Banco de Dados I

Ciclo 19/04/2021 a 30/04/2021

Prof. Dr. Tiago Nogueira
Curso Superior em Gestão da Tecnologia da Informação
Instituto Federal Baiano

CURSO SUPERIOR EM GESTÃO DA TECNOLOGIA DA INFORMAÇÃO - INSTITUTO FEDERAL BAIANO, CAMPUS BOM JESUS DA LAPA

Caro estudante,

Bem-vindos ao Ciclo (19 a 30 de abril) da disciplina de Banco de Dados I que tratará os conceitos teóricos para aprofundar seus conhecimentos sobre Banco de Dados no curso Superior em Gestão da Tecnologia da Informação do Instituto Federal Baiano – Campus Bom Jesus da Lapa.

Para que seu estudo se torne proveitoso e prazeroso, este ciclo foi organizada em 2 semanas, com temas e subtemas que, por sua vez, são subdivididos em seções (tópicos), atendendo aos objetivos do processo de ensino-aprendizagem.

Nesta primeira semana, trataremos sobre os conceitos de modelos de dados, procuraremos compreender os conceitos gerais sobre modelagem de dados (modelo de dados, Entidade-Relacionamento, Relacionamentos, Atributos Chaves, Normalização e Padronização).

Esperamos que, até o final deste ciclo vocês possam:

- Ampliar a compreensão sobre Banco de Dados; - Conceituar os Sistemas Gerenciadores de Banco de Dados (SGDB); - Compreender a importância da normalização de padronização;

Para tanto, teremos vídeo aulas expositivas que serão disponibilizadas em nosso Ambiente Virtual de Aprendizagem (AVA) do Instituto Federal Baiano, campus Bom Jesus da Lapa.

Porém, antes de iniciar a leitura, gostaríamos que vocês parassem um instante para refletir sobre algumas questões: Qual é a importância de Banco de Dados para as organizações?

Não se preocupe. Não queremos que vocês respondam de imediato essa questão. Mas esperamos que, até o final, vocês tenham respostas e também formulem outras perguntas.

Vamos, então, iniciar nossas aulas? Bons estudos!

1ª Ed., Abril de 2021

Sumário

I Parte Um

1. Conceitos de modelos de dados

Objetivo

Entender o porquê de modelar dados, os principais conceitos envolvidos e quais são os benefícios para o processo de desenvolvimento de sistemas.

1.1 Introdução

O objetivo deste texto é oferecer a você, estudante do Curso Superior em Gestão da Tecnologia da Informação, e também aos profissionais da área, uma visão abrangente e profunda sobre o tema. A ideia é fazer uma abordagem pragmática e objetiva, proporcionando a transmissão de conhecimentos teóricos aplicados de forma crítica a partir da experiência do autor no desenvolvimento, manutenção e implantação de diversos tipos de sistemas de informação, em empresas dos mais variados portes e ramos de atividades.

O texto utiliza uma metodologia em que um conteúdo teórico é seguido de um estudo de caso que serve como pano de fundo para discussão das possíveis soluções e de seus impactos no resultado final de um produto de software. O MySQL foi utilizado como banco de dados de referência para apresentação dos exemplos práticos.

No decorrer das aulas, o objetivo é mostrar e justificar para o leitor quais são as vantagens em projetar um modelo de dados com rigor, utilizando todo o poder oferecido pelos SGBDs atuais. Afinal os SGBSs são não somente um local seguro para armazenar e recuperar dados, mas principalmente um local capaz de garantir a riqueza semântica dos dados. Esta estratégia procura trazer parte do conhecimento ou das regras do negócio, que normalmente ficam espalhadas pelo código–fonte das linguagens de programação para dentro do banco de dados. Isto certamente trará ganhos de produtividade se considerado todo o processo de desenvolvimento de software.

Estudar este tema com afinco é vital para a formação do profissional. Portanto, espero ajudá-lo alcançar esse objetivo. Vamos ao trabalho!

1.2 Conceitos Gerais

Um **banco de dados** ou **base de dados** (sua abreviatura é BD, em inglês DB, *database*) são conjuntos de dados com uma estrutura regular que tem como objetivo organizar uma informação. Um banco de dados normalmente agrupa informações utilizadas para um mesmo fim de forma que possam representar coleções de informações que se relacionam de forma que crie um sentido. São de vital importância para empresas, e há duas décadas se tornaram a principal peça dos sistemas de informação.Um banco de dados é uma coleção de dados relacionados.

Entende-se por dado, toda a informação que pode ser armazenada e que apresenta algum significado dentro do contexto ao qual ele se aplica. Por exemplo, num sistema bancário, uma pessoa é identificada pelo seu cpf (cliente). Em um sistema escolar a pessoa é identificada pelo seu número de matrícula (aluno). Em um sistema médico a pessoa (paciente) é identificada pelo número do plano de saúde ou cartão SUS.

A lista telefônica é um exemplo de banco de dados. Nela percebemos que todos os dados referentes a uma pessoa estão na mesma linha, a isso chamamos de registros. O tipo ou categoria da informação (nome, telefone, etc.) sobre uma pessoa está separada em colunas, as quais chamamos campos. Uma lista de compras, lista telefônica, lista de contatos são exemplos de banco de dados presentes em nosso dia-dia.

Um banco de dados informatizado é usualmente mantido e acessado por meio de um software conhecido como **Sistema Gerenciador de Banco de Dados (SGBD)**, que e muitas vezes o termo banco de dados é usado como sinônimo de SGDB. Um SGBD - Sistema de Gerenciamento de Banco de Dados é uma coleção de programas que permitem ao usuário definir, construir e manipular Bases de Dados para as mais diversas finalidades.

