Projeto lógico de bancos de dados

Apostila elaborada pelo Professor Antonio Cesar de Barros Munari
FEV/1998 Revisada: MAR/2002

Índice

Introdução 1

Conceitos básicos dos bancos de dados relacionais 2

- Regras de integridade fundamentais no modelo relacional 7
- Outras regras de integridade 8

Mapeamento E/R para bancos de dados relacionais 10

- Atribuição de nomes 10
- Tipos de dados 11
- Regras básicas de mapeamento 12
- Análise de Relacionamentos 14

Normalização de tabelas 18

- Primeira forma normal 18
- Segunda forma normal 19
- Terceira forma normal 20
- Roteiro para normalização de dados 21

Referências bibliográficas 22

Introdução

O projeto lógico constitui a fase imediatamente posterior à modelagem conceitual do banco de dados, e procura detalhar a forma como as entidades, atributos e relacionamentos serão representados no banco de dados. É, por esse motivo, dependente da tecnologia que será utilizada, ao contrário do que ocorre com a modelagem conceitual. Assim, é importante saber se o banco de dados será criado, por exemplo, usando a abordagem relacional, objeto-relacional ou orientada a objetos e, num segundo momento, também qual seria exatamente o produto SGBD a ser adotado, já que uma grande parte do detalhamento depende de características que costumam variar entre um e outro SGBD. Existe ainda uma terceira fase, teoricamente subseqüente ao projeto lógico, denominada projeto físico, onde há o detalhamento dos dispositivos e recursos físicos a serem alocados para suportar o conteúdo proposto para o banco de dados, tais como índices, clusters, discos, etc. Esse projeto físico, entretanto, muitas vezes sobrepõe-se parcialmente ao projeto lógico (principalmente devido às características e recursos oferecidos por algumas ferramentas CASE utilizadas para o projeto lógico, que freqüentemente misturam elementos das duas fases), e em parte com a fase de programação e de ajuste de desempenho, devido à própria natureza iterativa do processo de desenvolvimento. Em geral, considera-se que um bom projeto lógico, no nível das definições que serão feitas a seguir, é suficiente para a geração e documentação de bons esquemas de bancos de dados.

Apesar de a tecnologia orientada a objeto ser alvo de intensa pesquisa e divulgação nos últimos tempos, é fato que a esmagadora maioria das implementações de bancos de dados ainda utiliza a abordagem relacional, que por esse motivo será a adotada neste material. Isso significa que o projeto lógico passará a ser então, para toda a discussão subseqüente, o mapeamento de entidades e relacionamentos para tabelas e chaves estrangeiras e de atributos para colunas.

Conceitos básicos dos bancos de dados relacionais

Em um ambiente de banco de dados relacional utilizamos alguns conceitos muito importantes, imprescindíveis para a correta implantação e operação de qualquer sistema de banco de dados. Alguns desses novos termos originaram-se diretamente da teoria de Conjuntos, outros são decorrentes da utilização de elos lógicos para implementar os relacionamentos entre os dados armazenados no banco de dados. A seguir será apresentada uma rápida visão desses conceitos fundamentais.

Tabela: A estrutura que armazena os dados referentes a cada uma das ocorrências de uma entidade ou relacionamento com atributos é uma tabela ou seja, a tabela é a materialização para o SGBD desses objetos conceituais. Assim, ao projetarmos um banco de dados relacional a partir de um modelo Entidade / Relacionamento criamos uma tabela para cada entidade ou relacionamento com atributos. As tabelas são também chamadas de relações (no sentido proposto pela Teoria dos Conjuntos). Uma tabela é composta por colunas e linhas (ou tuplas, segundo a Teoria dos Conjuntos), conforme ilustra o exemplo a seguir.

Nome da tabela: FUNCIONARIO

Linha

NrMatric	NmFunc	DtAdm	VrSalario	CdCargo	CdDepto
1001	JOAO SAMPAIO	10/08/93	750	C2	D2
1004	LUCIO TORRES	02/03/94	750	C2	D2
1034	ROBERTO PEREIRA	23/05/92	450	C3	D1
1021	JOSE NOGUEIRA	10/11/94	450	C3	D1
1029	RUTH DE SOUZA	05/01/92	350	C1	D3
1095	MARIA DA SILVA	03/09/92	600	C4	D1
1023	LUIZ DE ALMEIDA	12/01/93	750	C2	D2
1042	PEDRO PINHEIRO	29/07/94	600	C4	D1
1048	ANA SILVEIRA	01/06/93	2300	C5	D1
1015	PAULO RODRIGUES	17/08/92	750	C2	D2

Coluna

Figura 1: Conceitos de tabela, linha e coluna.

Esses conceitos básicos são freqüentemente indicados por mais de uma termo, devido tanto a razões conceituais como culturais e históricas, e muitas vezes acaba havendo alguma confusão com relação a terminologia adotada. A tabela a seguir, extraida de [MELO et al.,

1997], procura resumir a equivalência dos principais termos utilizados para retratar esses conceitos em diversas situações.

