

Estudos de Caso de Projetos de Bancos de Dados

[Desenvolvimento]

É sabido que o objetivo de um projeto de banco de dados é obter um conjunto de esquemas de relações que nos permita armazenar dados sem redundância e que as informações necessárias para tomadas de decisões possam ser geradas facilmente. Assim, para que um projeto de banco de dados possa atender a estes pressupostos, aplicamos o que podemos chamar de normalização dos dados.

Na edição número 47 desta revista, apresentamos os conceitos envolvidos na normalização de um projeto de banco de dados.

Neste artigo, utilizaremos os conceitos citados na referida edição e abordaremos o projeto de um banco de dados para uma biblioteca e para um sistema de ordens de serviço, com algumas restrições de escopo, que detalharemos a seguir.

Escolhemos estes projetos devido ao fato das regras de negócio serem de fácil compreensão. Entretanto, eles possuem vários pontos de questionamento

por parte dos projetistas de bancos de dados. Uma grande parcela destes profissionais tende a cometer grandes erros nos mais diversos projetos de bancos de dados e nestes dois exemplos, conseguiremos abordar os pontos que devem ser considerados, desde projetos de pequeno porte, até grandes projetos de bancos de dados.

Assim, inicialmente apresentaremos o estudo de caso para a biblioteca e posteriormente o de ordens de serviço.

Biblioteca

Todos nós já freqüentamos bibliotecas em algum momento de nossa vida acadêmica. Em linhas gerais, uma biblioteca possui livros que devem ser emprestados e devolvidos. Neste estudo de caso em particular, reduziremos o escopo de uma biblioteca com a finalidade de facilitar a compreensão do modelo de dados que apresentaremos. Entretanto, nada impede que o leitor acrescente outras funcionalidades no modelo aqui apresentado.



Ary Júnior
(ary@ajsolucoes.com.br)

é Mestre em Ciências com ênfase em Inteligência Artificial e Bancos de Dados, Professor Universitário e Gerente de TI. Atua como professor desde 2000, especialmente com disciplinas na área de Bancos de Dados. Na área de TI desde 1998 com desenvolvimento, análise e projeto de Sistemas. Hoje em dia, trabalha com Gerenciamento de Projetos e Sistemas de Bancos de Dados prestando serviços para diversas empresas nacionais e multinacionais.

Assim, um sistema de bibliotecas, para este estudo de caso, consiste das seguintes funcionalidades:

- Cadastro de Livros: Um cadastro de livro deve armazenar informações relativas ao Título do livro, Editora, Edição, Ano de Publicação, Autores, Assunto. Sabe-se também que um livro pode possuir vários exemplares e neste caso, são emprestados os exemplares e não os títulos.
- Cadastro de Alunos: Um cadastro de aluno deve possuir número de matrícula, nome, endereço, telefone, telefone celular, CPF, RG, e-mail.
- Cadastro de Professores: Um cadastro de professor deve possuir nome, endereço, telefone, telefone celular, CPF, RG, e-mail, titulação.
- Empréstimo de Livros: Esta funcionalidade refere-se ao empréstimo de livros propriamente dita aos alunos ou professores da instituição. Este empréstimo deve armazenar a data do empréstimo, data prevista de devolução, exemplar emprestado.
- Devolução de Livros: Após serem emprestados, os livros podem ser devolvidos. Neste ponto, os livros podem ter devoluções parciais, ou seja, um aluno pode pegar emprestado 3 livros diferentes e querer devolver apenas um.

Desta forma, neste artigo não abordaremos o pagamento de multas, assim como o cadastro de periódicos que também existem em bibliotecas, pelo fato de entendermos que fogem do objetivo deste exemplo. Também não será tratado o empréstimo de livros a funcionários

com regras distintas, pelo mesmo motivo. Em diversas instituições existem critérios distintos de empréstimos para professores, alunos e funcionários e neste artigo também não serão tratadas estas diferenciações de regras.

Também não serão abordadas outras funcionalidades como empréstimo de exemplares únicos, empréstimo de férias, reserva e renovação de livros.

A partir desta pequena definição das regras de negócio em questão, iremos iniciar a modelagem de dados. Não formalizamos esta definição das regras de negócio também por fugir dos objetivos deste artigo.

Na **Figura 1** apresentamos a primeira parte do modelo de dados, que comprehende o cadastro de livros.

Em posse da Tabela Livro (**Figura 1**), precisaremos verificar se ela está normalizada. O primeiro passo é verificar se ela está na 1FN (Primeira Forma Normal). A 1FN diz que “Uma tabela se encontra na 1FN se todos os atributos possuírem apenas valores atômicos (simples e indivisíveis) e os valores de cada atributo no registro também deve ser um valor simples (ou seja, o atributo não é composto)”.

Podemos perceber que esta tabela possui os atributos Autores e Assunto, que são multivalorados, ou seja, para cada livro, podemos ter mais de um autor e mais de um assunto. Assim, devemos separar estes atributos em outras tabelas. Assim, na **Figura 2** temos o modelo de dados normalizado segundo a 1 FN.

Podemos perceber que no modelo de dados da **Figura 2** foram inseridas as

tabelas Assunto, Autor, Livro_Assunto e Livro_Autor. A tabela Autor armazena o cadastro básico dos autores, assim como a tabela assunto armazena os mais diversos assuntos de livros. A tabela Livro_Autor é responsável por resolver o relacionamento n:m entre livro e autor, ou seja, um livro pode ter vários autores e cada autor pode ter vários livros publicados. Da mesma forma, a tabela Livro_Assunto.

O próximo passo é verificar se o modelo de dados da **Figura 2** se encontra na 2FN (Segunda Forma Normal). A 2FN diz que “Uma tabela se encontra na 2FN se estiver na 1FN e não possuir dependência funcional parcial. Caso existam atributos que não dependam integralmente da chave primária, devemos retirar da tabela todos eles e dar origem a uma nova tabela.” Entretanto, na **Figura 2** percebemos que apenas as tabelas Livro_Autor e Livro_Assunto possuem chaves primárias compostas. Além disso, ambas possuem apenas os atributos que compõem a chave. Logo, elas estão na 2FN.

Agora, iremos verificar se o modelo de dados da **Figura 2** encontra-se na 3FN (Terceira Forma Normal). A 3FN diz que “Uma tabela está na 3FN se estiver na 2FN e não possuir nenhuma dependência funcional transitiva”. Isto quer dizer que nenhum atributo pode depender de outro atributo que não seja a chave primária. Assim, percebemos que na tabela Livro da **Figura 2** existe o atributo Editora que não depende de CodL, que é chave primária. Desta forma, precisaremos criar outra tabela para armazenar o cadastro de Editoras.