Um banco de dados pode ser local, quer dizer utilizável em uma máquina por um usuário, ou repartida, quer dizer que as informações são armazenadas em máquinas distantes e acessíveis por rede. A vantagem essencial da utilização dos bancos de dados é a possibilidade de poder ser acessada por vários usuários, simultaneamente.

O modelo de dados mais adotado hoje em dia para representar e armazenar dados em um SGBD é o modelo relacional, onde as estruturas têm a forma de tabelas, compostas por linhas e colunas.

1.3 Conceitos sobre modelagem de dados

Esta aula começa apresentando uma discussão da importância dos dados como a base para o desenvolvimento do software. A partir daí estuda-se o que é e quais são os principais conceitos envolvidos no processo de modelagem de dados. Em destaque, uma discussão sobre as três formas normais e suas aplicações, pois afinal normalização é um importante conceito da modelagem de dados, que evita as anomalias, tais como redundância e perda de informação. Bom! Mas isso é o que veremos a seguir.

1.3.1 Por que devemos modelar dados?

Durante as últimas décadas, o processo de desenvolvimento de sistemas para fins comerciais vem ocorrendo tanto para uso específico de alguma empresa quanto elaborado por empresas desenvolvedoras de soluções para seus clientes. Nesse período surgiram diversas metodologias para o desenvolvimento de sistemas, diversas formas de abordar o problema e propor solução: análise estruturada, essencial, orientada a objetos, etc. Sem falar no lado tecnológico, que sofre uma nova onda a cada período e com ciclos cada vez menores. Foram várias as linguagens de programação e ambiente de desenvolvimento: Cobol, Clipper e a família dBase, o Delphi, o VisualBasic, PHP, Java, etc. Convivemos com diversas formas de interface com o usuário: texto, semigráfica, gráfica, web, etc. Durante todo esse período houve muitos erros e acertos. O sucesso da informática é inexorável; a prova disso é que ela está em toda a parte. O uso da tecnologia da informação é visto pelas empresas não apenas como necessário para condução das tarefas, mas, como uma vantagem competitiva. Em contrapartida, tivemos também muitos casos de insucesso na implantação de sistemas, em que um enorme volume de recursos foi gasto sem que se houvesse um retorno econômico esperado. Não é tão raro presenciarmos cancelamentos de projetos nos quais já se investiram valores significativos.

Diante desse mundo da TI, em que tudo é efêmero, vivenciamos situações em que sistemas de informação têm de ser substituídos, não porque suas funcionalidades não atendem aos processos de negócio, mas apenas porque estão defasados tecnologicamente. Isso leva à reconstrução de sistemas inteiros, trocando o ambiente de desenvolvimento, reescrevendo todos os programas para obter um novo resultado final, em um novo paradigma tecnológico. Mudanças tecnológicas normalmente levam a mudanças da metodologia de desenvolvimento de sistemas, pois, cada uma delas é mais adequada a um determinado arcabouço tecnológico. Entretanto, diante de tudo isso, podemos observar, que independe da arquitetura utilizada na construção da solução: sistema multiusuário, cliente-servidor ou em camadas. Ou ainda, no tocante à interface do front end: texto, gráfico ou web. O fato é que, desde a década de 1970, os bancos de dados estão aí como solução de repositório de dados e informações. Eles garantem integridade,

segurança, confiabilidade, etc. Dessa forma, podemos afirmar sem medo de estarmos cometendo algum equívoco, que a parte mais estável no tocante às tecnologias que envolvem as soluções de TI está relacionado com os dados e com a forma em que eles são armazenados e disponibilizados.

A seguir apresentaremos mais alguns argumentos de que a partir do entendimento do problema, uma boa forma de propor uma solução é tomar como base a elaboração de um modelo de **persistência** de dados com grande riqueza **semântica**.

- O dado é a origem da informação. Muitas das vezes não é possível, independentemente da tecnologia, alterar programas para obter uma informação desejada. Ou seja, armazenando apenas o todo não é possível conhecer as partes que o compõem.
- O dado é a infraestrutura do sistema e está por baixo da estrutura. Ele suporta a estrutura. Portanto, mexer na forma de armazenar os dados impacta toda a estrutura. O mesmo não é sempre verdade quando temos mudança na regra de negócio contida nos programas ou na forma visual da interface.
- Normalmente, as falhas de normalização irão gerar esforço de programação a fim de evitar inconsistência dos dados.
- Definir semântica no modelo de dados reduz a quantidade necessária de código- fonte a ser escrito no programa aplicativo.

Embutir regras de negócio no banco de dados melhora o desempenho do sistema, garante a unicidade do código, além de simplificar o programa de aplicação e criar maior independência da interface e do aplicativo.

Mas, como nem tudo são louros, obviamente, utilizar o banco de dados em sua plenitude aumenta o grau de dificuldade na sua administração, entretanto, de uma forma bastante controlada e justificável. Por fim, o trabalho é menor e o resultado será sempre melhor.

1.3.2 Modelo de dados ER

O modelo de dados é uma forma de abstração do mundo real. É um recorte no qual criamos um "minimundo" do que se deseja representar. O modelo expressa para o sistema a ser elaborado um entendimento de quais dados devem ser armazenados, qual é a relação de dependência que existe entre eles e qual é o significado semântico desses dados e de suas relações.

O modelo ou Diagrama de Entidade e Relacionamento (DER) é uma das formas de abordagem semântica mais conhecida e provavelmente das mais utilizadas atualmente. Ele foi proposto por Peter Chen em 1976, sendo, a partir daí, refinado e ampliado por diversos autores. Um DER é composto por Entidades (tabela que agrupam os dados de algo que se deseja armazenar no banco de dados). Ex.: (Aluno,

Curso) e Relacionamentos (representa a forma de como as entidades se relacionam entre si). É importante destacar que as entidades são compostas de atributos, os quais descrevem as características da própria entidade. (Ex.: nome do aluno, número da matrícula do aluno, nome do curso, etc.).