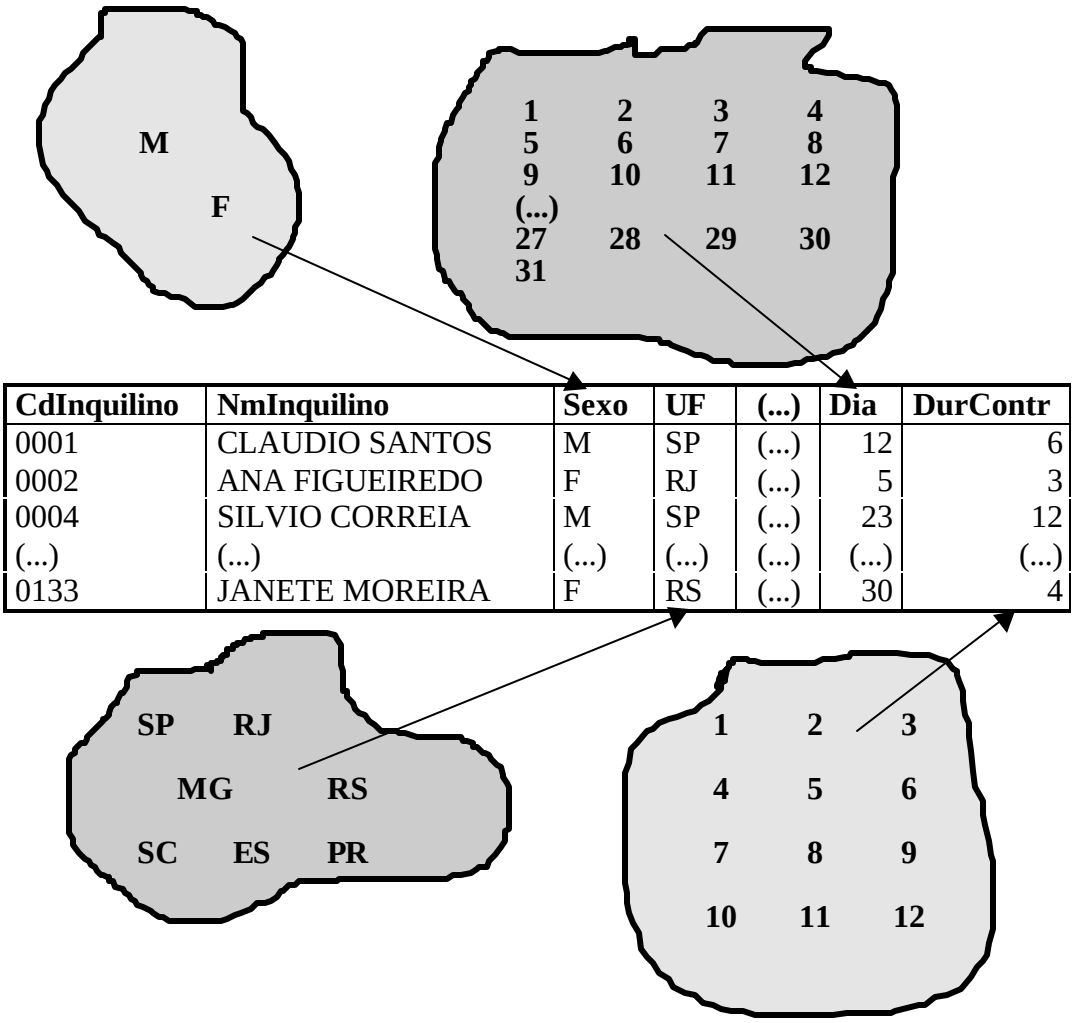
Contexto	***Terminologia***		
Modelo Relacional (formal)	Relação	Tupla	Atributo
Modelo Relacional (informal) e SQL	Tabela	Linha	Coluna
Teoria de conjuntos e Estrutura de dados	Conjunto	Elemento estruturado (lista)	Componente de lista
Análise estruturada moderna	Depósito de Dados	Registro	Campo
Processamento de dados	Arquivo	Registro	Campo

Tabela 1: Terminologia de conceitos básicos do Modelo Relacional em diversos contextos

Linha: Uma ocorrência em particular de dados em uma tabela ocupa uma linha dessa tabela. No exemplo anterior, os dados de um empregado ocupam uma linha na tabela Funcionario. Como existem 10 empregados nessa tabela, ela possui 10 linhas (ou tuplas, ou registros).

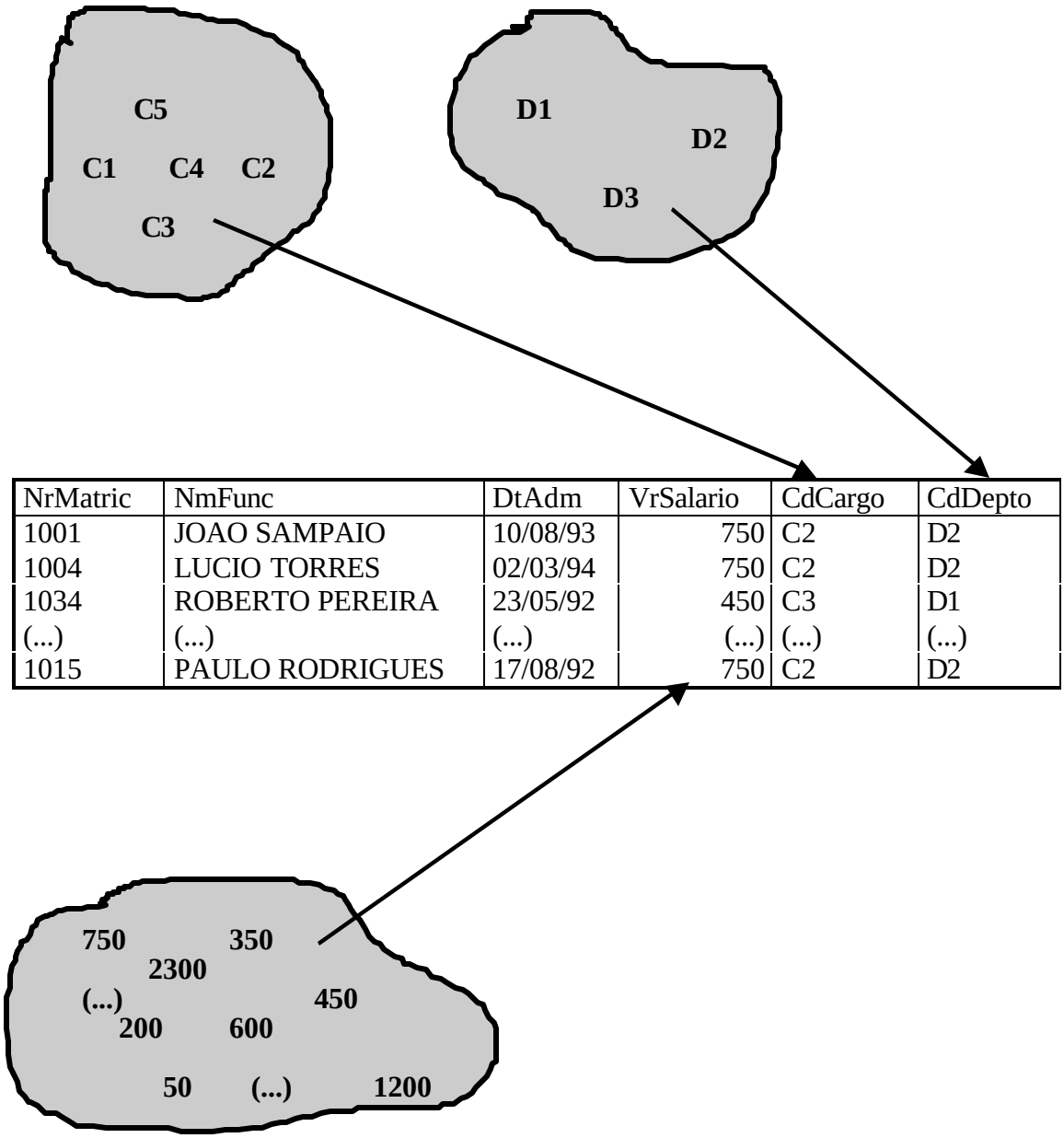
Coluna: Cada tipo de informação armazenada em uma tabela é uma coluna. Toda coluna de uma tabela deve possuir um nome pelo qual será referenciada sempre que necessário. Ao criarmos uma tabela definimos, para cada uma de suas colunas, o seu nome e também o seu tipo (numérico, alfabético, data, etc). Por exemplo, NmFunc e DtAdm são colunas (ou campos) da tabela Funcionario. Essa tabela possui um total de 6 colunas em sua estrutura.