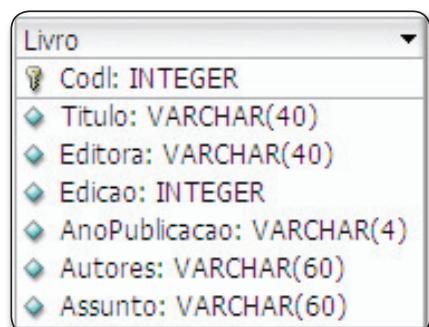


Figura 1. Tabela Livro Desnormalizada

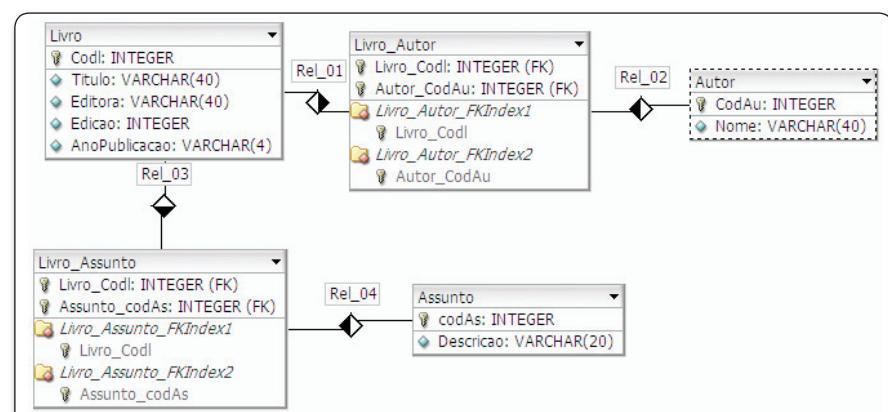


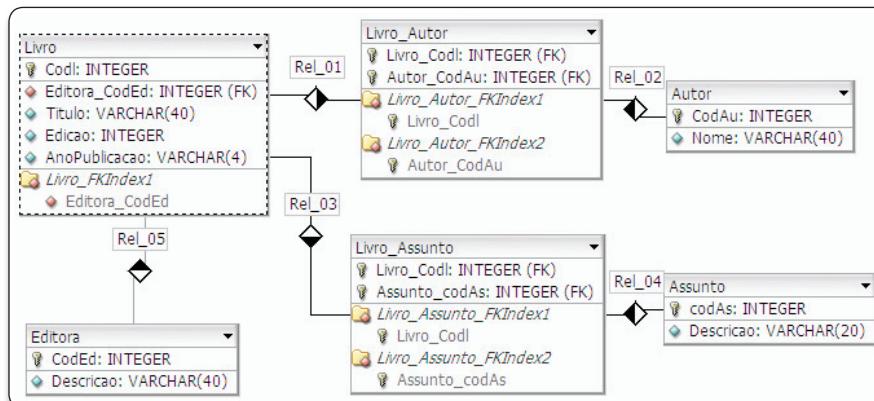
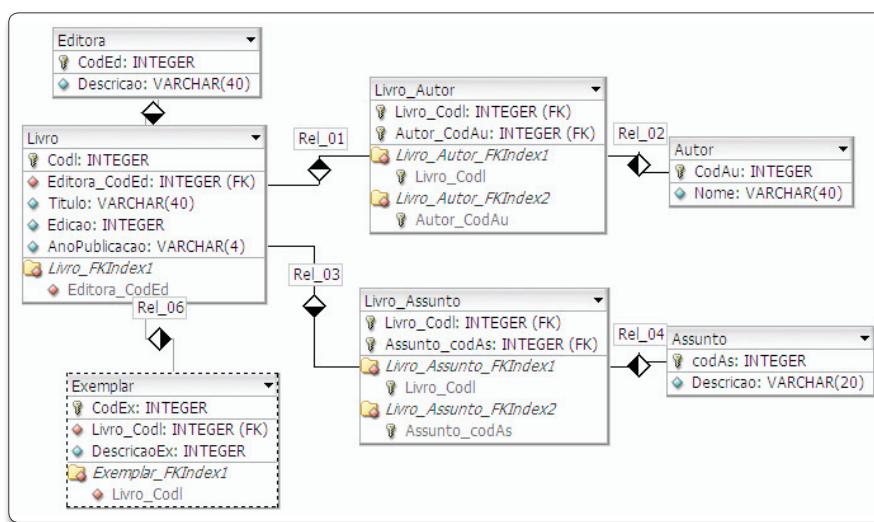
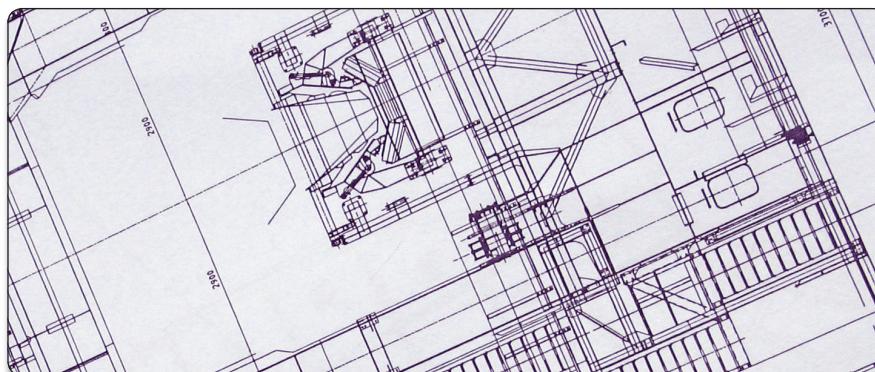
Figura 2. Modelo de Dados Livro – Assunto – Autores

Listagem 1. Consulta quantidade de exemplares por livro

```
SELECT titulo, count (codex)
FROM Livro, Exemplar
WHERE livro.codl = exemplar.Livro_codl
GROUP BY titulo
```

Listagem 2. Consulta quantidade de exemplares por livro

```
SELECT titulo, qdade_exemplar
FROM Livro
```

**Figura 3.** Modelo de Dados Livro – Assunto – Autores - Editora**Figura 4.** Modelo de Dados Livro – Assunto – Autores – Editora - Exemplar

Podemos perceber que o modelo de dados da **Figura 3** encontra-se na 3FN. Conforme abordamos no artigo “Normalização de Dados”, desta revista, em sua edição de número 47, podemos abrir mão das outras formas normais: BCNF (Forma Normal de Boyce Codd), na 4FN (Quarta Forma Normal) e na 5FN (Quinta Forma Normal). Isto pode ocorrer sem prejuízo para este modelo de dados, pois não verificamos dependências funcionais cíclicas ou multivaloradas.

Entretanto, precisamos resolver mais um item relacionado ao cadastro de livros definido no início deste tópico que é a existência de exemplares de livros. Para isto, basta criar uma tabela exemplar conforme a **Figura 4**.

Assim, o modelo de dados de livro fica completo. Neste caso, vale ressaltar um ponto com relação às informações que podem ser geradas a partir deste modelo de dados. Podemos extrair diversas informações, dentre as quais podemos citar:

- Relação de livros cadastrados;
- Relação de editoras cadastradas;
- Relação de autores cadastrados;
- Relação de assuntos cadastrados;
- Relação de livros com seus respectivos autores;
- Relação de livros com seus respectivos assuntos;
- Quantidade de exemplares por livro.

Repare que nesta ultima informação, precisaremos efetuar um cálculo a cada consulta efetuada no banco de dados. Obviamente que um cálculo simples como este pode ser bastante rápido (**Listagem 1**).

Porém, podemos abrir mão da normalização de dados e armazenar um atributo quantidade de exemplares (qdade_exemplar) na tabela livro que seja responsável por armazenar a quantidade de exemplares para o referido título. Entretanto, a definição pela utilização deste artifício deve ser exclusivamente do leitor. Assim, o modelo de dados contendo este atributo está representado na **Figura 5** e a consulta SQL na **Listagem 2**. Perceba que a consulta da **Listagem 2** é muito mais simples que a consulta da **Listagem 1** e por consequência, é muito mais rápida.

Repare no modelo de dados da **Figura 5**. Caso queiramos cadastrar outra edição de um determinado livro, deveremos cadastrar outro livro com o mesmo título. Isto é uma redundância que poderíamos ter evitado. Neste artigo não iremos efetuar esta mudança. Porém, o leitor facilmente poderia fazê-la, simplesmente criando outra tabela para armazenar o título e a editora, relacionando-os com livro. Isto retiraria a redundância e permitiria tal possibilidade.