1.3.3 Definindo entidade/atributos

Segundo Korth e Silberschatz (1989), uma entidade é um objeto que existe e se distingue de outros objetos. Um CNPJ de uma empresa 17.111.222.0001-44 é uma entidade, uma vez que identifica univocamente uma empresa/filial. As várias empresas formam o conjunto de entidades do mesmo tipo que formam a **relação**[1] Empresa. Elmasri e Navathe (2000) corroboram utilizando essa mesma nomenclatura. Date (2004) descreve o termo entidade como sendo normalmente utilizado na área de banco de dados para indicar qualquer objeto distinguível que seja neste representado.

Conforme definido pelos autores o termo entidade está relacionado a uma linha ou ocorrência em uma tabela do banco de dados, enquanto que, o termo conjunto de entidades estaria associado à tabela propriamente dita. Na prática, nenhum autor faz distinção entre esses termos, utilizando apenas o termo entidade como sinônimo de tabela.

Uma entidade pode ser: um objeto com existência física (entidade concreta), ex.: um empregado, um aluno, um carro; ou conceitual (entidade abstrata), ex.: uma empresa, uma matrícula, um relacionamento entre duas ou mais entidades.

Uma entidade pode ser descrita por propriedades particulares que a distinguem e a descrevem. Essas propriedades são chamadas de atributos da entidade.Fazendo associação às tabelas dos bancos de dados, as linhas são as ocorrências e os atributos são as colunas da tabela. Por exemplo: na entidade Piloto mostrada na Figura 1.1, temos dois atributos (colunas): CodPiloto, NomPiloto e três linhas nas quais é mostrado o conteúdo para cada piloto cadastrado na tabela. Cada ocorrência ou linha da entidade possui valores (atributos) que identificam um único piloto.

Tabela - Piloto

Cod_Piloto	Nom_Piloto
ALO	Fernando Alonso
MAS	Felipe Massa
RUB	Rubens Barrichello

Figura 1.1: Tabela Piloto

[1] termo "relação" pode ser entendido nesse contexto como sinônimo de tabela.

Os atributos podem ser:

- **Simples (atômicos)** – quando não fizer sentido semântico dividir o atributo em subpartes. Exemplo: a altura ou o peso de uma pessoa.
- **Composto** – quando o atributo puder ser dividido em subpartes com significado semântico. Exemplo: endereço de uma pessoa pode ser dividido em Cep, cidade, bairro, logradouro, número e complemento.
- **Multivalorado** – quando o atributo possuir simultaneamente mais de um valor para uma mesma ocorrência da entidade. Exemplo: a cor de um carro pode ser parte preta e parte branca.
- **Derivado** – quando o atributo puder ser calculado dinamicamente sem que haja necessidade de manter o dado armazenado no banco de dados. Exemplo: em uma entidade com itens de uma nota fiscal. A partir do preço unitário e da quantidade vendida o valor total do item pode facilmente ser calculado pela multiplicação dos dois termos.
- **Obrigatório** – quando não for possível incluir uma linha na tabela sem que haja alguma informação/valor nos atributos obrigatórios dela.
- Não obrigatório – quando ao inserir uma linha na tabela isto possa ser feito independentemente da obrigação de informar dados aos atributos que aceitam valores nulos.
- Complexo – quando juntamos os conceitos de atributos composto e multivalorados.
- **Delimitado** – quando existir definição no banco de dados de uma regra de domínio que delimite os valores válidos possíveis a serem armazenados.

1.3.4 Definindo relacionamentos

O relacionamento é uma representação da forma de como as entidades se relacionam umas com as outras. Ele expressa certas restrições existentes no minimundo a ser modelado que delimitam como uma ou mais ocorrências de uma entidade se relacionam com uma ou mais ocorrências de outra entidade. Este conceito é denominado cardinalidade do relacionamento. Para um conjunto de relacionamento binário, entre duas tabelas T1 e T2, a cardinalidade pode ser:

Um para um – uma linha da tabela T1 está associada com no máximo uma linha da tabela T2 e vice-versa, conforme mostrado na Figura1.2 a seguir.

Um para muitos – uma linha da tabela T1 está associada com de zero a “n” linhas da tabela T2, enquanto que, uma linha da tabela T2 está associada com no máximo uma linha da tabela T1, conforme mostrado na Figura 1.3. Observe que relacionamentos muitos para um é o inverso de um para muitos.

Muitos para muitos – uma linha da tabela T1 está associada com zero a “n” linhas da tabela T2 e

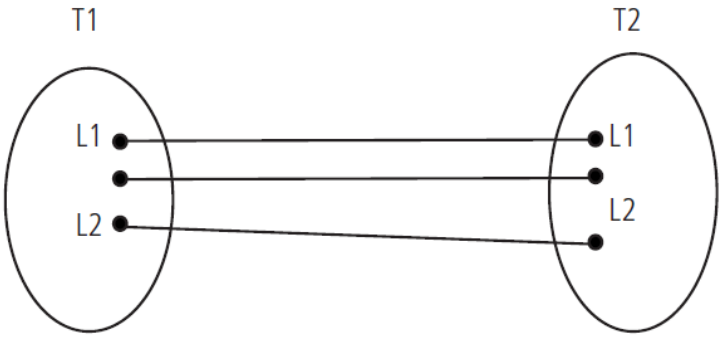

Figura 1.2: Representação da Cardinalidade um para um

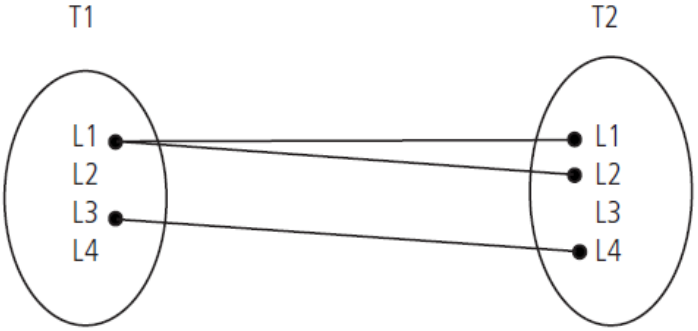

Figura 1.3: Representação da Cardinalidade um para muitos

vice-versa, conforme mostrado na Figura 1.4 a seguir.

