra);

--  Criação de chave primária PK_Exemplar (PrimaryKey) da tabela Exemplar
(Exemplar)
ALTER TABLE Exemplar ADD CONSTRAINT PK_Exemplar PRIMARY KEY(CodExemplar);

--  Criação de chave primária PK_Funcionario (PrimaryKey) da tabela
Funcionario (Funcionario)
ALTER TABLE Funcionario ADD CONSTRAINT PK_Funcionario PRIMARY KEY(CodFunc);

--  Criação de chave primária PK_Parametros (PrimaryKey) da tabela
Parametros (Parametros)
ALTER TABLE Parametros ADD CONSTRAINT PK_Parametros PRIMARY KEY(NomeInst);

--  Criação de chave primária PK_Historico (PrimaryKey) da tabela Historico
(Historico)
ALTER TABLE Historico ADD CONSTRAINT PK_Historico PRIMARY
KEY(CodUserH,CodExempH);

--  Criação de chave primária PK_ObraExcluida (PrimaryKey) da tabela
ObraExcluida (ObraExcluida)
ALTER TABLE ObraExcluida ADD CONSTRAINT PK_ObraExcluida PRIMARY
KEY(CodObraEx);
```

```
--  Criação de chave primária PK_ExempExcluido (PrimaryKey) da tabela
ExempExcluido (ExempExcluido)
ALTER TABLE ExempExcluido ADD CONSTRAINT PK_ExempExcluido PRIMARY
KEY(CodExemplarExc);

--  Criação de chave primária PK_Local (PrimaryKey) da tabela Local (Local)
ALTER TABLE Local ADD CONSTRAINT PK_Local PRIMARY KEY(CodLocal);

--  Criação de chave primária PK_Reserva (PrimaryKey) da tabela Reserva
(Reserva)
ALTER TABLE Reserva ADD CONSTRAINT PK_Reserva PRIMARY KEY(CodUser,CodObra);

--  Criação de chave primária PK_Emprestimo (PrimaryKey) da tabela
Emprestimo (Emprestimo)
ALTER TABLE Emprestimo ADD CONSTRAINT PK_Emprestimo PRIMARY
KEY(CodUser,CodExemplar);

--  Criação das chaves estrangeiras da tabela Usuario
--  Criação da chave estrangeira FK_Usuario1_Funci
(CE_Usuario_Funcionario_1)
ALTER TABLE Usuario ADD CONSTRAINT FK_Usuario1_Funci FOREIGN KEY(CodFunc)
REFERENCES Funcionario ON DELETE RESTRICT;

--  Criação de validação referente à tabela Usuario(Usuario)
ALTER TABLE Usuario ADD CHECK (Vinculo In (Aluno,Professor,Servidor));

--  Criação de validação referente à tabela Usuario(Usuario)
ALTER TABLE Usuario ADD CHECK (UF In
(AM,AL,AP,AC,BA,CE,DF,ES,GO,MA,MG,MS,MT,PA,PB,PE,PI,PR,RJ,RN,RO,RR,RS,SC,SE
,SP,TO));

--  Criação das chaves estrangeiras da tabela Obra
--  Criação da chave estrangeira FK_Obra1_Funci (CE_Obra_Funcionario_1)
ALTER TABLE Obra ADD CONSTRAINT FK_Obra1_Funci FOREIGN KEY(CodFunc)
REFERENCES Funcionario ON DELETE RESTRICT;

--  Criação de validação referente à tabela Obra(Obra)
ALTER TABLE Obra ADD CHECK (Tipo In (1));

--  Criação de validação referente à tabela Obra(Obra)
ALTER TABLE Obra ADD CHECK (NumPag >= 1);

--  Criação de validação referente à tabela Obra(Obra)
ALTER TABLE Obra ADD CHECK (Idioma_1 In
(POR,ENG,SPA,GER,ITA,LAT,DUT,CHI,JPN,RUS,POL,NOR,SWE,CZE,UKR,KOR,DAN,UMB,NY
A,GAL,GRE,KOK));

--  Criação de validação referente à tabela Obra(Obra)
ALTER TABLE Obra ADD CHECK (Idioma_2 In
(POR,ENG,SPA,GER,ITA,LAT,DUT,CHI,JPN,RUS,POL,NOR,SWE,CZE,UKR,KOR,DAN,UMB,NY
A,GAL,GRE,KOK));