Feita a modelagem de dados para o cadastro de livros, podemos partir para o cadastro de alunos e professores. Conforme definição no início deste tópico, tanto o cadastro de alunos quanto o cadastro de professores possuem os atributos, nome, endereço, telefone, celular, CPF, RG, e-mail. Entretanto, o aluno possui número de matrícula e o professor possui titulação. Este é o típico caso de relacionamentos generalização-especialização. Assim, podemos ter uma tabela pessoa responsável por armazenar os registros das pessoas, independente se são alunos, professores ou ambos. E desta tabela pessoa, especializamos em duas novas tabelas, que são: alunos e professores.

Na **Figura 6**, apresentamos o modelo de dados de Pessoa. Da mesma forma como trabalhamos no modelo de dados de livros (**Figura 1**), precisaremos normalizar este modelo de dados. Assim, inicialmente verificaremos se ele está na 1FN. Repare que os atributos, endereço, telefone, celular e e-mail podem ser multivalorados. Além disso, o atributo endereço é composto. Dizemos que endereço é multivalorado porque podemos armazenar vários endereços para uma mesma pessoa, como, endereço comercial, residencial, dentre outros. Assim, na **Figura 7** temos o modelo de dados da **Figura 6** de acordo com a 1FN.

Podemos perceber que o modelo de dados da **Figura 7** encontra-se na 1FN pois não possui atributos multivalorados e nem atributos compostos. Um grande ponto que podemos discutir é que neste modelo normalizamos também as tabelas TipoLogradouro, Logradouro, Bairro, Cidade e UF. Repare que em Tipo de Logradouro armazenaremos os

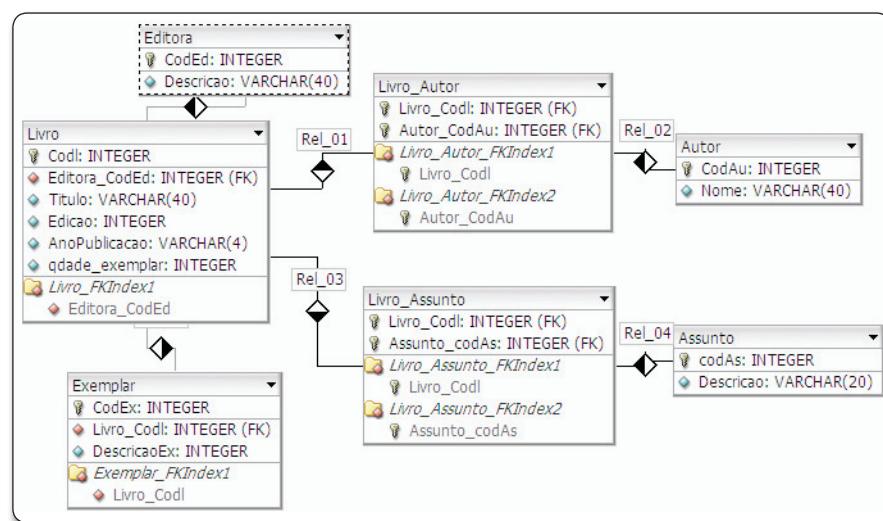


Figura 5. Modelo de Dados Livro – Assunto – Autores – Editora – Exemplar, com quantidade de exemplares

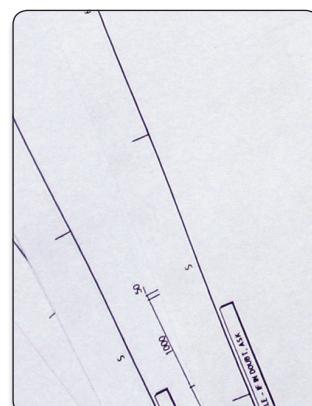
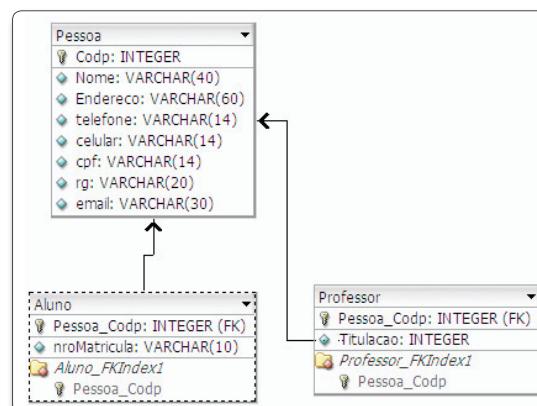


Figura 6. Modelo de Dados Livro Pessoa

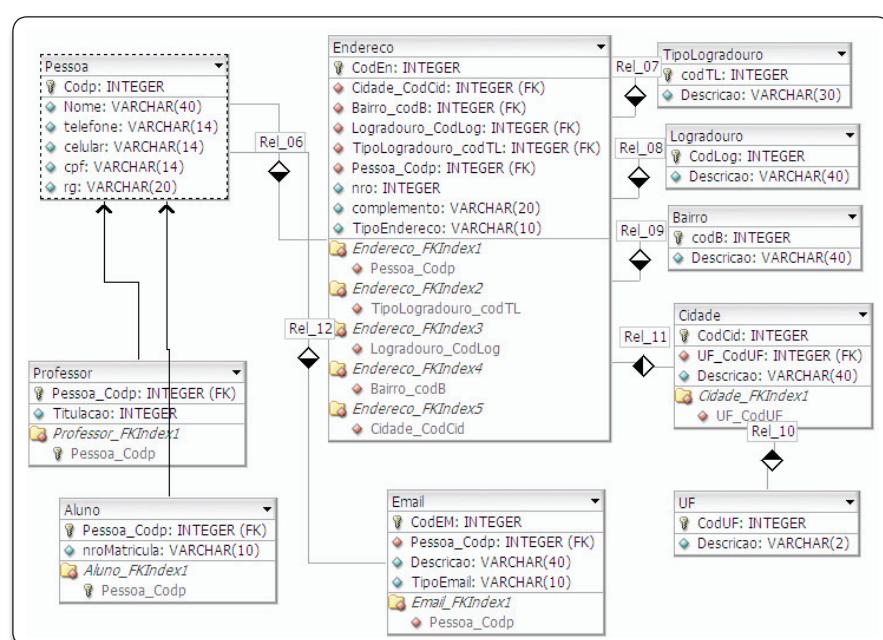


Figura 7. Modelo de Dados Livro Pessoa 1FN

mais diversos tipos de logradouros que existem, tais como rua, avenida, praça, alameda, dentre outros. Em Logradouro armazenaremos a descrição do endereço, como, Getúlio Vargas, que pode ser uma rua, avenida, dentre outras opções, dependendo da cidade em questão. Os atributos Bairro, cidade e UF também devem ser tratados da mesma forma. Entretanto, decidimos não relacionar bairro com cidade, e bairro com logradouro, para não ser necessário mapear todos os endereços brasileiros. Obviamente que isto pode gerar inconsistência nos dados, como, uma pessoa morar em uma avenida chamada Brasil e outra morar em uma praça Brasil, na mesma cidade, caso o usuário do sistema efetue um cadastro errado. Porém, deixamos esta discussão de lado e o leitor tem total liberdade para avaliar o seu caso para decidir como relacionar tais tabelas.

Neste caso, fizemos desta maneira, pois estamos interessados, por exemplo, em saber:

- Quantidade de pessoas que residem em avenidas;
- Quantidade de pessoas que residem em determinado logradouro, bairro, cidade ou estado.