Domínio: Sempre que identificamos um atributo de uma entidade, temos também uma idéia de qual o tipo de informação que ele poderá vir a conter. Por exemplo, se reconhecemos Sexo como sendo um atributo de uma entidade qualquer, sabemos de antemão que a informação ali contida será, sempre, "Masculino" ou "Feminino". De forma semelhante, se um atributo for Estado, com certeza seu conteúdo será a sigla ou o nome de um dos Estados do Brasil. Chamamos a essa faixa de valores que um atributo pode conter de domínio desse atributo. Assim, para o atributo Dia do Mês, o domínio será o conjunto de números entre 1 e 31; o domínio de um atributo Salário será o conjunto de números entre 0 e o teto salarial estipulado pela empresa, e assim por diante.



CdInquilino	NmInquilino	Sexo	UF	(...)	Dia	DurContr
0001	CLAUDIO SANTOS	M	SP	(...)	12	6
0002	ANA FIGUEIREDO	F	RJ	(...)	5	3
0004	SILVIO CORREIA	M	SP	(...)	23	12
(...)	(...)	(...)	(...)	(...)	(...)	(...)
0133	JANETE MOREIRA	F	RS	(...)	30	4

Figura 2: Exemplos de domínios simples (discretos).



NrMatric	NmFunc	DtAdm	VrSalario	CdCargo	CdDepto
1001	JOAO SAMPAIO	10/08/93	750	C2	D2
1004	LUCIO TORRES	02/03/94	750	C2	D2
1034	ROBERTO PEREIRA	23/05/92	450	C3	D1
(...)	(...)	(...)	(...)	(...)	(...)
1015	PAULO RODRIGUES	17/08/92	750	C2	D2

Figura 3: Outros exemplos de domínios (contínuos e discretos).

Chave primária: Um dos princípios do modelo relacional diz que uma linha de uma tabela deve sempre poder ser referenciada de forma única. O atributo diferenciador (que não se repete) entre as linhas de uma tabela é chamado de chave primária, que no caso da tabela Funcionario, por exemplo, poderia ser a coluna NrMatric, já que essa informação não se repete para dois funcionários distintos.

Muitas vezes, porém, uma entidade não possui, entre seus atributos, um que por si só seja suficiente para identificar univocamente uma ocorrência. Nesses casos deve sempre ser possível que a combinação de dois ou mais atributos tenha a capacidade de se constituir numa chave primária. Chamamos a essas chaves primárias formadas pela combinação de vários atributos de **chaves primárias compostas**.

Chave estrangeira: Os relacionamentos entre tabelas são implementados no modelo relacional através de atributos de relacionamento. Essa técnica consiste na distribuição de alguns atributos-chave pelas tabelas que representam as entidades envolvidas no relacionamento, de forma que seja possível a associação lógica entre as linhas das tabelas com base na comparação de suas chaves. Na prática uma chave estrangeira é uma coluna (ou combinação de colunas) que indica um valor que deve existir como chave primária numa outra tabela (chamada de Tabela Pai). No caso da tabela Funcionário existem duas chaves estrangeiras: as colunas CdCargo e CdDepto que referenciam, respectivamente, as tabelas de Cargos e de Departamentos.

NrMatric	NmFunc	DtAdm	VrSalario	CdCargo	CdDepto
1001	JOAO SAMPAIO	10/08/1993	750	C2	D2
1004	LUCIO TORRES	02/03/1994	750	C2	D2
1034	ROBERTO PEREIRA	23/05/1992	450	C3	D1
1021	JOSE NOGUEIRA	10/11/1994	450	C3	D1
1029	RUTH DE SOUZA	05/01/1992	350	C1	D3
1095	MARIA DA SILVA	03/09/1992	600	C4	D1
1023	LUIZ DE ALMEIDA	12/01/1993	750	C2	D2
1042	PEDRO PINHEIRO	29/01/1994	600	C4	D1
1048	ANA SILVEIRA	01/06/1993	2300	C5	D1
1015	PAULO RODRIGUES	17/08/1992	750	C2	D2

CdCargo	NmCargo
C1	COZINHEIRA
C3	AUX. ESCRITORIO
C7	VIGIA
C2	MECANICO
C5	GERENTE
C4	ESCRITURARIO

CdDepto	NmDepto
D1	ADMINISTRACAO
D2	OFICINA
D3	SERVICOS GERAIS
D4	VENDAS

Figura 4: Exemplo de chaves estrangeiras para a tabela de funcionários.

Nulo: Representa um valor desconhecido para um dado. Não devemos confundir o conceito de nulo com espaços em branco ou o número zero, por exemplo, que são valores conhecidos. Nulo é a ausência de informação. Uma coluna de preenchimento obrigatório numa tabela deve possuir todos os seus valores não-nulos. Se, por exemplo, uma linha da tabela Funcionário contiver um nulo na coluna DtAdm, significa que a data de admissão do funcionário correspondente àquela linha é desconhecida.

NrMatric	NmAluno	(...)	Sexo	RG	Reservista
22941010	CLAUDIO SILVA	(...)	M	21212121	91929
22941012	ANA RIBEIRO	(...)	F	14141414	?! ?! ?! ?!
(...)	(...)	(...)	(...)	(...)	(...)
22941088	JOSE SANTOS	(...)	M	34343434	828122

Figura 5: Exemplo de atributo com valor nulo permitido (“Reservista”).

Regras de integridade fundamentais no modelo relacional

O modelo relacional, ao definir conceitos como Tabela, Coluna, Nulo, Domínio, Chave Primária e Chave Estrangeira, deixa implícitas algumas regras fundamentais para a manutenção da consistência da base de dados. Elas são chamadas de Regras de Integridade e tratam dos cuidados que analistas, projetistas e programadores devem observar ao implementar as rotinas de Inclusão, Alteração e Exclusão de dados nos objetos. Consideramos que existem dois principais tipos de integridade a serem mantidas numa base de dados relacional adequadamente projetada: a Integridade de Entidade e a Integridade Referencial.

Integridade de Entidade (ou de Identidade ou Existencial): Refere-se às chaves primárias e procura garantir que toda e qualquer linha de uma tabela deve poder ser acessada com base apenas no conteúdo de sua chave primária.

Regra básica:

> *"Nenhum atributo que faça parte de uma chave primária pode ter valor nulo."*

Isso significa que os conteúdos de todos os atributos que constituem uma chave primária devem ser conhecidos. Um conteúdo nulo representa uma informação desconhecida ou, em outras palavras, a ausência da informação, o que não pode ser permitido em qualquer elemento de uma chave primária.

Adicionalmente lembramos que, por definição, não se deve permitir que numa mesma tabela existam duas ocorrências com chaves primárias iguais.