--  Criação de validação referente à tabela Obra(Obra)
ALTER TABLE Obra ADD CHECK (Idioma_3 In
(POR,ENG,SPA,GER,ITA,LAT,DUT,CHI,JPN,RUS,POL,NOR,SWE,CZE,UKR,KOR,DAN,UMB,NY
A,GAL,GRE,KOK));

--  Criação das chaves estrangeiras da tabela Exemplar
--  Criação da chave estrangeira FK_Exemplar3_Local (CE_Exemplar_Local_1)
ALTER TABLE Exemplar ADD CONSTRAINT FK_Exemplar3_Local FOREIGN
KEY(CodLocal) REFERENCES Local ON DELETE RESTRICT;
```

```
--  Criação da chave estrangeira FK_Exemplar2_Obra (CE_Exemplar_Obra_1)
ALTER TABLE Exemplar ADD CONSTRAINT FK_Exemplar2_Obra FOREIGN KEY(CodObra)
REFERENCES Obra ON DELETE RESTRICT;

--  Criação dos índices da tabela Exemplar
--  Criação do índice I_Exemplar_CodExem (CodExemplarIndex)
CREATE  UNIQUE INDEX I_Exemplar_CodExem ON Exemplar(CodExemplar ASC);

--  Criação de validação referente à tabela Exemplar(Exemplar)
ALTER TABLE Exemplar ADD CHECK (Volume >= 1);

--  Criação das chaves estrangeiras da tabela Funcionario
--  Criação da chave estrangeira FK_Funcionario1_Lo
(CE_Funcionario_Local_1)
ALTER TABLE Funcionario ADD CONSTRAINT FK_Funcionario1_Lo FOREIGN
KEY(CodLocal) REFERENCES Local ON DELETE RESTRICT;

--  Criação das chaves estrangeiras da tabela Parametros
--  Criação da chave estrangeira FK_Parametros1_Fun
(CE_Parametros_Funcionario_1)
ALTER TABLE Parametros ADD CONSTRAINT FK_Parametros1_Fun FOREIGN
KEY(CodFunc) REFERENCES Funcionario ON DELETE RESTRICT;

--  Criação das chaves estrangeiras da tabela Reserva
--  Criação da chave estrangeira FK_Reserva1_Obra (CE_Reserva_Obra_1)
ALTER TABLE Reserva ADD CONSTRAINT FK_Reserva1_Obra FOREIGN KEY(CodObra)
REFERENCES Obra ON DELETE CASCADE;

--  Criação da chave estrangeira FK_Reserva2_Usuar (CE_Reserva_Usuario_1)
ALTER TABLE Reserva ADD CONSTRAINT FK_Reserva2_Usuar FOREIGN KEY(CodUser)
REFERENCES Usuario ON DELETE CASCADE;

--  Criação das chaves estrangeiras da tabela Emprestimo
--  Criação da chave estrangeira FK_Emprestimo1_Exe
(CE_Emprestimo_Exemplar_1)
ALTER TABLE Emprestimo ADD CONSTRAINT FK_Emprestimo1_Exe FOREIGN
KEY(CodExemplar) REFERENCES Exemplar ON DELETE CASCADE;

--  Criação da chave estrangeira FK_Emprestimo3_Fun
(CE_Emprestimo_Funcionario_1)
ALTER TABLE Emprestimo ADD CONSTRAINT FK_Emprestimo3_Fun FOREIGN
KEY(CodFunc) REFERENCES Funcionario ON DELETE RESTRICT;