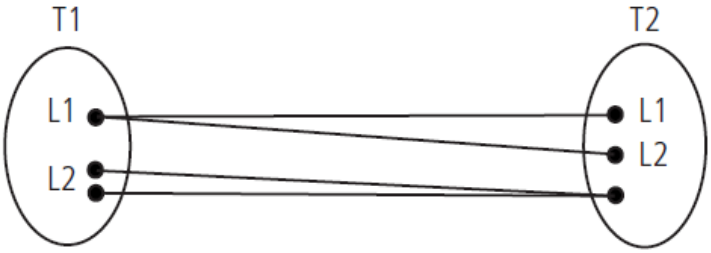

Figura 1.4: Representação da Cardinalidade muitos para muitos

1.3.5 Atributo chave (Candidata, Primária e Estrangeira)

Outra importante tarefa da modelagem é determinar dentre os atributos de uma entidade/tabela quais são aqueles que de forma individual ou agrupada são capazes de identificar uma única ocorrência da entidade. Esse conjunto de atributos é chamado de chaves candidatas. Ou seja, são os possíveis candidatos a serem definidos como chave primária. Entende-se como chave primária a menor quantidade de atributos capaz de identificar uma ocorrência da tabela de forma unívoca.

Podem ocorrer casos em que tenhamos diversos conjuntos distintos de chaves candidatas. Cabe ao projetista escolher dentre as possibilidades aquela que for mais adequada como solução para o problema

proposto. Vejamos um exemplo da entidade Veículo. Dentre os atributos de um veículo, tais como marca, modelo, ano de fabricação, cor, placa e chassi, Renavam, etc., existem pelo menos três possíveis chaves candidatas: placa, chassi e Renavam, pois, cada uma delas, individualmente, identifica de forma única um veículo. Para um sistema que trabalhe apenas com veículos licenciados para rodar no Brasil, o uso da placa como chave é provavelmente o mais apropriado, pois placa pode ser representada por um menor número de caracteres que o Renavam e o chassi. Portanto, é mais fácil de digitar e mais fácil de visualizar. O chassi fica posicionado no veículo em locais mais difíceis de ver do que a placa, e o Renavam fica informado apenas no documento do veículo. Já o uso do chassi como chave primária poderia ser utilizado em um sistema de controle de veículos novos, não emplacados, que estão sendo transferidos, importados e exportados, entre diversos países. O que queremos dizer é que a motivação para a tomada de decisão depende de fatores externos às regras de modelagem.

É também possível que uma entidade não possua atributos suficientes para formação de uma chave primária. Nesse caso poderá ser arbitrado um novo atributo sem significado semântico, mas que sirva com chave ou parte da chave. Normalmente uma sequência numérica serve como atributo complementar nas entidades fracas, que herdam a chave da tabela pai, entidade forte. Esse atributo numérico sequencial sempre começa de um (1) para cada registro da tabela associado à tabela pai. Ele representa uma sequência de ocorrências na tabela filha, relacionada a apenas uma ocorrência da tabela pai. Veja um exemplo de uma entidade Funcionário (entidade forte) e Dependente (entidade fraca) mostrado na Figura 1.5. O funcionário "José dos Santos" tem três dependentes, sequência (1, 2 e 3).

Funcionario		
Cod_Funcionario	Nom_Funcionario	Nro_Telefone
1	José dos Santos	(31) 2222-2222
2	Maria da Silva	(31) 3333-3333

Dependente		
Cod_Funcionario	Seq_Dependente	Nom_Dependente
1	1	José dos Santos Junior
1	2	Antônio dos Santos
1	3	Joana dos Santos
2	1	João Pedro

Figura 1.5: Representação das tabelas Funcionário e Dependente

Outro exemplo que podemos citar é o de uma entidade que não possui nenhum atributo natural capaz de representar de forma unívoca uma linha da tabela. Nesse caso, a chave primária pode ser definida como um atributo **autoincrementado**[2].

Um importante conceito é o de **chave estrangeira**[3]. A chave estrangeira é utilizada para ligar uma ocorrência de uma entidade a uma ou mais ocorrências de outra entidade. Ela se caracteriza por aparecer

[2]É uma sequência numérica gerada automaticamente pelo SGBD.

[3]Corresponde a um conjunto de um ou mais atributos de uma linha de uma entidade com o objetivo de referenciar uma ou mais linha em outra entidade na qual esses mesmo atributos são uma chave primária parcial ou completa.

em uma entidade com atributo originado de outra entidade na qual ela é uma **chave primária**[4]. Ou seja, a chave estrangeira é uma referência numa entidade que aponta para a chave primária de outra entidade, possibilitando dessa forma as junções entre as entidades envolvidas. É ela que informa ao banco de dados a integridade referencial entre as tabelas envolvidas no relacionamento.

1.3.6 Normalização

A normalização é um conjunto de regras criado para ajudar os projetistas de modelos de dados a definir qual é a melhor forma de agrupar os atributos em uma tabela e de como essas tabelas se relacionam criando um esquema que evite as anomalias de atualização dos dados na inserção, remoção ou atualização. As anomalias, características indesejáveis, que devemos eliminar do modelo são: redundância da informação; incapacidade de junção adequada da informação e perda da informação.