Integridade Referencial: Diz respeito às chaves estrangeiras e visa manter a integridade dos relacionamentos previstos na base de dados. Na definição dos cuidados referentes a esse tipo de integridade, utilizaremos dois conceitos novos:

> *Tabela-Pai (Parent Table)*: é aquela onde o atributo de relacionamento desempenha o papel de chave primária.

Tabela-Filho (Dependent Table): tabela onde o atributo de relacionamento desempenha o papel de chave estrangeira.

Regra básica:

"O conteúdo de uma chave estrangeira deve, necessariamente, ser igual ao de uma ocorrência da Tabela-Pai ou então ser nulo."

O primeiro caso mencionado pela regra significa que a ocorrência da Tabela-Filho possui uma correspondente na Tabela-Pai. Já o segundo caso representa a ausência da correspondência: para aquela ocorrência em particular, não existe o relacionamento.

As conseqüências da Integridade Referencial refletem-se nas consistências necessárias ao se proceder às operações de Inclusão, Alteração e Exclusão de dados nas Tabelas Pai e Filho. É isso o que mostra o quadro a seguir.

Operação	***Tabela-Pai***	***Tabela-Filho***
Inclusão	Não afeta.	Não pode incluir se não tiver na Tabela-Pai
Alteração	Se envolver a chave primária, utilizar um dos seguintes critérios: não alterar se tiver dependentes; alterar e colocar nulos nas chaves estrangeiras de todos os dependentes da Tabela-Filho; alterar e colocar o novo valor nas chaves estrangeiras de todos os dependentes na Tabela-Filho.	Se envolver a chave estrangeira, só aceitar valores que existam na Tabela-Pai
Exclusão	Utilizar um destes 3 critérios: não deletar se tiver dependentes; deletar e colocar nulos nas chaves estrangeiras da Tabela-Filho; deletar e também eliminar todos os dependentes.	Não afeta.

Tabela 2: Implicações da integridade referencial nas operações de gravação em tabelas.

Outras regras de integridade

Além das regras de integridade de entidade e referencial, um banco de dados relacional pode suportar um conjunto adicional de regras (ou restrições) cuja finalidade é especificar aspectos próprios de cada coluna e respectivo domínio, complementando com isso a definição de suas características lógicas. As principais restrições de integridade complementares tratam sobre a obrigatoriedade e unicidade de valores e sobre conjuntos de valores permitido.

Obrigatoriedade: indica se deve ou não ser permitida a existência de nulos numa coluna. Colunas que não aceitam nulos são então de preenchimento obrigatório como, por exemplo, o nome de um funcionário na tabela de funcionários da figura 4, enquanto que a coluna Reservista é não obrigatória na tabela de alunos da figura 5.

Unicidade: indica se deve ser permitido ou não que uma coluna possua valores idênticos em duas ou mais linhas. Uma coluna que não pode possuir valores repetidos é uma coluna de valores únicos. Um exemplo de coluna de valores únicos é a coluna RG na tabela de alunos da figura 5 enquanto que, nessa mesma tabela, a coluna Sexo não é de valores únicos, pois deve aceitar valores repetidos.

Verificação de valores específicos: indica explicitamente qual o conjunto de valores permitidos para uma coluna. Por exemplo, para a coluna Sexo da tabela de alunos da figura 5, os valores permitidos são { ‘M’, ‘F’}.

Mapeamento E/R para bancos de dados relacionais

O conjunto de objetos de banco de dados que implementam um modelo de dados específico é chamado de esquema do banco de dados. Em princípio ele é composto pelas tabelas que deverão armazenar as informações necessárias, mas ao final do projeto, objetos de outros tipos acabam também sendo necessários, como índices, regras e subrotinas, por exemplo. As recomendações apresentadas a seguir procuram organizar a abordagem que o projetista de um banco de dados precisa adotar ao desenvolver um projeto lógico.

Atribuição de nomes

As tabelas e colunas que constituirão o banco de dados devem receber nomes próprios, que precisam seguir as regras de formação de nomes específicas de cada produto, e que algumas vezes são bastante restritivas, conforme indica o quadro apresentado a seguir. Isso significa que nem sempre será possível adotar no esquema das tabelas os mesmos nomes utilizados no modelo E/R. Além disso, há também pelo menos dois outros motivos para não se repetir cegamente no esquema do banco de dados os mesmos nomes do modelo conceitual: com a mudança do nível de abstração empregado, há uma correspondente (mesmo que ligeira) modificação dos significados de alguns elementos, que acabam inevitavelmente requerendo adaptações ou, muitas vezes, levam ao desmembramento de um atributo em mais de uma coluna e, em segundo lugar, é inegável que nomes curtos são sempre mais práticos que nomes longos, principalmente quando se trata de programação. Um clássico exemplo de variação na nomenclatura entre o modelo conceitual e o modelo lógico é o dos nomes das tabelas que, em geral, são substantivos no plural ao contrário do que ocorre com as entidades, costumeiramente batizadas por meio de substantivos no singular.A preocupação com o emprego de bons critérios para a atribuição de nomes deve ser permanente, levando à documentação das convenções adotadas, principalmente no que se refere a padrões de abreviatura, grafia e regras para a geração de nomes compostos. Num outro plano, todo o esquema lógico deve ser documentado num dicionário de dados, indicando o nome de cada objeto do banco de dados, sua finalidade e estrutura, pelo menos.

Padrão / *Aspecto*	*SQL Ansi*	*DBF*	*Paradox*	*MS-Access*	*Oracle*	*MS-SQL Server*	*Interbase*	*MySQL*
	92	***III+***	***7***	***2000***	***8***	***7***	***4.2***	***3.23***
Tam. nome tabela	128	***	***	64	30	128	31	64
Tam. nome coluna	128	10	25	64	30	128	31	64
Caracteres permitidos	Letras, números e underscore.	Letras, números e alguns caracteres especiais. P/ tabelas: ***	P/ colunas, tudo exceto: vírgula; pipe; exclamação. P/ tabelas: ***	Tudo exceto: ponto dec; exclamação; acento grave; colchetes.	Letras, números, underscore, cifrão e number (#).	Letras, números, underscore, cifrão e number (#).	Letras, números, underscore e cifrão.	Tudo, exceto: ponto dec. P/ tabelas tb não permite usar contra-barras.
Obs.	Primeiro caracter deve ser uma letra.	Espaços em branco e pontuações não são permitidos	Primeiro caracter não pode ser um espaço	Primeiro caracter não pode ser um espaço	Primeiro caracter deve ser uma letra.	Primeiro caracter deve ser letra ou underscore.	Primeiro caracter deve ser uma letra.	Nomes não podem ser compostos apenas por números.