--  Criação da chave estrangeira FK_Emprestimo2_Usu
(CE_Emprestimo_Usuario_1)
ALTER TABLE Emprestimo ADD CONSTRAINT FK_Emprestimo2_Usu FOREIGN
KEY(CodUser) REFERENCES Usuario ON DELETE CASCADE;
```

7.3.3 Manipulação dos dados

Para permitir a pesquisa do usuário foram escritas consultas em linguagem SQL, primeiramente serão apresentadas suas descrições, logo após será listado o script que efetiva a implementação física das mesmas.

7.3.3.1 Descrição das consultas

Nesta seção temos a descrição de todas as consultas que permitem a manipulação da base de dados.

Nome: C_ass_ res
Descrição: Retornará da tabela exemplar todos os exemplares relacionados com o assunto.
Comando:
```
select *, from exemplar
where CodObra in
  (select CodObra, from obras
   where Assunto like %TAssunto%)
```

Nome: C_ass_aut_comp
Descrição: Retornará da tabela obra todas as obras relacionadas com o assunto e o autor informados.
Comando:
```
select *, from obra
where Autor Like %TAutor% and
      Assunto Like %TAssunto%
```

Nome: C_ass_aut_res
Descrição: Retornará da tabela exemplar todos os exemplares relacionados com o assunto e o autor informados.
Comando:
```
select *, from exemplar
where Autor Like %TAutor% and CodObra in
  (select CodObra, from obra
   where Assunto Like %TAssunto%)
```

Nome: C_ass_aut_tit_comp
Descrição: Retornará da tabela obra todas as obras relacionadas com o assunto, o autor e o título informados.
Comando:
```
select *, from obra
where Assunto Like %TAssunto% and
      Autor Like %TAutor% and Titulo Like %TTitulo%
```

Nome: C_ass_aut_tit_res
Descrição: Retornará da tabela exemplar todos os exemplares relacionados com o assunto, o autor e o título informados.

Comando:
```
select *, from exemplar
where Autor Like %TAutor% and Titulo Like %TTitulo%
  and CodObra in
  (select CodObra, from obra
   where Assunto Like %TAssunto%)
```

Nome: C_ass_comp
Descrição: Retornará da tabela obra, todas as obras relacionadas com o assunto.
Comando:
```
select *, from obra
where Assunto Like %TAssunto%
```

Nome: C_ass_res
Descrição: Retornará da tabela exemplar todos os exemplares relacionados com o assunto informado.
Comando:
```
select *, from exemplar
where CodObra in
  (select CodObra, from obra
   where Assunto like %TAssunto%)
```

Nome: C_ass_tit_comp
Descrição: Retornará da tabela obras todos as obras relacionadas com o assunto e o título informados.
Comando:
```
select *, from obra
where Assunto Like %TAssunto% and
      Titulo Like %TTitulo%
```

Nome: C_ass_tit_res
Descrição: Retornará da tabela exemplares todos os exemplares relacionados com o assunto e o título informados
Comando:
```
select *, from exemplar
where Titulo Like %TTitulo% and CodObra in
  (select CodObra, from obra
   where Assunto Like %TAssunto%)
```

Nome: C_aut_comp
Descrição: Retornará da tabela obra, todas as obras relacionadas com o autor.
Comando:
```
select *, from obra
where Autor Like %TAutor%
```

Nome: C_aut_res
Descrição: Retornará da tabela exemplares todos os exemplares relacionados com o autor.
Comando:
```
select *, from exemplar
where Autor Like %TAutor%
```

Nome: C_aut_tit_comp
Descrição: Retornará da tabela obra todas as obras relacionadas com o autor e o título informados.
Comando:
```
select *, from obra
where Autor Like %TAutor% and
      Titulo Like %TTitulo%
```

Nome: C_aut_tit_res
Descrição: Retornará da tabela exemplar todos os exemplares relacionados com autor e título informados.
Comando:
```
select *, from exemplar
where Autor Like %TAutor% and
      Titulo Like %TTitulo%
```

Nome: C_idi_comp
Descrição: Retornará da tabela obra todas as obras relacionadas com o idioma informado.

Comando: select *, from obra
where Idioma = "Tldioma")

Nome: C_idi_res
Descrição: Retornará da tabela exemplar todos os exemplares relacionadas o idioma informado.
Comando: select *, from exemplar
where CodObra in
(select CodObra, from obra
where Idioma = "TIdioma")

Nome: C_loc_comp
Descrição: Retornará da tabela obra todas as obras relacionadas com o local informado.
Comando: select *, from obra
where CodObra in
(select CodObra, from exemplar
where Tipo Like %TTipo%)

Nome: C_loc_res
Descrição: Retornará da tabela exemplar todas os exemplares relacionadas com o local informado.
Comando: select *, from exemplar
where Local Like %"TLocal%

Nome: C_tipo_comp
Descrição: Retornará da tabela obra todas as obras relacionadas com o tipo de material informado.
Comando: select *, from obra
where Tipo Like %TTipo%

Nome: C_tipo_res
Descrição: Retornará da tabela exemplar todos os exemplares relacionados com o tipo de material informado.
Comando: select *, from exemplar
where CodObra in
(select CodObra, from obra
where Tipo Like %TTipo%)

Nome: C_tit_comp
Descrição: Retornará da tabela obra todas as obras relacionadas com o título.
Comando: select *, from obra
where Titulo Like %TTitulo%

Nome: C_tit_res
Descrição: Retornará da tabela exemplar todos os exemplares relacionados com o título.
Comando: select *, from exemplar
where Titulo Like %TTitulo%

7.3.3.4 Implementação das consultas

A seguir é listado o script que possibilita a implementação das consultas descritas na seção anterior.