Edgard F. Codd (apud DATE,2002), em 1972, foi o primeiro a escrever sobre o conceito de decomposição sem perdas e dependência funcional, que é a base para o procedimento de normalização. Inicialmente Codd propôs as três formas normais, conhecido pelo acrônimo 3FN. A dependência funcional é uma propriedade que nos leva a decompor uma relação com o objetivo de fazer com que todos os atributos em uma tupla (linha) dependam exclusivamente da sua chave primária. O conceito de decomposição sem perdas significa dizer que ao decompor uma relação em duas ou mais relações, não poderá haver perdas de informação. Todos os atributos deverão permanecer no banco de dados. A decomposição apenas rearranja esses atributos em novas tabelas a fim de evitar as anomalias citadas anteriormente.

Vamos usar como referência para explicar o tema o seguinte exemplo: suponha que desejamos armazenar dados dos empregados de uma empresa em um banco de dados. Os seguintes atributos devem ser persistidos:

- código do empregado (Cod Emp);
- nome do empregado (NomEmp);
- data de admissão (DatAdmis);
- nome do dependente (NomDep);
- data de nascimento do dependente (Dat Nasc);
- descrição da função (DesFuncao);
- percentual de participação no lucro (PerPar).

A seguir, na Figura 1.6, apresentamos um exemplo de um conjunto de dados definidos para a tabela Empregado.

[4]Pode ser definida como a menor quantidade de atributos capaz de identificar uma linha de uma tabela de forma unívoca.

Cod_Emp	Nom_Emp	Dat_Admis	Nom_ Dep	Dat_Nasc	Des_ Funcao	Per_ Par
00001	José	01/05/2000	Alex Marina	05/03/2000 20/07/2005	Gerente	200
00002	Marcos	15/08/2000			Engenheiro	100
00003	Francisco	01/09/2000	Pedro Mateus Carla	12/02/1995 17/03/1998 07/08/2002	Encarregado	50
00004	Manuel	01/09/2000	Beatriz	28/06/2006	Servente	50

Figura 1.6: Tabela Empregado

Primeira forma normal 1FN – visa eliminar os atributos multivalorados, atributos compostos e suas combinações. Ou seja, para cada ocorrência da chave primária, só pode corresponder a uma ocorrência dos demais atributos não chave. Os atributos de uma relação devem ser atômicos (indivisíveis) não repetitivos. O uso dessa regra gera alguma controvérsia. Existem alguns produtos no mercado chamados NFNF, que significa Non First Normal Form, que permitem definir atributos multivalorados. Na prática, quando temos um controle bem definido dos valores de domínio de um atributo, sabemos o número exato de repetições e sendo a quantidade de repetições constante e bem reduzida é possível, por opção do projetista do modelo de banco de dados, violar a 1FN sem ficar com nenhum remorso.

Como pudemos ver no exemplo proposto pela Figura 1.6, cada empregado pode ter de nenhum até vários dependentes. Nesse caso existem três possibilidades em criar um modelo que atenda à 1FN.

A primeira solução seria o uso de atributos multivalorados ou, na inexistência destes, definir o nome seguido de uma sequência numérica, como por exemplo: NomDep1, NomDep2, ... , NomDepN. Ou seja, criar espaço para armazenar os "N" dependentes de cada empregado. Entretanto, as condições apresentadas anteriormente para abrir uma exceção quanto ao uso de atributos repetitivos não existem neste caso. Portanto, é fácil perceber que esta solução não é a mais adequada. Não parece viável, de antemão, criar, por exemplo, dez atributos para cadastrar dependentes sabendo que muitos empregados podem ter nenhum e outros podem ter mais de dez dependentes. Quanto espaço e trabalho seriam desperdiçados nesses casos? A solução ideal para cada caso depende das demandas do minimundo que queremos representar e do bom senso do projetista. Portanto, iremos descartá-la.

A segunda alternativa é que como existem vários dependentes para cada empregado e não é razoável criar atributos repetitivos, a opção seria repetir os dados de um determinado empregado em tantas tuplas quantos fossem o número de dependentes dele. Como existirão várias tuplas de um mesmo empregado, a chave primária não poderá ser composta apenas no CodEmp. Será necessário criar uma chave composta agregando, por exemplo: uma sequência numérica para o cada dependente incluído por empregado.

Também nos parece óbvio que esta solução, mesmo atendendo 1FN, gera muita redundância dos dados do empregado tais como CodEmp, NomEmp, DatAdmis e DesFuncao.

Portanto, a solução mais adequada seria remover o atributo multivalorado NomDep da entidade empregado criando uma nova entidade Dependente. A chave de Dependente é composta pela propagação da chave primária de Empregado, agregada a uma sequência numérica dos dependentes. A entidade Dependente é uma entidade fraca, pois ela herda a chave da entidade pai Empregado (entidade forte). Nela deve-se acrescentar um atributo a mais para compor sua chave primária. Pois, como não é possível determinar com exatidão quantos dependentes cada empregado pode ter, a entidade Dependente terá para um determinado empregado exatamente uma tupla para cada dependente. Veja a solução proposta na Figura 1.7 a seguir.