Outro ponto que podemos perceber é que realmente uma pessoa pode ter vários e-mails e cada e-mail é de apenas uma pessoa. Na tabela e-mail, adicionamos um atributo TipoEmail para separarmos o e-mail em pessoal, profissional,

dentre outros tipos possíveis. Da mesma forma, criamos o atributo TipoEndereco na tabela endereço para sabermos se o endereço é residencial, comercial, dentre outros tipos.

Repare também que neste contexto continuamos armazenando um telefone e um celular. Caso seja necessário armazenar mais valores como no caso do endereço e do e-mail, deve-se ter uma tabela para telefones relacionada com pessoa. Neste caso, sugerimos que se adicione também um atributo tipo de telefone para informar se o telefone é um telefone celular ou um telefone fixo. Veja que desta forma, a tabela pessoa não está totalmente na 1FN uma vez que estes atributos são multivalueados. Entretanto, no contexto deste artigo optamos por manter desta forma, para exemplificar a desnormalização desta tabela, informando que dependendo de cada caso podemos optar pela não normalização de algumas tabelas em detrimento dos contextos das aplicações a serem desenvolvidas.

O próximo passo é verificar se o modelo de dados da **Figura 7** se encontra na 2FN. Verificamos que sim (exceção se faz na tabela pessoa, pelos motivos citados anteriormente aos atributos telefone e celular), uma vez que este modelo não possui tabela com chave primária composta.

Assim, iremos verificar a aplicação da 3FN no modelo da **Figura 7**. Também verificamos tal aplicação (exceção se faz

na tabela pessoa, pelos motivos citados anteriormente aos atributos telefone e celular), uma vez que nenhum atributo de nenhuma tabela depende de outro atributo que não seja a chave primária da referida tabela.

Desta forma, podemos partir agora para os empréstimos e devoluções de livros. Para efetuar tais ações, precisaremos unir os modelos de dados das **Figuras 5 e 7** com os empréstimos e com as devoluções.

Para tratarmos os empréstimos temos duas hipóteses. Ou adotaremos os empréstimos de cada livro por pessoa individualmente ou por itens de empréstimo. Ou seja, cada pessoa pode pegar emprestado vários livros de uma única vez. Assim, cada livro pode ser tratado como um empréstimo isolado ou todos serem tratados em um único empréstimo. Neste artigo iremos abordar como empréstimos isolados por simples definição de escopo deste projeto de banco de dados.

Assim, a **Figura 8** apresenta o modelo de dados com os empréstimos e com as devoluções. Devemos lembrar que estamos utilizando a tabela Exemplar da **Figura 5** e Pessoa da **Figura 7**. O modelo de dados completo é apresentado na **Figura 9**.

Repare que possibilitamos empréstimo de exemplares ao invés de livros. Isto pode nos dar segurança ao efetuar empréstimos, pois conseguiremos saber quais exemplares foram emprestados para quais pessoas. Obviamente que

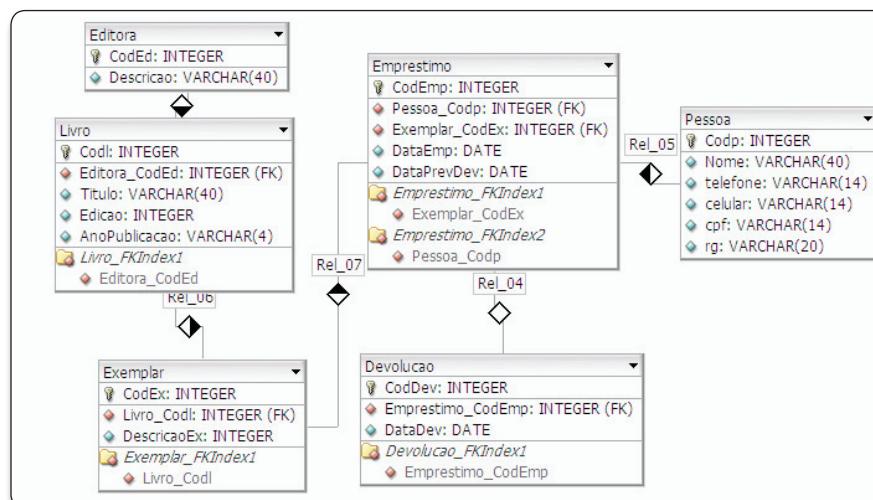


Figura 8. Modelo de Dados Empréstimo



sabendo desta informação, conseguiremos descobrir quais os títulos dos livros que foram emprestados. Assim, caso um exemplar se danifique, por exemplo, é possível efetuar uma auditoria. Entretanto, caso o leitor não queira trabalhar com tal hipótese, pode-se ter um atributo em livros, para dizer qual a quantidade de exemplares do referido livro, conforme abordamos anteriormente. Desta forma, ao emprestar um livro, o sistema precisaria ter uma regra para verificar se todos os exemplares estão emprestados, uma vez que cada livro possui uma quantidade finita de exemplares e não podemos emprestar um livro que não esteja na biblioteca.

Outro ponto que abordamos neste modelo de dados é o empréstimo para pessoas. Isto porque no escopo deste artigo, conforme abordado anteriormente, são tratados os empréstimos e devoluções tanto para alunos quanto para professores sem distinções. Assim, como os dois são especializações da generalização pessoa, podemos realizar o empréstimo diretamente de pessoa.

Perceba também que para os empréstimos estamos armazenando apenas data do empréstimo e data prevista de devolução. Poderíamos armazenar outros atributos. Porém, o leitor pode ficar livre para adicioná-los caso tenha interesse, assim como na tabela devoluções. Também conforme tratamos no início deste artigo, não abordamos o pagamento de multas caso a devolução seja fora do prazo determinado.

Percebemos também que o modelo de dados da **Figura 8** encontra-se na 3FN. Exceção se faz na tabela pessoa, pelos motivos citados anteriormente aos atributos telefone e celular.

Na **Figura 9** apresentamos o modelo de dados final aplicando a 1FN, 2FN e 3FN. Conforme falamos anteriormente, a tabela pessoa está desnormalizada, com relação aos atributos telefone e celular, visando apresentar esta possibilidade aos leitores. Fique à vontade para utilizar e modificar tal modelo de acordo com as suas necessidades.

Perceba também que poderíamos ter tratado os autores como uma especialização de pessoa. Porém, no contexto

deste artigo, optamos por não fazer tal especialização, uma vez que simplificamos o cadastro de autores para armazenar apenas o nome deles. E caso esta tabela tivesse sido tratada como uma especialização de pessoa, precisaríamos possuir os dados de telefone, celular, CPF e RG para que se efetuasse o cadastramento. Obviamente que dizemos isto porque não gostaríamos de ter uma tabela com vários atributos nulos cadastrados. Assim, optamos por manter uma tabela separada para os autores.

Dessa forma, finalizamos este estudo de caso apresentando um modelo de dados para empréstimos e devoluções de livros, conforme **Figura 9**.

Ordens de Serviço

Todos nós já freqüentamos alguma empresa que necessita tratar ordens de serviço, como oficinas mecânicas ou empresas de manutenção de microcomputadores, por exemplo. Em todos estes casos são necessários alguns produtos para a execução dos serviços. Por exemplo, levamos nosso computador para manutenção e o técnico faz o levantamento do problema, dizendo que o problema é na fonte de alimentação. Para execução

deste serviço, a empresa cobra o serviço de substituição da peça com problema e também o valor da peça. Assim, uma ordem de serviço pode possuir produtos e serviços. Mas, para este estudo de caso, optamos por reduzir o escopo de um sistema de ordens de serviço visando facilitar a compreensão por parte do leitor. Entretanto, nada impede que o leitor acrescente outras funcionalidades no modelo aqui apresentado.