*** Varia conforme o sistema de arquivos do sistema operacional

Tabela 3: Regras de nomes em alguns SGBDs relacionais.

Tipos de dados

O detalhamento de uma tabela deve apresentar todas as colunas, juntamente com seu tipo de dados e, quando necessário, o tamanho correspondente. Novamente, as diferenças entre cada SGBD são marcantes, conforme mostra o quadro a seguir, que reúne os tipos de dados de uso mais comum em uma variedade de produtos de banco de dados.

Padrão / *Tipo*	*SQL Ansi* 92	*DBF* III+	*Paradox* 7	*MS-Access* 2000	*Oracle* 8	*MS-SQL Server* 7	*Interbase* 4.2	*MySQL* 3.23
Texto	Char Varchar	Character	Alpha	Text Memo	Char Varchar2 Long	Char Varchar	Char Varchar	Char Varchar
Data	Date	Date	Date	Datetime	Date	Datetime	Date	Datetime
Hora	Time	-	Time	Datetime	Date	Datetime	Date	Datetime
Número	Numeric Integer Float	Numeric	- Long Integer Number	- Long Double	Number	Numeric Int Real	Numeric Integer Float	Numeric Integer Double
Moeda	-	-	Money	Currency	-	Money	-	-
Lógico	-	Logic	Logical	Bit	-	Bit	-	-
Contador	-	-	Autoincrement	Counter	-	-	-	-
Binários	-	-	Bynary	LongBinary	BLOB	Varbinary	BLOB	BLOB

Tabela 4: Principais tipos de dados suportados por alguns SGBDs relacionais.

Exemplo:

Entidade

@ Cód. Prod.
Nome
Preço Unit.
Qtde Estoque
Data última compra

Tabela de produto no formato DBF

Coluna	*Tipo*	*Tamanho*
CdProd	Numeric	5, 0
NmProd	Character	30
VrUnit	Numeric	10, 2
QtEstoque	Numeric	8, 4
DtUltCompr	Date	-

Tabela de produto no formato Paradox

Coluna	*Tipo*	*Tamanho*
CdProd	Long Integer	-
NmProd	Alpha	30
VrUnit	Money	-
QtEstoque	Number	-
DtUltCompra	Date	-

Tabela de produto no Access

Coluna	***Tipo***	***Tamanho***
CdProd	Long	-
NmProd	Text	30
VrUnit	Currency	-
QtEstoque	Double	-
DtUltCompra	Datetime	-

Tabela de produto no Oracle

Coluna	***Tipo***	***Tamanho***
CdProd	Number	5, 0
NmProd	Varchar2	30
VrUnit	Number	10, 2
QtEstoque	Number	8, 4
DtUltCompra	Date	-

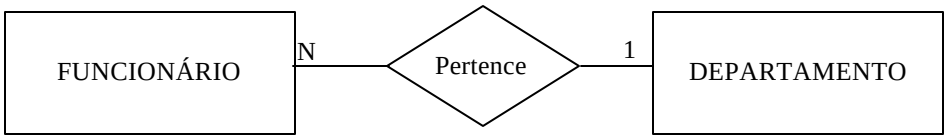
Tabela de produto no MS-SQL Server

Coluna	***Tipo***	***Tamanho***
CdProd	Numeric	5, 0
NmProd	Varchar	30
VrUnit	Money	-
QtEstoque	Numeric	8, 4
DtUltCompra	Datetime	-

Regras básicas de mapeamento

Qualquer componente do modelo conceitual que possua dados deve ser representado no banco de dados através de uma tabela. Isso significa que, em princípio, a cada entidade ou relacionamento com atributos próprios deve corresponder uma tabela do banco de dados. Geralmente representamos esse mapeamento através de algum tipo de diagrama onde aparecem todas as tabelas, seus relacionamentos e suas chaves primárias, pelo menos. Os relacionamentos entre as tabelas são implementados por meio de colunas especiais, chamadas *chaves estrangeiras*, que permitem vincular uma linha de uma tabela com a sua correspondente em outra tabela.

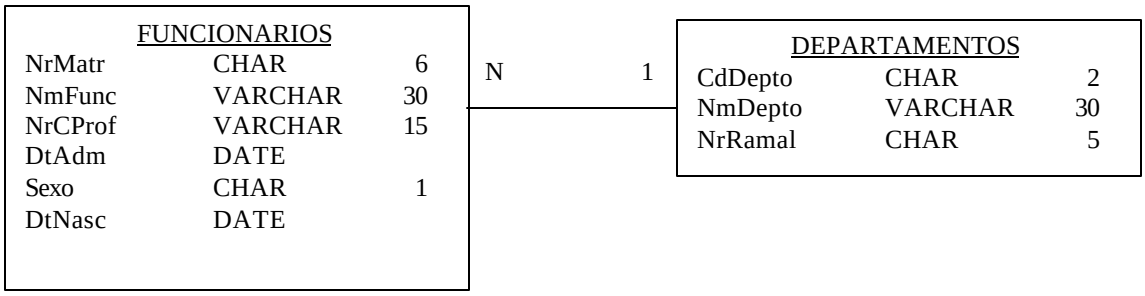
Exemplo: Vamos supor o seguinte diagrama E/R



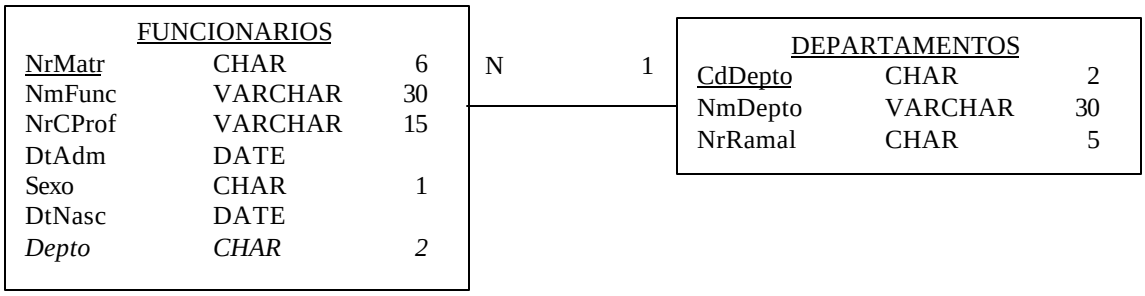
@ Número de Matrícula
Nome completo
Número da Carteira Prof.
Data de Admissão
Sexo
Data de Nascimento

@ Código do Depto
Nome completo
Ramal telefônico

Uma representação das tabelas para esse modelo poderia ser dada por:



