



Aula 6 - Modelo relacional: tópicos avançados

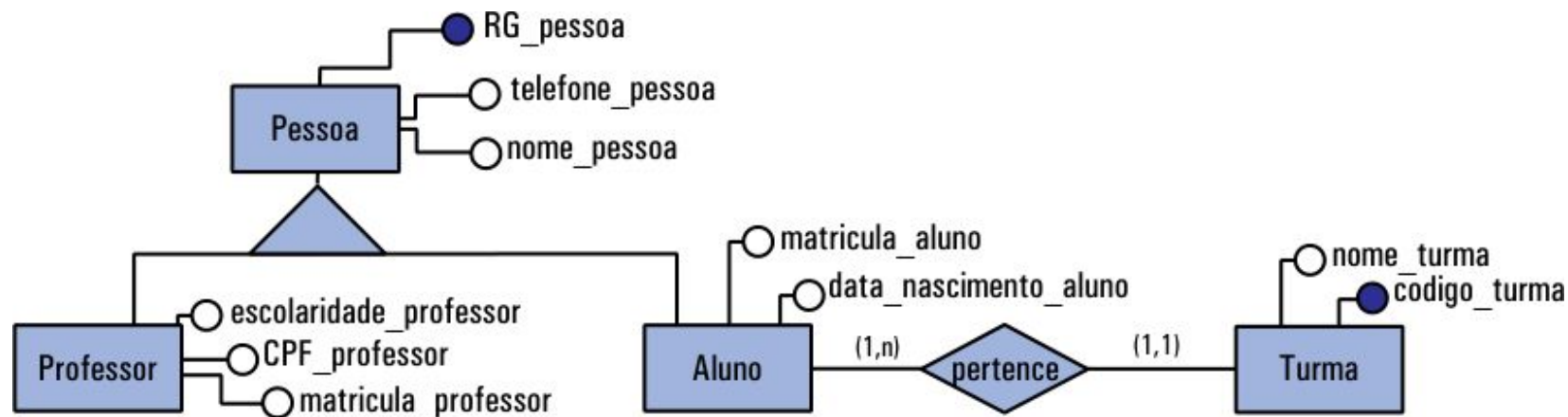
UC10 - Criar banco de dados

Especialização



A transformação de uma especialização do modelo de ER para o modelo relacional pode ser feita de 3 diferentes modos:

1. Criando uma tabela apenas para a entidade pai;
2. Criando tabelas apenas para as entidades filhas;
3. Criando uma tabela para cada entidade (tanto para a entidade pai, quanto para as filhas).





1. Criando uma tabela apenas para a entidade pai

Na primeira situação, será criada uma tabela única com o nome da entidade pai e essa tabela irá conter: todos os atributos das entidades pai (genérica), os atributos da(s) entidade(s) filha(s) (entidades especializadas), atributos referentes a possíveis relacionamentos e um atributo chamado “tipo” que identificará qual entidade especializada está sendo representada em uma linha. A chave primária dessa tabela será a própria chave primária da entidade pai.



1. Criando uma tabela apenas para a entidade pai

```
tbTurma(codigo_turma, nome_turma)
```

```
tbPessoa(RG_pessoa, telefone_pessoa, nome_pessoa, matricula_aluno,  
data_nascimento_aluno, codigo_turma, escolaridade_professor, CPF_professor,  
matricula_professor, tipo_pessoa)  
codigo_turma referencia tbTurma
```



1. Criando uma tabela apenas para a entidade pai

A primeira abordagem irá conter muitos valores nulos, uma vez que dado o tipo do objeto somente os atributos referentes àquele objeto serão preenchidos. Por isso, nem todos os atributos serão obrigatórios. Por outro lado, essa primeira abordagem tem a vantagem de dispensar a necessidade de junção entre tabelas, uma vez que os dados estão todos na mesma tabela.



2. Criando tabelas apenas para as entidades filhas

Na segunda situação, serão criadas tabelas apenas para as entidades filhas. Cada entidade filha que virar uma tabela terá como atributos tantos os seus atributos específicos e de seus relacionamentos diretos, quanto os atributos da entidade pai, mais os atributos dos relacionamentos de outras entidades com a entidade pai. A chave primária de cada uma das tabelas especializadas será a chave primária da entidade pai. As tabelas criadas serão completamente independentes umas das outras.