Empregado	Dependente
# Cod_Emp	# Cod_Emp
Nom_Emp	# Seq_Dep
Des_Funcao	Nom_dep
Per_Part	Dat_Nasc

Figura 1.7: Representação das entidades Empregado e Dependente

Segunda forma normal 2FN – uma entidade está na 2FN se, e somente se, estiver na 1FN e se cada atributo não chave depender funcionalmente da totalidade da chave primária. Basicamente, um determinado atributo tem dependência funcional com relação à chave, quando para realizar seu acesso é necessário conhecer a chave. Vejamos o atributo DesFuncao na entidade Empregado. A descrição de uma função não dependente do empregado que exerce a função. Podemos ter vários empregados que têm a mesma função. Portanto, a descrição da função está ligada a uma chave primária que identifica a função. Nesse caso, a entidade Empregado deve possuir como atributo apenas uma referência (chave estrangeira) a esta chave primária. Portanto, para adequarmos o modelo à 2FN teremos de remover os atributos da entidade Empregado que não tenham dependência funcional com a chave primária como um todo. Veja a solução proposta na Figura 1.8 a seguir.

Empregado	Dependente	Funcao
# Cod_Emp	# Cod_Emp	
Nom_Emp	# Seq_Dep	# Cod_Funcao
* Cod_Funcao	Nom_dep	Des_Funcao
Per_Part	Dat_Nasc	

Figura 1.8: Representação das entidades Empregado, Dependente e Função

É importante esclarecer que ao criarmos a entidade Função, foi necessário definir um atributo para ser

chave primária da entidade. Como proceder neste caso? Bem, primeiro devemos identificar pela natureza dos dados associados à entidade em questão, se existe alguma chave natural. Entenda como chave natural algum atributo que já é reconhecido no mundo real como unívoco para identificação de uma ocorrência ou linha da entidade. No nosso caso podemos utilizar a Classificação Brasileira de Ocupações (CBO) como uma possível chave candidata ou definir uma chave autoincrementada sem significado semântico, entre outras soluções. Mais uma vez o leitor deve estar se perguntando: qual será a melhor solução? A resposta mais uma vez é: depende. Ao adotarmos o CBO como chave, não teremos liberdade para separar ou agregar funções que são trabalhadas no minimundo modelado. Ao adotarmos um número sequencial, estaremos criando uma chave "burra" para entidade e provavelmente teremos de criar o CBO como um atributo não chave, caso tenhamos de enviar dados para o Ministério do Trabalho. Cabe ao projetista definir a melhor opção caso a caso, dependendo do contexto.

Terceira forma normal 3FN –uma entidade está na 3FN se, e somente se, estiver na 2FN e não existam atributos que dependam do(s) atributo(s) chave que não sejam chave primária, ou se já não exista dependência transitiva entre os atributos. No nosso exemplo, o percentual de participação do lucro não depende do empregado e sim da função que ele exerce. Se ele for promovido, mudar de função, automaticamente ele poderá ter seu percentual de participação alterado. Na entidade Empregado existe uma dependência transitiva entre a chave de função e o percentual de participação. Portanto, o modelo não atende à 3FN. Veja a alteração feita na Figura 1.9 para resolver o problema.

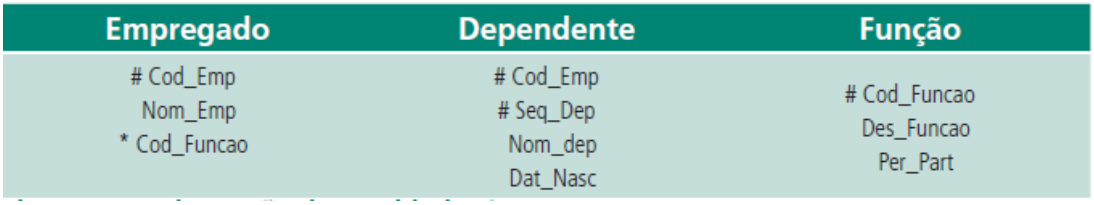

Figura 1.9: Adequação das entidades à 3FN

A Figura 1.10 apresenta um resumo do processo de normalização apresentado.

É importante ressaltar que o conhecimento e aplicação das regras de normalização são fundamentais para construir ou mesmo refinar o modelo de dados. Entretanto, na prática, existem algumas honrosas exceções em que se justifica a violação de alguma dessas regras: por exemplo, para melhoria do desempenho na recuperação de um dado, ou quando se deseja manter informações históricas, portanto estáticas, que perdem a integridade referencial, podendo transferir atributos referenciados em outra tabela como sendo atributos da própria tabela.