```
--  Criação do Banco de Dados.
CREATE DATABASE SB;

--  Conexão ao banco de dados.
CONNECT TO SB;

--  Exclusão da consulta C_ass__res (C_ass_ res)
DROP VIEW C_ass__res;

--  Exclusão da consulta C_ass_aut_comp (C_ass_aut_comp)
DROP VIEW C_ass_aut_comp;

--  Exclusão da consulta C_ass_aut_res (C_ass_aut_res)
DROP VIEW C_ass_aut_res;

--  Exclusão da consulta C_ass_aut_tit_comp (C_ass_aut_tit_comp)
DROP VIEW C_ass_aut_tit_comp;

--  Exclusão da consulta C_ass_aut_tit_res (C_ass_aut_tit_res)
DROP VIEW C_ass_aut_tit_res;

--  Exclusão da consulta C_ass_comp (C_ass_comp)
DROP VIEW C_ass_comp;

--  Exclusão da consulta C_ass_res (C_ass_res)
DROP VIEW C_ass_res;

--  Exclusão da consulta C_ass_tit_comp (C_ass_tit_comp)
DROP VIEW C_ass_tit_comp;

--  Exclusão da consulta C_ass_tit_res (C_ass_tit_res)
DROP VIEW C_ass_tit_res;

--  Exclusão da consulta C_aut_comp (C_aut_comp)
DROP VIEW C_aut_comp;

--  Exclusão da consulta C_aut_res (C_aut_res)
DROP VIEW C_aut_res;

--  Exclusão da consulta C_aut_tit_comp (C_aut_tit_comp)
DROP VIEW C_aut_tit_comp;

--  Exclusão da consulta C_aut_tit_res (C_aut_tit_res)
DROP VIEW C_aut_tit_res;

--  Exclusão da consulta C_idi_comp (C_idi_comp)
DROP VIEW C_idi_comp;

--  Exclusão da consulta C_idi_res (C_idi_res)
DROP VIEW C_idi_res;

--  Exclusão da consulta C_loc_comp (C_loc_comp)
DROP VIEW C_loc_comp;
```

```
--  Exclusão da consulta C_loc_res (C_loc_res)
DROP VIEW C_loc_res;

--  Exclusão da consulta C_tipo_comp (C_tipo_comp)
DROP VIEW C_tipo_comp;

--  Exclusão da consulta C_tipo_res (C_tipo_res)
DROP VIEW C_tipo_res;

--  Exclusão da consulta C_tit_comp (C_tit_comp)
DROP VIEW C_tit_comp;

--  Exclusão da consulta C_tit_res (C_tit_res)
DROP VIEW C_tit_res;

--  Criação da consulta C_ass__res (C_ass_ res)
CREATE VIEW C_ass__res AS select *, from exemplar
where CodObra in
  (select CodObra, from obras
   where Assunto like %TAssunto%);;

--  Criação da consulta C_ass_aut_comp (C_ass_aut_comp)
CREATE VIEW C_ass_aut_comp AS select *, from obra
where Autor Like %TAutor% and
         Assunto Like %TAssunto%;;

--  Criação da consulta C_ass_aut_res (C_ass_aut_res)
CREATE VIEW C_ass_aut_res AS select *, from exemplar
where Autor Like %TAutor% and CodObra in
  (select CodObra, from obra
   where Assunto Like %TAssunto%);;

--  Criação da consulta C_ass_aut_tit_comp (C_ass_aut_tit_comp)
CREATE VIEW C_ass_aut_tit_comp AS select *, from obra
where Assunto Like %TAssunto% and
         Autor Like %TAutor% and Titulo Like %TTitulo%;;

--  Criação da consulta C_ass_aut_tit_res (C_ass_aut_tit_res)
CREATE VIEW C_ass_aut_tit_res AS select *, from exemplar
where Autor Like %TAutor% and Titulo Like %TTitulo%
  and CodObra in
  (select CodObra, from obra
   where Assunto Like %TAssunto%);;