2. Criando tabelas apenas para as entidades filhas

```
tbTurma(codigo_turma, nome_turma)
```

```
tbAluno(RG_pessoa, nome_pessoa, telefone_pessoa, matricula_aluno,  
data_nascimento_aluno, codigo_turma)
```

```
codigo_turma referencia tbTurma
```

```
tbProfessor(RG_pessoa, nome_pessoa, telefone_pessoa, matricula_professor,  
CPF_professor, escolaridade_professor)
```



2. Criando tabelas apenas para as entidades filhas

A segunda abordagem é pouco recomendada, porque pode gerar redundância de dados, uma vez que os dados da entidade genérica são repetidos em todas tabelas especializadas. Assim, se uma pessoa for tanto professor como aluno, teremos as informações referentes a essa pessoa repetida nas duas tabelas. Portanto, essa abordagem só deve ser utilizada quando tivermos uma especialização exclusiva, ou seja, uma pessoa ou é do tipo aluno ou do tipo professor.



3. Criando uma tabela para cada entidade

Na terceira situação, serão criadas tabelas para todas as entidades (pai e filhas). Cada tabela terá seus atributos específicos, e os atributos dos seus relacionamentos. As tabelas referentes às entidades filhas também receberão como chave estrangeira a chave primária da entidade pai. A chave primária para cada entidade filha será a chave estrangeira, que neste caso terá as duas funções (PK e FK). Caso exista algum atributo que identifique unicamente a entidade filha, ele poderá ser escolhido como chave primária e a chave primária da entidade pai passa apenas como chave estrangeira.



3. Criando uma tabela para cada entidade

```
tbTurma(codigo_turma, nome_turma)
```

```
tbPessoa(RG_pessoa, nome_pessoa, telefone_pessoa)
```

```
tbProfessor(matricula_professor, RG_pessoa, CPF_professor, escolaridade_professor)  
RG_pessoa referencia tbPessoa
```


```
tbAluno(matricula_aluno, RG_pessoa, data_nascimento_aluno, codigo_turma)  
codigo_turma referencia tbTurma  
RG_pessoa referencia tbPessoa
```