- Modelo Relacional

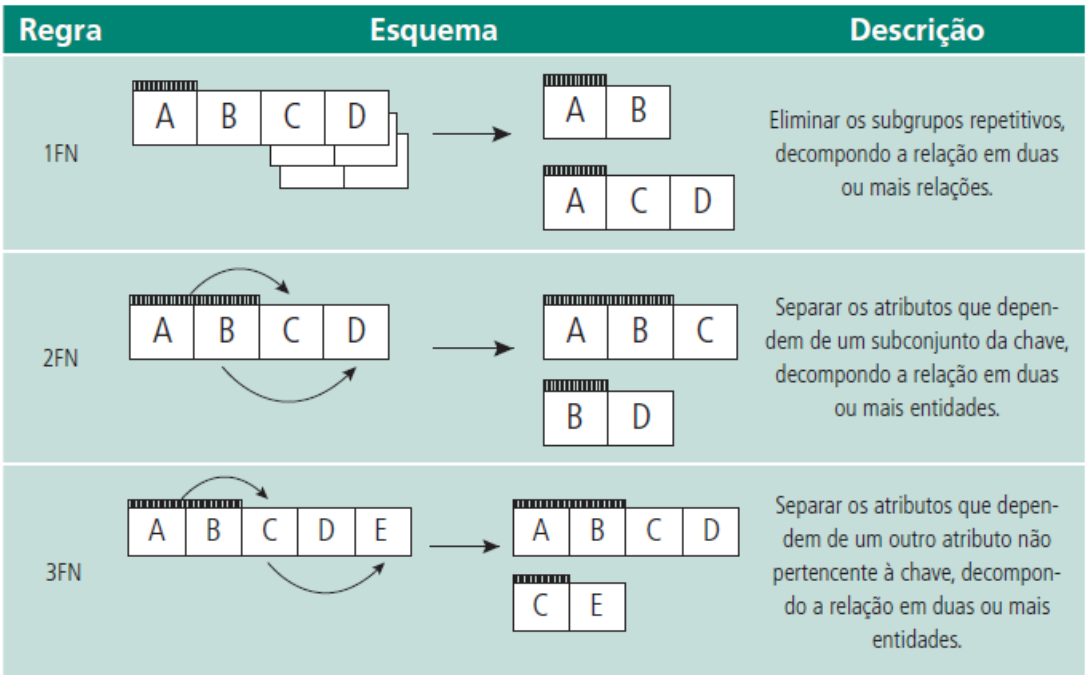

Figura 1.10: Resumo adaptado das três formas normais

- Normalização
- Diagramas E-R e Tabelas Originadas

Vejamos um outro exemplo do uso da 3FN

Normalmente, um bloco de pedidos em uma loja qualquer tem um formulário em duas vias onde se deve preencher à mão os dados do pedido, do cliente, da forma de pagamento, condições de entrega, itens relacionados entre outros. Para simplificar, estamos representando apenas parte desses dados na tabela de cabeçalho do pedido (CabecPedido) conforme mostrado na Figura 1.11 a seguir.

Nro_Pedido	Dat_Pedido	Nom_Cliente	Des_Endereco	Nro_Telefone
00001	05/01/2011	José	R. Itu, 100	3222-2222
00002	05/01/2011	Antônio	R. Sabará, 322	3111-3333
00003	06/01/2011	José	R. Itu, 100	3222-2222
00004	06/01/2011	Antônio	R. Sabará, 322	3111-3333

Figura 1.11: Tabela de CabecPedido

Como podemos perceber, um mesmo cliente pode fazer vários pedidos; isto faz com que seus dados sejam repetidos em diferentes folhas do bloco de pedidos. Mas ao projetarmos o modelo de um banco de dados capaz de persistir as informações do pedido, ele não terá exatamente a mesma estrutura em que se encontra no mundo real. Teremos de fazer uma decomposição sem perdas de dados de tal forma a evitar anomalias, tais como: se o cliente fizer um novo pedido, tenho que lhe pedir novamente seus dados

de cadastro? Se o cliente pedir para cancelar todos os seus pedidos, tenho que pedir que ele informe o número de cada pedido? Ou senão? Tenho que pesquisar em todos os pedidos quem tem o nome José? Poderá haver outros clientes com o mesmo nome? E se ao cadastrar o cliente forem utilizadas grafias diferentes: Jose, JOSE, Jozé, etc. Problemas semelhantes podem ocorrer caso o cliente solicite para alterar o endereço de todos os seus pedidos.

Como podemos ver, se fizermos uma conversão direta do bloco de pedidos em um modelo de banco de dados, diversas anomalias ocorrerão no modelo proposto. Como um cliente pode fazer vários pedidos, os atributos de cliente não dependem funcionalmente na chave primária de CabecPedido. Os atributos de cliente devem ser separados em uma nova relação. Para tal, devemos identificar ou criar entre os atributos de cliente quais são o(s) conjunto(s) de atributos que podem ser considerados como chaves candidatas. E entre as chaves candidatas, qual é a mais indicada para ser chave primária. Neste ponto, cabe um debate sobre o assunto: Qual é a forma correta de escolher a chave primária? Resposta: depende. Sempre quando estamos definindo um modelo de banco de dados, estamos fazendo uma representação do minimundo. Portanto, nem sempre é possível, ao cadastrar um cliente, obtermos todos os seus dados. Suponha que o cliente está comprando em dinheiro e pagou à vista; ele pode não querer fornecer seu número de CPF ou sua carteira de identidade. Se um desses campos for chave primária da tabela, como cadastrar os diferentes clientes que tenham essa mesma conduta. Nesse caso, não seria melhor pesquisar o cliente pelo nome, buscando identificar se ele já está cadastrado? Dessa forma, talvez a melhor saída seja a geração de uma chave primária automática "burra" sem nenhum significado semântico. Apenas um número sequencial. Os demais atributos, chaves candidatas: CPF ou carteira de identidade poderiam ser atributos que aceitem valores nulos. Desse modo, torna-se possível fazer um cadastramento do cliente de forma incompleta para uma venda à vista. Para os casos de venda a prazo, seria possível criar uma regra de negócio que verifique se o cadastro do cliente está completo, não permitindo a realização da venda sem o preenchimento completo e verificação de aprovação de crédito do cliente.

1.3.7 Padronização

Ao iniciar um processo de desenvolvimento de um sistema, assim como qualquer outra atividade na qual se pretende chegar a um produto final, é de fundamental importância estabelecer previamente regras de como chegar lá. Qual será o processo utilizado? Quais serão as atividades dentro do processo? Enfim, qual será a metodologia, composta de quais artefatos, gerando quais produtos intermediários, para possibilitar a validação das várias fases do processo de desenvolvimento. Todo esse questionamento nos remete à área da Engenharia de Software, sobre a qual na verdade não é o objetivo deste texto discorrer.

Mas, independentemente da metodologia utilizada, o que importa nesse contexto é que todos dados identificados na etapa de levantamento do problema, que devam ser persistidos no banco de dados, sejam definidos a partir de um padrão de nomes.