--  Criação da consulta C_ass_comp (C_ass_comp)
CREATE VIEW C_ass_comp AS select *, from obra
where Assunto Like %TAssunto%;;

--  Criação da consulta C_ass_res (C_ass_res)
CREATE VIEW C_ass_res AS select *, from exemplar
where CodObra in
  (select CodObra, from obra
   where Assunto like %TAssunto%);;

--  Criação da consulta C_ass_tit_comp (C_ass_tit_comp)
CREATE VIEW C_ass_tit_comp AS select *, from obra
where Assunto Like %TAssunto% and
         Titulo Like %TTitulo%;;

--  Criação da consulta C_ass_tit_res (C_ass_tit_res)
```

```
CREATE VIEW C_ass_tit_res AS select *, from exemplar
where Titulo Like %TTitulo% and CodObra in
  (select CodObra, from obra
   where Assunto Like %TAssunto%);;

--  Criação da consulta C_aut_comp (C_aut_comp)
CREATE VIEW C_aut_comp AS select *, from obra
where Autor Like %TAutor%;;

--  Criação da consulta C_aut_res (C_aut_res)
CREATE VIEW C_aut_res AS select *, from exemplar
where Autor Like %TAutor%;;

--  Criação da consulta C_aut_tit_comp (C_aut_tit_comp)
CREATE VIEW C_aut_tit_comp AS select *, from obra
where Autor Like %TAutor% and
         Titulo Like %TTitulo%;;

--  Criação da consulta C_aut_tit_res (C_aut_tit_res)
CREATE VIEW C_aut_tit_res AS select *, from exemplar
where Autor Like %TAutor% and
         Titulo Like %TTitulo%;;

--  Criação da consulta C_idi_comp (C_idi_comp)
CREATE VIEW C_idi_comp AS select *, from obra
where Idioma = "Tldioma");;

--  Criação da consulta C_idi_res (C_idi_res)
CREATE VIEW C_idi_res AS select *, from exemplar
where CodObra in
  (select CodObra, from obra
   where Idioma = "TIdioma");;

--  Criação da consulta C_loc_comp (C_loc_comp)
CREATE VIEW C_loc_comp AS select *, from obra
where CodObra in
  (select CodObra, from exemplar
   where Tipo Like %TTipo%);;

--  Criação da consulta C_loc_res (C_loc_res)
CREATE VIEW C_loc_res AS select *, from exemplar
where Local Like %"TLocal%;;

--  Criação da consulta C_tipo_comp (C_tipo_comp)
CREATE VIEW C_tipo_comp AS select *, from obra
where Tipo Like %TTipo%;;

--  Criação da consulta C_tipo_res (C_tipo_res)
CREATE VIEW C_tipo_res AS select *, from exemplar
where CodObra in
  (select CodObra, from obra
   where Tipo Like %TTipo%);;

--  Criação da consulta C_tit_comp (C_tit_comp)
CREATE VIEW C_tit_comp AS select *, from obra
where Titulo Like %TTitulo%;;

--  Criação da consulta C_tit_res (C_tit_res)
CREATE VIEW C_tit_res AS select *, from exemplar
where Titulo Like %TTitulo%;;
```

8. CONSIDERAÇÕES FINAIS

O trabalho realizado apresentou uma alternativa de sistema para solucionar os problemas existentes no sistema da Biblioteca Setorial de Ciência e Tecnologia, mas especificamente em sua base de dados.

O esquema de banco de dados, definido com base na abordagem relacional, permitirá a plena adaptação da base de dados a qualquer uma das plataformas relacionais disponíveis no mercado. Além disso, a utilização o ferramenta CASE Dr.Case 3.0, deixa o modelo muito mais confiável pois evita anomalias e redundâncias dentro do esquema de banco de dados.

Como continuação do trabalho, sugere-se um estudo para viabilizar a total integração entre o sistema de bibliotecas e o sistema acadêmico, o que não ocorre nos dias atuais.

9. REFERÊNCIAS BIBLIOGRÁFICAS

[KOR99] – KORTH, H. F. – “Sistemas de banco de dados” – MAKRON BOOKS, São Paulo: 1999

[DAT99] – DATE, C.J., - “Introdução a Sistemas de Banco de Dados” – tradução da 6ª. Ed. Americana – Campus, Rio de Janeiro: 1999

[MAC96] – MACHADO, F. R. – “Projeto de Banco de Dados: uma visão prática” – Érica, São Paulo: 1996