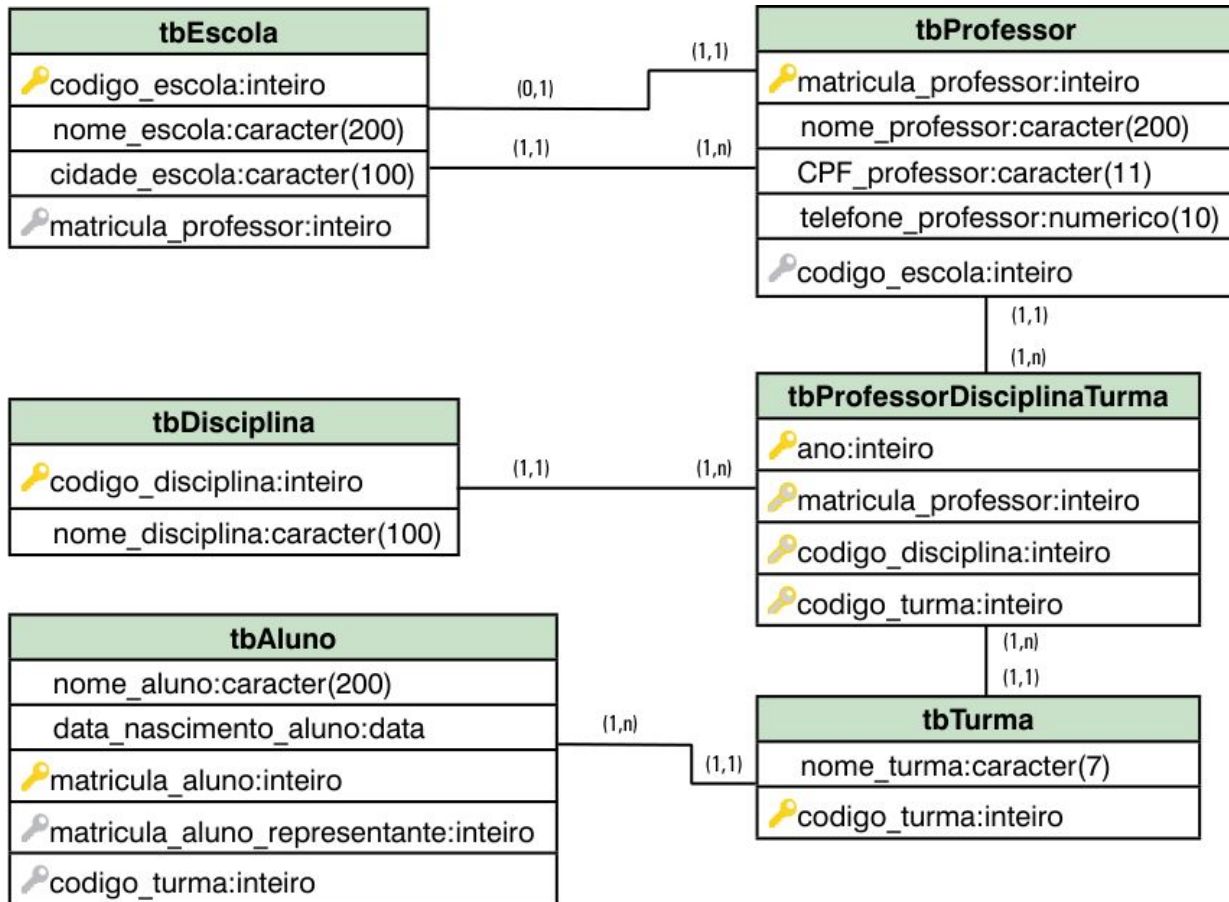



3. Criando uma tabela para cada entidade

A terceira abordagem tem a vantagem de evitar os valores nulos que aparecem na primeira abordagem e ainda a de não permitir a duplicidade como na segunda abordagem. Assim, o desenvolvedor da base de dados deve analisar todos os aspectos referentes à situação que se está modelando e optar pela solução que seja mais adequada ao problema.


Diagrama do modelo relacional

- 
- O modelo relacional pode ser descrito, como fizemos nos exemplos anteriores, ou pode ser diagramado (forma mais comum).
 - No Diagrama do Modelo Relacional, tudo que virou tabela, aplicando-se as regrinhas de conversão entre modelos, será representado por um retângulo. Esse retângulo irá conter o nome da tabela, seus atributos, os tipos dos atributos, a identificação da chave primária, a identificação da chave estrangeira e a cardinalidade do modelo.
 - A cardinalidade é atribuída considerando-se o Modelo de ER. Cardinalidade do tipo 1:1 e 1:N são representadas da mesma forma que no modelo de ER. Já a cardinalidade N:N não aparece no diagrama do modelo relacional, uma vez que o relacionamento N:N virou uma tabela. Portanto, todo relacionamento N:N dará origem a dois relacionamentos do tipo 1:N.





- 
- A chave primária nesse diagrama é indicada pela chave dourada, mas também poderia ser indicada grifando-se os atributos que formam a chave primária.
 - A chave estrangeira é indicada pela chave de cor prata, mas também poderia ser indicada pelo asterisco.
 - Aqueles atributos que são chaves estrangeiras e também ajudam a formar a chave primária, como mostrado na tabela `tbProfessorDisciplinaTurma`, são indicados pela chave dourada com o interior prateado.

Dicionário de dados da base de dados


- 
- Após o modelo relacional ter sido descrito ou diagramado, é necessário criar o Dicionário de Dados para a base de dados.
 - O Dicionário de Dados da Base de Dados tem por objetivo descrever as propriedades de uma tabela, sua estrutura física e as restrições que cada atributo possui. Assim, o desenvolvedor que irá implementar o banco de dados saberá exatamente como a base deve ser criada.
 - No Dicionário de Dados da Base de Dados, cada tabela do modelo relacional deverá ser descrita e deverá conter os seguintes campos: Nome do Atributo, Descrição do Atributo, Tamanho, Tipo e Restrições (Valor Nulo, Regra de Domínio, Chaves, Valor Default e Unique).