Em um SGBD, ao criarmos um banco de dados, será necessário atribuir a ele um nome. O banco de dados, depois de criado, possibilita definir quais tabelas ele conterá. Cada tabela deve possuir um nome único dentro de um mesmo banco de dados. Uma tabela é formada por um conjunto de atributos (partes indivisíveis de informação). Cada atributo deverá ter um nome único dentro de uma mesma tabela. Diversas regras semânticas poderão ser definidas no banco de dados, tais como: chaves primárias, visões, restrição de domínio, etc. Cada uma dessas restrições, ao ser definida, recebe um nome que a identifique de forma unívoca. Enfim! Esses assuntos serão mais bem detalhados no decorrer do texto.

Na verdade, o que importa neste momento é dizer que mesmo na fase do projeto conceitual é importante definir nomes de tabelas e atributos seguindo alguma regra de padronização. Isto irá ajudar muito na organização e principalmente no mapeamento do modelo conceitual para o esquema físico do banco de dados, fazendo com que os ajustes da sua estrutura física sejam mínimos em relação ao projeto lógico.

Após defendermos a bandeira da padronização, afinal, qual deve ser padrão? Na verdade, a resposta para esta pergunta é que não existe uma regra única e obrigatória. Normalmente, cada grupo de desenvolvedores dentro de uma empresa normatiza e escreve as suas próprias regras. A título de exemplo, iremos apresentar algumas sugestões de padronização.

Mesmo apesar de alguns SGBDs aceitarem características da língua portuguesa, tais como: acentos, cedilha e espaços em branco. Essas situações devem ser evitadas na criação de nomes de tabelas, atributos, etc. Agindo dessa forma você estará evitando possíveis problemas de incompatibilidades. Além disso, um detalhe importante é atentarmos para as limitações de nomenclatura do próprio SGBD que será utilizado como repositório. Ao propor uma regra o importante é utilizar o bom senso. Afinal! E o que é bom senso? Vamos apresentar nossa visão.

Nome de entidade/tabela

- Deve-se evitar uso de nomes codificados, como por exemplo, S01010, pois, ao olharmos para o nome, ele não tem nenhum significado.
- Os nomes devem ter um limite de tamanho; normalmente 18 caracteres é o máximo.
- Deve-se procurar utilizar nomes que traduzam de forma clara o sentido da entidade, normalmente no singular. Exemplo: cliente, aluno, etc.
- Deve-se evitar vincular ao nome de uma tabela a associação a um determinado sistema. Pode

ocorrer que, em uma nova versão do sistema, a mesma tabela passe a ser utilizada por dois ou mais sistemas. Nesse caso, seria necessário trocar o seu nome, ou mantê-la no banco de dados com um nome que não expressa a realidade. Exemplo: o sistema de contabilidade tem um prefixo "contab"; então, usamos para nomear as tabelas este prefixo: contab-plano-conta.

- Uma possível solução para reduzir o tamanho do nome é criar um mnemônico com um tamanho fixo de caracteres, por exemplo, seis caracteres, sendo a formação do nome composta pelas iniciais do nome por extenso. Exemplo: fornecedor = fornec; agência bancária = ageban, etc. Nesse caso foi criada uma codificação, mas que mantém ainda assim certo teor semântico.

Nome de atributos

- Deve-se evitar uso de nomes codificados. Também se deve evitar usar o nome da tabela como parte do nome do atributo ex.: S01-Cod, Cliente-Cod, Cliente-Nome.
- Os nomes devem ter um limite de tamanho; normalmente 18 caracteres é o máximo.
- Deve-se dividir o nome do atributo em pelo menos duas partes. Um prefixo identificador, para identificar o tipo do atributo, seguido de um underscore (caracter de sublinhado) e uma sequência de caracteres para qualificar o atributo. Veja a Figura 1.12 com os identificadores sugeridos.

Quadro 1.1: Sugestão de identificadores dos atributos

Identificador	Descrição
Aut	Para identificar atributos que são autoincrementados. Normalmente é uma sequência numérica que funciona como chaves primárias na relação. Exemplo: Aut_Pedido; Aut_OS.
Cod	Para identificar atributos que são codificados. Normalmente são chaves primárias na relação. Exemplo: Cod_Disciplina; Cod_Depto.
Dat	Para identificar atributos que representam datas. Exemplo: Dat_Admissao.
Des	Para identificar descrição. Exemplo: Des_Peça; Des_Serviço.
Hor	Para representar um identificador horas. Exemplo: Hor_Inicio.
Idt	Para representar um identificador que possui restrição de domínio de valores. Exemplo: Idt_Situacao; Idt_Sexo.Possui como valores válidos "M" ou "F".
Nro	Para identificar campos numéricos. Exemplo: Nro_Filhos.
Nom	Para identificar nomes. Exemplo: Nom_Cliente; Nom_Cidade.
Txt	Para identificar um texto. Exemplo: Txt_Aviso_Cobranca.
Qtd	Para identificar quantidade. Exemplo: Qtd_Vendida.
Vlr	Para identificar valor ou preço. Exemplo: Vlr_unitario.

Figura 1.12: Sugestão de identificadores dos atributos

1.4 Considerações Finais

Nesta aula discutimos a importância que os dados têm para um sistema de informação. Na verdade, os sistemas de informação são criados para melhor lidar com os dados existentes nos processo de negócio

das organizações. Nesse sentido foram apresentados os conceitos de entidades e de relacionamentos que podem ser representados através de um modelo DER que expressa como os dados se relacionam no “minimundo” que desejamos modelar. Além disso, foram apresentados aspectos de normalização dos dados a fim de minimizar a redundância e a inconsistência dos dados. Finalizamos apresentando a importância da padronização dos nomes das entidades e atributos.

Bibliografia

Livros

[1] William Pereira Alves. *Banco de dados: teoria e desenvolvimento*. Saraiva Educação SA, 2009.

[2] Felipe Nery Rodrigues Machado e Mauricio Pereira de Abreu. *Projeto de banco de dados: uma visão prática*. Saraiva Educação SA, 2004.

Índice