A questão agora é determinar como vincular a linha de um funcionário com a linha correspondente ao seu respectivo departamento, e vice-versa. A solução é colocar na tabela Funcionarios uma nova coluna, do tipo CHAR com tamanho 2, destinada a conter o código do departamento ao qual o funcionário pertence. Com isso, para um funcionário qualquer, basta obter o valor contido nessa nova coluna e procurá-lo na coluna CdDepto da tabela Departamentos para saber o nome completo do departamento ou o seu ramal. Essa coluna nova criada para implementar essa amarração entre as tabelas é uma chave estrangeira. O desenho a seguir mostra como ficaria o esquema lógico das tabelas implementando a chave estrangeira para o relacionamento "Pertence":



As descrição completa da estrutura de cada tabela, indicando o nome, o tipo de dado e o tamanho para cada coluna deve ser documentada em um dicionário de dados à parte, se nenhuma ferramenta CASE estiver sendo utilizada no projeto. Esse dicionário pode conter informações adicionais sobre o significado de cada tabela, a indicação do(s) objeto(s) correspondente(s) no modelo conceitual e a descrição de características complementares importantes das colunas (como, por exemplo, se é algum tipo de chave ou possui alguma regra de validação que deve ser considerada para o dado). Também é conveniente que esse dicionário esteja adequadamente organizado de maneira a facilitar as consultas que se fizerem necessárias, e geralmente isso é obtido colocando-se a descrição das tabelas por ordem alfabética de nome.

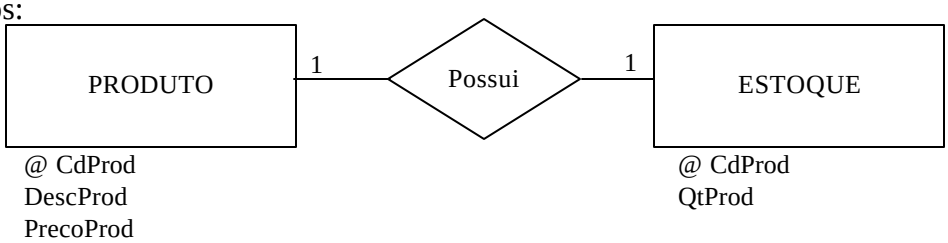
Análise de Relacionamentos

Um mesmo modelo conceitual (como o MER) pode dar origem a diversos modelos lógicos (relacionais e não-relacionais) distintos. Algumas entidades podem ser fundidas em uma única tabela; outras podem ser desmembradas em duas ou mais tabelas distintas, o mesmo ocorrendo com os atributos. Também os relacionamentos merecem alguma atenção quanto às várias possibilidades de implementação. A seguir, discute-se os principais aspectos dos relacionamentos que devem ser considerados ao se projetar as tabelas de um banco de dados relacional.

Relacionamentos 1 para 1

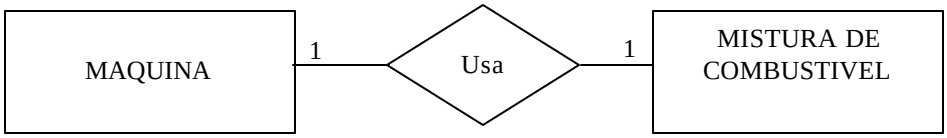
Sempre que é detectado um relacionamento de 1 para 1, a questão é saber se as duas entidades são realmente distintas ou se elas podem ser unidas em uma única tabela. Se possuem o mesmo atributo identificador, há uma forte razão para representar essas entidades em uma tabela só, a menos que o relacionamento seja opcional com cardinalidade mínima zero de um lado e um de outro lado.

Exemplos:



Neste diagrama, observamos que as entidades PRODUTO e ESTOQUE possuem seguramente o mesmo identificador, logo podemos representar as informações todas em uma só tabela (PRODUTOS, no caso).

Se tivermos, como no diagrama a seguir, uma entidade MAQUINA e outra MISTURA DE COMBUSTÍVEL, onde cada máquina tem sua própria mistura de combustível e cada mistura relaciona-se com uma e somente uma máquina, devemos considerar que a mistura de combustível utilizada pode ser representada como uma coluna da tabela MÁQUINAS

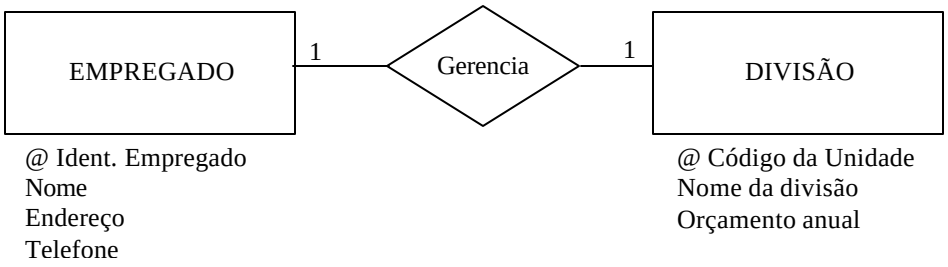


Dicas práticas:

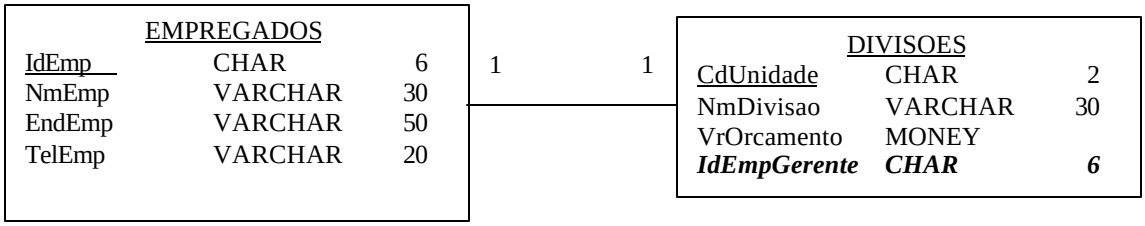
- Se as duas entidades forem realmente distintas, o relacionamento se dará através de um elo explícito, ou seja, uma chave estrangeira.
- Ao projetar, devemos analisar a possibilidade de o relacionamento 1 para 1 futuramente tornar-e um relacionamento 1 para muitos.

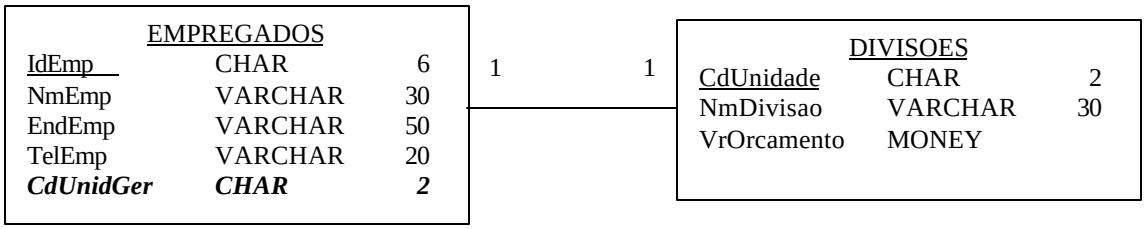
Exemplo:

Vamos supor um modelo retratando um relacionamento entre um funcionário e a divisão a que pertence conforme o diagrama a seguir.