Assim, um sistema de ordens de serviço, para este estudo de caso, consiste das seguintes funcionalidades:

- Cadastro de Produtos e Serviços: Um cadastro de produtos precisa ter o nome do produto, uma descrição do produto e o preço do produto e o cadastro de serviços precisa de nome do serviço, descrição do serviço e preço do serviço.
 - Cadastro de Orçamento: Um cadastro de orçamento consiste de Cliente solicitante, data do orçamento, produtos e serviços que compõem o orçamento, valor do orçamento.
 - Cadastro de Ordem de Serviço: Um cadastro de ordem de serviço consiste da data da aprovação do orçamento, pois a ordem de serviço é gerada após a aprovação do orçamento, data de início dos

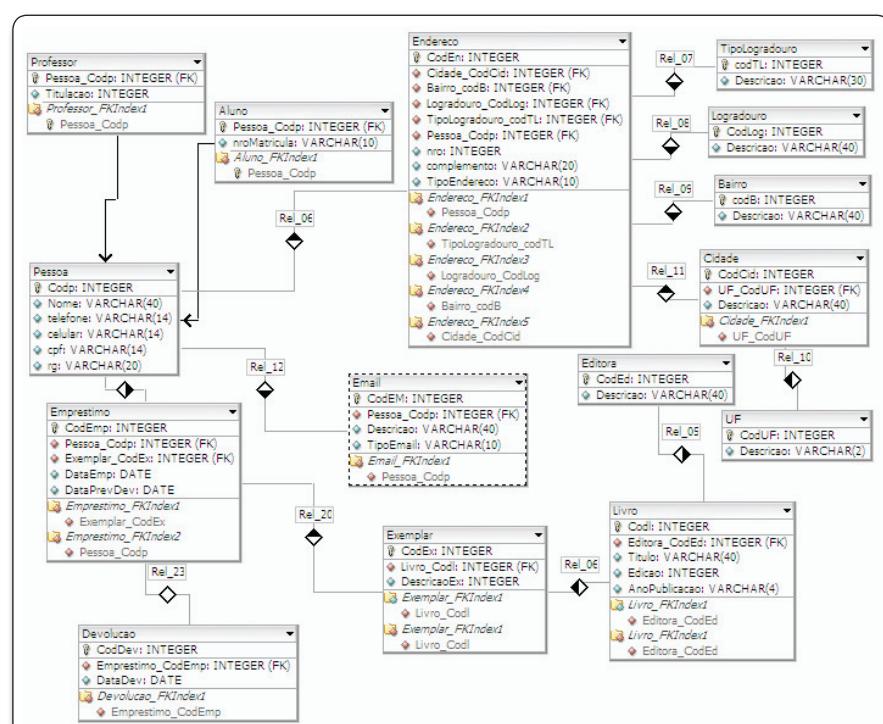


Figura 9. Modelo de Dados Biblioteca Completo

serviços e data prevista para conclusão.

• Cadastro de Clientes: Um cadastro de clientes é composto por nome do cliente, endereço do cliente, telefone do cliente, e-mail do cliente, CPF do cliente, RG do cliente, caso seja pessoa física e CNPJ, Inscrição Estadual caso seja pessoa jurídica.

Assim, para este estudo de caso, inicialmente é gerado um orçamento para o cliente, que pode conter produtos e/ou serviços. Após a aprovação deste orçamento é que serão registradas as ordens de serviço (OS).

Desta forma, neste artigo **não** abordaremos algumas funcionalidades que este modelo de dados poderia ter, tais como:

- Cadastro de fornecedores de produtos;
- Cadastro de fabricantes de produtos;
- Controle de Estoque dos produtos;
- Condições de pagamento;
- Pagamentos e Recebimentos;
- Controles de Cheque.

Mas, apesar destas restrições de escopo, o modelo de dados que será apresentado no final deste estudo de caso com certeza poderá ser útil não só para visualizarmos o projeto do banco de

dados, mas também para utilização em projetos futuros do leitor.

Assim, a partir desta pequena definição das regras de negócio em questão, iremos iniciar a modelagem de dados. Também não formalizamos esta definição das regras de negócio por fugir dos objetivos deste artigo.

Na **Figura 10** apresentamos a primeira parte do modelo de dados, que comprehende o cadastro de clientes. Neste estudo de caso iremos apresentar diretamente a tabela de cliente com os relacionamentos necessários para a normalização, uma vez que este caso já foi definido anteriormente de forma semelhante e apresentado na **Figura 7**.

Na **Figura 10**, conforme exposto anteriormente, são apresentados os relacionamentos envolvendo a tabela clientes, normalizado segundo a 3FN de forma semelhante à da **Figura 7**. Entretanto, neste modelo adicionamos os relacionamentos entre cliente e as tabelas física, jurídica e telefone. A tabela física e a jurídica foram adicionadas porque na definição das regras existem atributos que devem ser preenchidos quando se tratar de uma pessoa física (CPF e RG) e

e atributos que devem ser preenchidos quando se tratar de pessoa jurídica (CNPJ e Inscrição Estadual). Além disso, a tabela telefone foi adicionada para podermos cadastrar vários telefones para um mesmo cliente. O atributo TipoTel desta tabela será responsável por informar se o referido telefone é um telefone residencial, comercial ou celular.

Após finalizar o modelo para cadastro de clientes, podemos avançar no modelo de dados, tratando agora o cadastro de produtos e serviços. Conforme a definição das regras de negócio deste estudo de caso, tanto produtos quanto serviços possuem os mesmos atributos, que são: nome, descrição e preço. Para criarmos as tabelas, temos duas opções. A primeira delas seria criar apenas uma tabela com estes atributos e podermos inserir um atributo com o nome definição, por exemplo, para informar se o registro depois de cadastrado trata-se de um produto ou de um serviço, recebendo as letras P e S respectivamente, por exemplo, conforme **Figura 11**.

Outra opção seria criarmos um relacionamento de generalização-especialização. Com este relacionamento, não precisaríamos ter o atributo definição, conforme **Figura 12**.

Podemos perceber que na **Figura 12** apresentamos a tabela ProdutoServiço com os atributos comuns tanto a produto quanto a serviço. Entretanto, neste caso, são todos os atributos. Assim, a tabela Produto só possui um atributo que é a chave estrangeira que referencia a tabela ProdutoServiço, ou seja, quando for um produto, também cadastraremos um registro na tabela Produto. Da mesma forma para a tabela Serviço.

Como neste nosso estudo de caso definimos os atributos como sendo os mesmos para produtos e serviços, escolhemos o modelo de dados da **Figura 11** para darmos sequência neste exemplo. O modelo de dados da **Figura 12** teria melhor utilização caso tivéssemos atributos diferentes de produtos e serviços, como, quantidade de produto em estoque, fornecedores de produtos, dentre outros atributos que Serviços não possuem.