As soluções possíveis de distribuição da chave estrangeira são as seguintes (indicadas em negrito itálico):





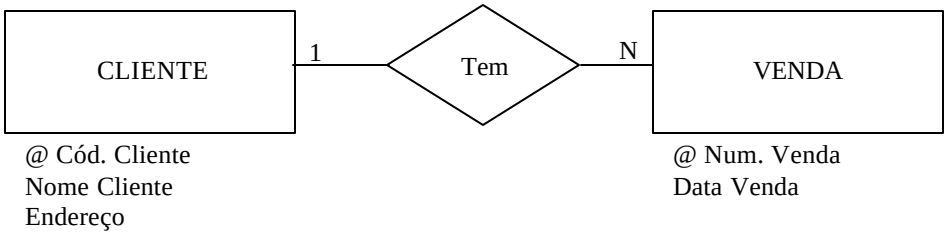
A estrutura de colunas proposta à tabela DIVISOES, na primeira solução, permite que no futuro um EMPREGADO possa vir a gerenciar mais de uma divisão.

Uma discussão mais detalhada sobre mapeamento de relacionamentos um para um pode ser encontrada em [SHOVAL, 1991].

Relacionamentos 1 para muitos

Este é um tipo de relacionamento mais fácil para análise. A chave estrangeira deve ser colocada na tabela do lado MUITOS do relacionamento e pode fazer parte da chave primária dessa tabela ou não.

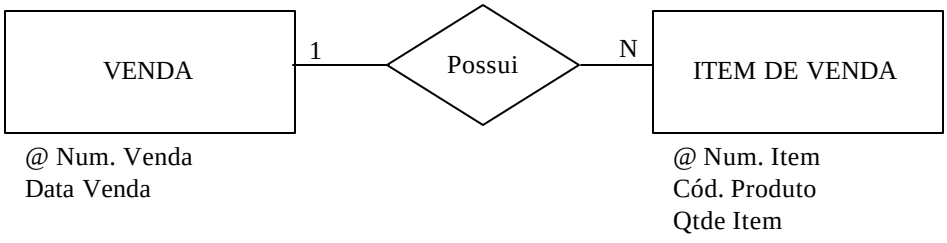
Exemplo:



CLIENTES		
CdCliente	NUMERIC	6,0
NmCliente	VARCHAR	30
EndCliente	VARCHAR	50

1 — N

VENDAS		
NrVenda	NUMERIC	6,0
DtVenda	DATE	
CdCliente	***NUMERIC***	***6,0***



VENDAS		
NrVenda	NUMERIC	6,0
DtVenda	DATE	
CdCliente	*NUMERIC*	*6,0*

1 — N

ITENSVENDA		
NrItem	NUMERIC	6,0
CdProd	CHAR	8
QtItem	NUMERIC	8,2
NrVenda	***NUMERIC***	***6,0***

Se considerarmos que um CLIENTE pode ter muitas vendas sendo feitas a ele, o código do cliente deverá fazer parte da tabela VENDAS, mas não necessariamente da chave, pois o número da venda identifica claramente a venda. Nesse exemplo um cliente está associado a "N" vendas e cada venda, a "N" ítens de venda.

Deve então o código do cliente ser incluído também na tabela ITEM DE VENDA?

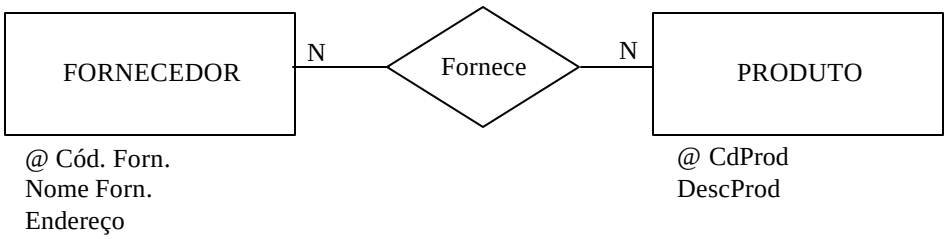
Obrigatoriamente não, pois se quisermos saber quais clientes pediram ou compraram um determinado produto, podemos pesquisar todas as ocorrências de itens de venda para o produto em pauta, obtendo o número de suas vendas para, posteriormente, pesquisar em VENDAS quais os clientes que as efetuaram.

Entretanto, é igualmente verdade que simplificaria bastante a programação e a pesquisa se colocássemos o código do cliente também em ITEM DE VENDA.

Relacionamentos muitos para muitos

Em todo relacionamento muitos para muitos a pergunta que se faz é: QUEM REFERENCIA QUEM?

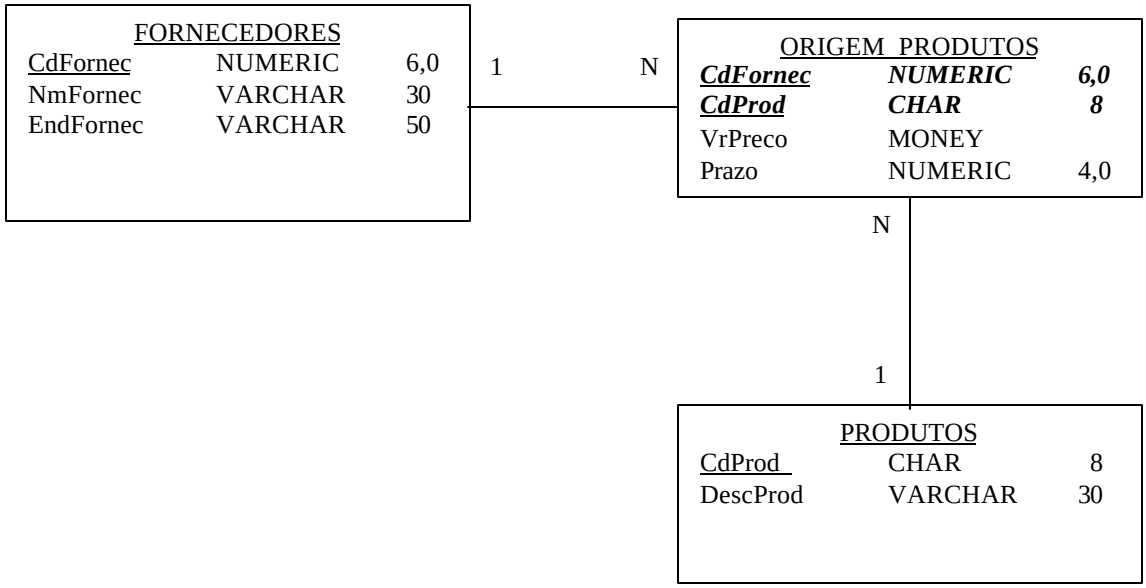
Este caso sempre pode ser resolvido com o desdobramento em 2 relacionamentos 1 para muitos e a criação de uma tabela intermediária que faça a associação entre as duas tabelas principais. A chave primária dessa nova tabela é, em princípio, a concatenação ou combinação das chaves primárias das duas tabelas principais.



Em nosso exemplo temos a situação em que FORNECEDOR fornece muitos PRODUTOS e qualquer PRODUTO poderá ser fornecido por vários fornecedores. Pergunta-se então:

- Qual tabela possuirá esta concatenação de chaves?
- Que colunas constituem essa tabela?
- Que elementos podem ser determinados se soubermos que estamos lidando com determinado PRODUTO de um FORNECEDOR específico? (Qual a razão desta ligação?)

Podemos dar então o nome de ORIGEM_PRODUTOS a essa tabela. Um produto poderá estar disponível em diferentes fornecedores, com preços e prazos diferenciados, e um fornecedor poderá ofertar vários produtos. Observe a chave primária composta nessa tabela, formada pela combinação das suas 2 chaves estrangeiras (as colunas CdFornec e CdProd).



Normalização de tabelas

A normalização de tabelas tem por objetivo principal resolver problemas de atualização de bases de dados, minimizando redundâncias. As principais características de uma base de dados normalizada são :

- geração de aplicações mais estáveis (e por isso também mais alteráveis) ;
- aumento do número de tabelas utilizadas ;
- diminuição dos tamanhos médios das tabelas.

O processo de normalização ocorre em etapas sucessivas. A cada etapa corresponde uma determinada forma normal, que representa um progressivo refinamento na estrutura das tabelas. Assim, uma tabela que se encontra na Terceira Forma Normal é considerada mais normalizada (mais "enxuta", pode-se dizer) que uma tabela que se encontra apenas na Segunda Forma Normal. Atualmente os teóricos da modelagem de dados consideram a existência de inúmeras formas normais, mas geralmente basta que uma tabela atenda às 3 primeiras para que possa ser considerada como normalizada.

Primeira Forma Normal

Requer que a tabela não apresente dados repetitivos em sua estrutura.

Exemplo: Livros

a) Estrutura original

Tabela "Livros"

IdLivro	***Título***	***Assunto***	***Autor1***	***Autor2***	***Autor3***
21237	Os Sertões	Ficção	E. Cunha		
33455	Eletricidade básica	Física	A. Silva	B. Santos	
12312	Atlas do Brasil	Geografia	IBGE		

b) Estrutura normalizada (1FN)

Tabela "Livros"

IdLivro	***Título***	***Assunto***
21237	Os Sertões	Ficção
33455	Eletricidade básica	Física
12312	Atlas do Brasil	Geografia

Tabela "Autorias"

IdLivro	***Autor***
21237	E. Cunha
33455	A. Silva
33455	B. Santos
12312	IBGE

Segunda Forma Normal

Exige que a tabela não tenha atributos não-chave se referindo a apenas uma parte da chave primária (dependência funcional parcial).

Exemplo: Disciplinas

a) Estrutura original

Tabela "Disciplinas"

Discipl	***Curso***	***Fac***	***Prof***	***Cargo***	***CredSem***	***TotCred***
BD1	PD	PD	005	Assistente	4	72
TGA	ADM	ADM	004	Associado	4	72
TGA	ECON	ADM	004	Associado	3	54

b) Estrutura normalizada (2FN)

Tabela "Disciplinas"

Discipl	***Curso***	***Prof***	***Cargo***	***CredSem***	***TotCred***
BD1	PD	005	Assistente	4	72
TGA	ADM	004	Associado	4	72
TGA	ECON	004	Associado	3	54

Tabela "Cursos"

Curso	***Nome***	***Fac***	***Parecer***
PD	Processamento de Dados	PD	(...)
ADM	Administração de Empresas	ADM	(...)
ECON	Economia	ADM	(...)

Terceira Forma Normal

Para estar na terceira forma normal a tabela não pode ter atributos não-chave se referindo a outros atributos não-chave.

Exemplo: Disciplinas

a) Estrutura original

Tabela "Disciplinas"

Discipl	***Curso***	***Prof***	***Cargo***	***CredSem***	***TotCred***
BD1	PD	005	Assistente	4	72
TGA	ADM	004	Associado	4	72
TGA	ECON	004	Associado	3	54

b) Estrutura normalizada (3FN)

Tabela "Disciplinas"

Discipl	***Curso***	***Prof***	***CredSem***
BD1	PD	005	4
TGA	ADM	004	4
TGA	ECON	004	3

Tabela "Professores"

Prof	***Nome***	***Cargo***
005	João	Assistente
004	Pedro	Associado
008	Maria	Pleno

Roteiro para normalização de dados

Tabela não normalizada

↓

Desmembrar a tabela em uma ou mais tabelas
sem grupos repetitivos de itens.
Designar um ou mais atributos como Chave Primária das novas tabelas.

↓

Tabela na 1ª Forma Normal

↓

Verificar a existência de atributos parcialmente dependentes da Chave Primária.
Criar novas entidades, que absorverão os atributos com dependência funcional parcial, herdando a chave parcial.

↓

Tabela na 2ª Forma Normal

↓

Verificar se não existem atributos dependentes de outros atributos não-chave.
Destacar atributos dependentes de uma Chave Estrangeira e incorporar na tabela da chave estrangeira ou criá-las, se não existir.
Eliminar os atributos obtidos por cálculos a partir de outros atributos da mesma tabela.

↓

Tabela na 3ª Forma Normal

Referências bibliográficas

BARBIERI, Carlos. **Modelagem de Dados**. Rio de Janeiro: Infobook, 1994.

CHEN, Peter. **Modelagem de Dados: A abordagem Entidade - Relacionamento para projeto lógico**. São Paulo: Makron Books, 1990.

ELMASRI, Ramez & NAVATHE, Shamkant B. **Fundamentals of Database Systems**. 3th Ed. Reading: Addison Wesley, 2000.

HEUSER, Carlos Alberto. **Projeto de banco de dados**. Porto Alegre: Sagra Luzzatto, 1998.

KORTH, Henry F. & SILBERSCHATZ Abraham. **Sistema de Banco de Dados**. 2ª Ed. Rev. São Paulo: Makron Books, 1995.

MELO, Rubens N. et al. **Banco de Dados em Aplicações Cliente-Servidor**. Rio de Janeiro: Livraria e Editora InfoBook, 1997.

SHOVAL, Peretz. *One-to-One Dependencies in Database Design*. In: **IEEE Transactions on Knowledge and Data Engineering**, vol. 3, no. 3, September 1991, pp. 371-379.