Assim, de posse da Tabela ProdutoServiço (**Figura 11**), precisaremos verificar

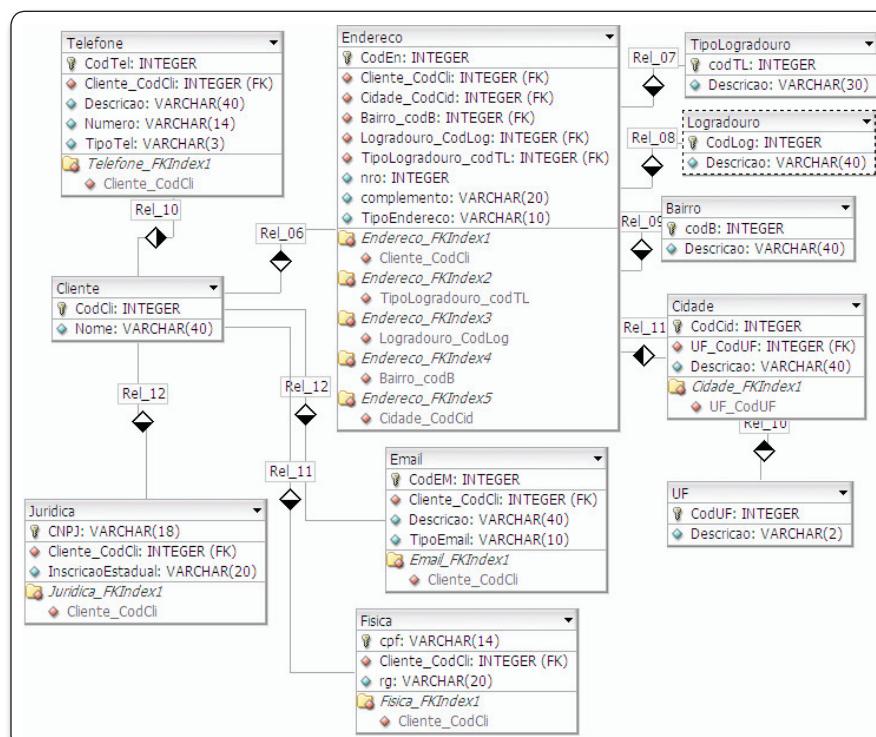


Figura 10. Relacionamentos Envolvendo Cliente, Normalizado

se ela está normalizada. O primeiro passo é verificar se ela está na 1FN. Percebemos que esta tabela está na 1FN, pois não possui atributos multivalorados ou compostos.

Uma vez definida a tabela para armazenar os produtos e serviços, podemos partir para a criação de orçamentos. Na **Figura 13** apresentamos um modelo de dados para os orçamentos. Para que um orçamento seja realizado, é necessário um cliente, a data do orçamento e o valor total. Entretanto, para que se tenha o valor total, é necessário conhecer todos os itens do orçamento, que podem ser produtos e/ou serviços, que estarão armazenados na tabela ItemOrcamento. Assim, nesta tabela, será armazenado o código do produto_servico (código) o código do orçamento(codorc), a quantidade de produtos ou serviços do referido item e o valor deste item.

O próximo passo é verificar se este modelo de dados da **Figura 13** está normalizado de acordo com a 1FN. Podemos perceber que sim, uma vez que não existem atributos compostos ou multivalorados. Também podemos verificar que o referido modelo encontra-se na 2FN uma vez que não existem tabelas com chaves primárias compostas. Já na 3FN verificamos que as tabelas Orcamento e ItemOrcamento possuem os atributos valor que não dependem de suas chaves primárias. Isto quer dizer que estas duas tabelas não estão na 3FN. Em outras palavras, podemos verificar que este atributo valor pode ser calculado a partir de outros atributos, ou seja, em Item Orcamento, o atributo valor nada mais é que o resultado entre (Qdade*Preco). E o atributo valor da tabela Orcamento nada mais é que a somatória do atributo valor para todos os itens que compõe o orçamento em questão.

ProdutoServico	
Código:	INTEGER
Nome:	VARCHAR(40)
Descrição:	VARCHAR(40)
Preço:	FLOAT
Definição:	CHAR(1)

Figura 11.Tabela ProdutoServico

Entretanto, a ausência destes valores que podem ser calculados podem causar alguns danos a longo prazo para as informações a serem geradas. Por exemplo, se não existisse o atributo valor na tabela ItemOrcamento, quando o preço do produto ou serviço for alterado seria impossível ter uma informação do preço do produto considerado no momento em que foi feito um determinado orçamento. Mas, ao desnormalizarmos segundo a 3FN esta tabela ItemOrcamento, conseguíramos ter acesso a esta informação. Para isto, basta dividir o atributo valor pelo atributo qdade da tabela ItemOrcamento, mesmo que o preço do produto já tenha sido alterado.

Outro ponto que deve ser considerado é o atributo valor da tabela Orçamento. Este atributo tem a única finalidade de já armazenar a somatória dos preços por itens de orçamento, visando facilitar a geração dos relatórios futuros. Com isto, ao gerar o referido relatório, não seria

necessário efetuar cálculos (**Listagem 4**). Caso não existisse este atributo, para calcular o valor do orçamento seria necessária uma consulta para efetuar cálculos para toda solicitação desta informação, conforme consulta da **Listagem 3**.

Assim, neste estudo de caso optamos por não normalizar as tabelas Item Orcamento e Orcamento de acordo com a 3FN pelos motivos citados.

Uma vez efetuado o modelo de dados para os orçamentos podemos iniciar a modelagem para o armazenamento das ordens de serviço. Repare que neste caso para toda ordem de serviço, devemos ter um orçamento prévio e somente após a aprovação do orçamento é que será dado início à ordem de serviço.

Para este caso, teremos algumas opções para modelagem e as apresentaremos a seguir. A primeira delas seria relacionar a tabela Orcamento com a tabela OS conforme **Figura 14**.

Entretanto, para este modelo devemos

Listagem 3. Cálculo do valor do orçamento

```
SELECT SUM(valor)
FROM ItemOrcamento
WHERE ItemOrcamento.Orcamento_CodOrc = X
--Tal que X é o número do orçamento que se quer calcular o valor
```

Listagem 4. Consulta Valor do orçamento

```
SELECT valor
FROM Orcamento
WHERE Orcamento.CodOrc = X
--Tal que X é o número do orçamento que se quer calcular o valor
```

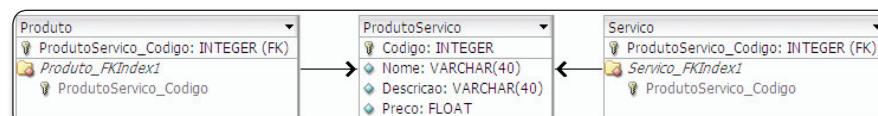


Figura 12.Tabela ProdutoServiço com Relacionamento Generalização Especialização

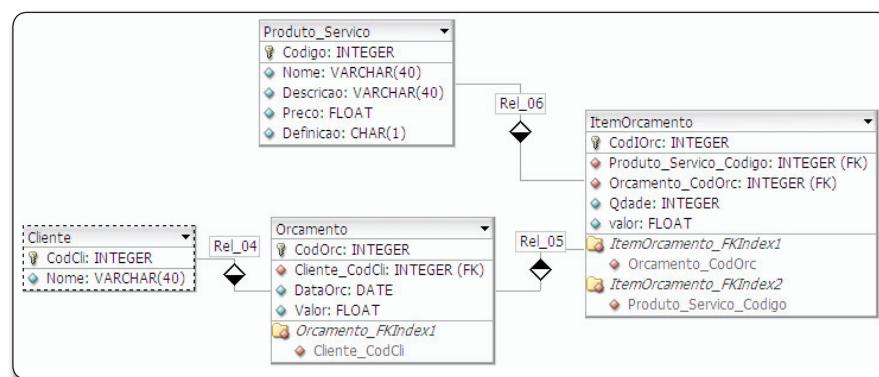


Figura 13.Relacionamento Orçamento

verificar o seguinte fato: todos os itens do orçamento foram aprovados para a execução da OS? Para o modelo da **Figura 14** a resposta seria afirmativa. A única coisa que estamos armazenando seria a data de aprovação do orçamento(DataAprov), a data prevista para conclusão (DataConclusao), valor aprovado (ValorAprov), desconto ou acréscimo. Isto é, o valor aprovado pode ser diferente do valor do orçamento, tendo apenas um desconto ou um acréscimo. Da mesma forma como na **Figura 13**, esta tabela não estaria normalizada segundo a 3FN visando facilitar estes cálculos, pois caso não existisse estes atributos desconto e acréscimo, os mesmos poderiam facilmente ser calculados. Basta verificar o valor aprovado da tabela OS e o valor da tabela Orcamento. Caso a diferença de

valor aprovado por valor fosse positiva, houve um acréscimo. Caso contrário, um desconto. Mas, para este estudo de caso também vamos optar por manter da forma apresentada na **Figura 14**.

Todavia, gostaríamos de verificar quais itens do orçamento foram aprovados para execução. Podemos ter claramente este caso uma vez que você pode não aprovar todos os itens por diversos motivos. Assim, podemos gerar uma tabela ItemOS que seria responsável por armazenar somente os itens que foram aprovados, conforme **Figura 15**. Mas, caso optemos por este modelo, podemos perceber que é possível gerar uma inconsistência de dados muito grande e grave. Um usuário pode cometer um erro inconsciente, por exemplo, de armazenar os itens da ordem de serviço diferentes dos itens do

orçamento. Isto seria um grave erro, uma vez que a ordem de serviço é um resultado de um orçamento para este estudo de caso. Assim, transferiríamos a responsabilidade de manter a consistência dos dados para a aplicação, ou seja, o software deve passar a fazer estas validações.

Para o modelo de dados da **Figura 15**, também não normalizamos de acordo com a 3FN pelos mesmos motivos do modelo anterior. Repare que a tabela ItemOS possui os atributos ValorAprov, Desconto e Acrescimo. Estes atributos possuem a mesma finalidade dos atributos semelhantes da tabela OS. Com a existência deles, poderemos controlar para cada registro da tabela, se foi concedido um desconto ou um acréscimo.

Para resolver este problema da possibilidade da inconsistência dos dados, podemos alterar o modelo de dados da **Figura 14** adicionando alguns atributos conforme **Figura 16**.

Podemos perceber que foram adicionados os seguintes atributos na tabela ItemOrcamento:

- Aprovado: que recebe um caractere S ou N para informar se o item foi aprovado ou não.
- Desconto: que recebe o valor do desconto individual, ou seja, se o item em questão teve um desconto no preço.
- Acrescimo: que recebe o valor do acréscimo individual, ou seja, se o item em questão teve um acréscimo no preço.
- ValorAprov: que recebe o valor aprovado para a ordem de serviço.

Dessa forma, o atributo valor da tabela ItemOrcamento mantém o valor original do orçamento feito e o valor aprovado seria o valor aprovado por item para facilitar também os cálculos para a tabela OS. Os atributos ValorAprov, Desconto e Acrescimo da tabela OS continuariam com a mesma função, ou seja, armazenar o valor aprovado da ordem de serviço, os descontos ou acréscimo. Suponha que nenhum item teve concedido desconto ou acréscimo individualmente, mas ao fechar a OS tenha sido concedido um desconto. Na tabela OS teríamos este valor armazenado.

Desta forma, verificamos que a tabela ItemOrcamento, Orcamento e OS não

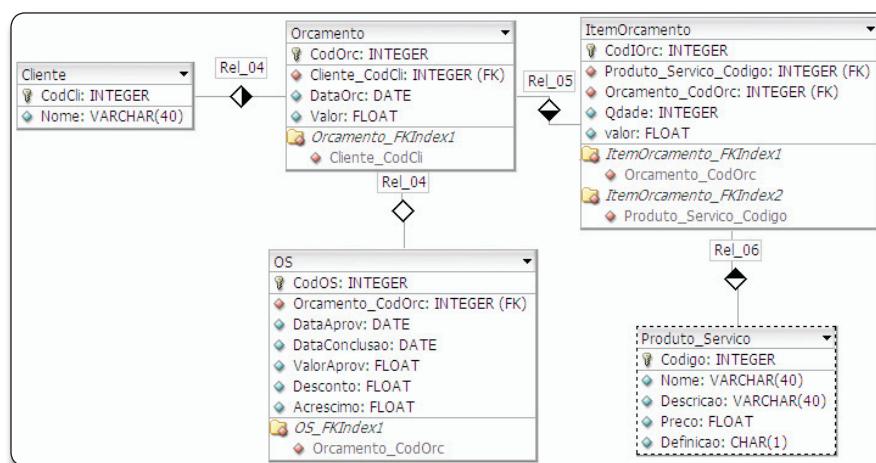


Figura 14. Relacionamento Orçamento x Ordem de Serviço (OS)

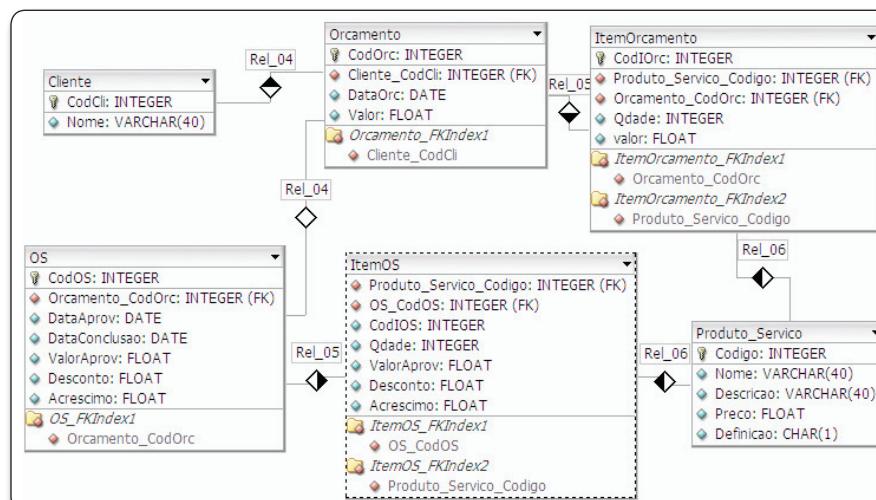


Figura 15. Relacionamento Orçamento x Ordem de Serviço (OS) x ItemOS

estão normalizadas segundo a 3FN. Mas, de acordo com a 1FN e 2FN elas estão.

Na **Figura 17** apresentamos o modelo de dados para ordens de serviço completo. Nele consta a opção da **Figura 16** para a aprovação dos itens do orçamento.

Considerações finais

Podemos perceber que a adoção da normalização de dados é de grande valia para que o projeto de um banco de dados não possua redundância e consequentemente inconsistência. Porém, vale ressaltar que a normalização não pode gerar perdas no poder de geração de informações a partir dos bancos de dados. Caso isto ocorra, em muitos casos pode ser interessante o processo de desnormalização para melhorar o desempenho das consultas, conforme abordamos no decorrer dos exemplos apresentados e do artigo “Normalização de Dados”, desta revista em sua edição de número 47. Entretanto, este custo pode ser muito alto, comprometendo a garantia de consistência dos dados.

Em resumo, um bom projeto de banco de dados deve oferecer informações com qualidade para tomada de decisões por parte dos gestores das organizações. E para que isto ocorra, devemos avaliar com bastante critério a normalização de dados para garantir a consistência dos dados ou os pontos onde não irá ocorrer normalização e por quais motivos. Além disso, devemos avaliar se a não normalização irá gerar impactos negativos para a qualidade das informações.

Neste artigo apresentamos o modelo de dados para uma biblioteca e para um sistema de ordens de serviço. Ambos, com várias restrições de escopo conforme abordados no decorrer deste artigo. Entretanto, vale frisar que mesmo com estas restrições, inúmeras informações podem ser geradas através deste modelo, dentre as quais podemos citar, para o sistema de bibliotecas:

- Livros cadastrados;
- Alunos cadastrados;
- Professores cadastrados;
- Editoras cadastradas;
- Autores cadastrados;
- Assuntos cadastrados;
- Livros por assunto;

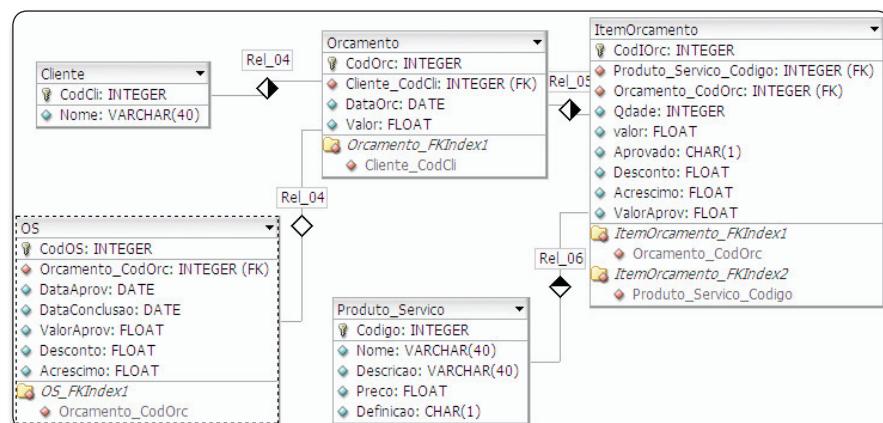


Figura 16. Ordem de Serviço

PENSE...

QUANTO TEMPO
VOCÊ GASTARIA
PARA DESENVOLVER
COBRANÇA COM BOLETOS
BANCÁRIOS PARA
APENAS UM BANCO
NO SEU SOFTWARE

COBREDEM

 56 BANCOS E MAIS DE 430 CARTEIRAS DE COBRANÇA PARA IMPRESSÃO E/OU ENVIO DE BOLETO BANCÁRIO POR EMAIL;

 GERAÇÃO DE BOLETOS ON LINE;

 GERAÇÃO E LEITURA DE ARQUIVOS (REMESSA/RETORNO) NOS PADRÕES FEBRABAN E CNAB;

 MAIS DE 40 EXEMPLOS EM DIVERSAS LINGUAGENS DE PROGRAMAÇÃO



DOWNLOADS E INFORMAÇÕES EM WWW.COBREDEM.COM

- Percentuais de empréstimo por bairro das pessoas;
 - Livros mais emprestados;
 - Pessoas que mais pegaram livros emprestados;
 - Quantidade de livros emprestados por pessoa e por período;
 - Livros que não foram devolvidos;
 - Livros que foram devolvidos fora do prazo;
 - Quantidade de exemplares de cada livro.

Para o sistema de ordens de serviço também podem se gerar um bom volume de informações para tomadas de decisão, tais como:

 - Relação de clientes cadastrados;
 - Relação de clientes por bairro;
 - Relação de produtos cadastrados;

Para o sistema de ordens de serviço também podem se gerar um bom volume de informações para tomadas de decisão, tais como:

- Relação de serviços cadastrados;
 - Relação de Orçamentos Efetuados;
 - Relação de Orçamentos Aprovados;
 - Relação de Orçamentos não Aprovados;

• Relação de Ordens de Serviço;

• Descontos concedidos por Ordem de Serviço e por cliente;

• Acréscimos efetuados por Orçamento e por cliente;

• Produtos que mais foram utilizados nas ordens de serviço;

• Serviços mais executados;

• Cliente que mais efetuou serviços;

• Percentuais de clientes por bairro;

• Bairros que geram mais faturamento;

- Quantidade de Ordens de Serviço Executadas.

Um sistema de informação gerencial obrigatoriamente precisa ter um modelo de dados que possua dados armazenados de forma que seja possível gerar informações. Entretanto, infelizmente ainda existem no mercado diversos sistemas que não possuem tal foco. Muitos deles apenas substituem o meio de armazenamento de papel para os computadores e informações para tomadas de decisão praticamente não existem. Este tipo de sistema não pode ser tratado e devemos focar claramente em informações gerenciais, mesmo que o sistema inicialmente seja apenas transacional, ou seja, apenas para armazenar dados relativos às informações rotineiras da empresa. Mesmo em um sistema transacional, o gestor da área precisará tomar decisões em algum momento da atividade profissional. ●

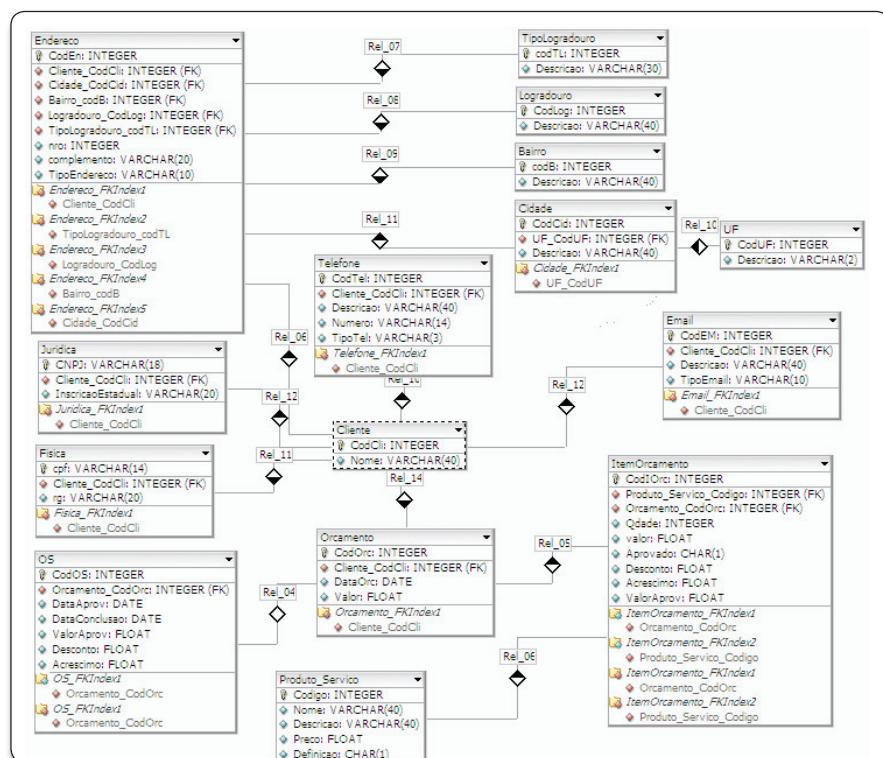


Figura 17. Ordem de Serviço Final

Referências

DATE, C.J. Introdução a Sistemas de Bancos de Dados, Editora Campus, 2004.

HARRINGTON, J. L. Projetos de Bancos de Dados Relacionais, Editora Campus, 2002.

MONTEIRO, Emiliano S. Projeto de Sistemas e Bancos de Dados, Brasport, 2004.

MULLER, Robert J. Projeto de Banco de Dados, Editora Berkeley, 2002.

ROCHA JÚNIOR, Ary dos Santos, Normalização de Dados, SQL Magazine número 47, DevMedia, outubro 2007.