Nome	Descrição	Tipo	Tamanho	Nulo	Regra (check)	Chave	Default	Único
matricula_aluno	Armazena a matrícula do aluno	Númerico	5	Não	—	PK	—	Não
RG_aluno	Armazena o RG do aluno	Caracter	11	Não	—	—	—	Sim
nome_aluno	Armazena o nome do aluno	Caracter	100	Não	—	—	—	Não
data_nascimento_aluno	Armazena a data de nascimento do aluno	Data	—	Não	—	—	—	Não
cidade_aluno	Armazena a cidade em que o aluno mora	Caracter	20	Sim	—	—	Curitiba	Não
matricula_aluno_representante	Armazena a matrícula do aluno representante	Númerico	5	Sim	—	—	—	Não
codigo_turma	Armazena o código da turma do aluno	Inteiro	—	Não	—	FK que referencia tbTurma	—	Não
sexo_aluno	Armazena o sexo a que o aluno pertence	Caracter	1	Não	M – Masculino F – Feminino	—	—	Não


- 
- Para alguns tipos de dados, não é possível definirmos o tamanho, como por exemplo o tipo Data, porque esses tipos já têm tamanho pré-definido pelo SGBD. As restrições aplicadas a um atributo definem as propriedades desse atributo.
 - A restrição de Nulo define se um atributo permite ou não o valor nulo, ou seja, define se o atributo será obrigatório ou não.
 - Uma restrição de Domínio ou Regra de Domínio define quais valores serão permitidos cadastrar para um atributo. No exemplo, temos uma regra de domínio que diz que os valores permitidos para sexo são apenas “M” ou “F”.

- 
- As restrições de chave permitem identificar a chave primária (PK) e as chaves estrangeiras (FK). É interessante que na definição da chave estrangeira também seja identificado a que tabela ela referencia.
 - A restrição de default permite que seja inserido um valor padrão caso o usuário não digite nada para o campo. No nosso exemplo, definiu-se que se o usuário não digitar nada para o campo `cidade_aluno`, o próprio SGBD armazena o valor “Curitiba” para esse campo.
 - A última restrição é a de unicidade. Essa restrição é aplicada apenas para atributos que não são chave primária e que não podem se repetir. No exemplo, o atributo `RG_aluno` não é chave primária e não pode se repetir. Sendo assim, pode-se definir o atributo como `unique` (único). É redundante dizer que uma chave primária é `unique`, já que ela não se repete.

Normalização

- 
- O processo de normalização geralmente é aplicado quando temos uma base de dados que foi criada antes da existência de um banco de dados relacional ou foi desenvolvida sem considerar a existência de um banco de dados relacional.
 - Na maioria dos casos, esses sistemas são antigos e seus arquivos de dados possuem muitas informações redundantes e inconsistentes. Além disso, normalmente não existe nenhum documento que especifique o modelo de dados, o que torna muito difícil a manutenção desses sistemas ou as migrações para um banco de dados relacional.

- 
- Às vezes, também encontramos sistemas não muito antigos e que, no entanto, foram implementados usando uma única tabela. Nesses casos, as informações estarão repetidas, haverá muitos valores nulos na base e será muito difícil executar consultas que retornem valores consistentes com o esperado.
 - Para resolver ou minimizar os problemas apresentados, pode-se fazer o que chamamos de engenharia reversa do projeto, utilizando para isso o processo de normalização. A engenharia reversa do projeto consiste em, a partir dos dados armazenados, obter um modelo conceitual da base de dados e eliminar as redundâncias.

- 
- Também podemos utilizar o processo de normalização para conferir se o nosso modelo de dados está normalizado e, caso não esteja, pode-se normalizá-lo antes da implementação da base de dados no SGBD.
 - O processo de normalização consiste em um conjunto de regras, denominadas formas normais. A literatura apresenta 6 formas normais: 1FN, 2FN, 3FN, 4FN, 5FN e a de Boyce/Codd. No entanto, para a maioria das bases de dados, a aplicação até a 3FN é suficiente. Sendo assim, só iremos abordar as três primeiras formas normais.

Atributo multivalorado

```
tbPedido(codigo_pedido, valor_total, data_pedido, (telefone_contato),  
(codigo_produto, nome_produto, preco_unitário, quantidade, valor_pago_por_produto))
```

Grupo de valores repetidos

O modelo relacional acima poderia ser escrito como mostra a tabela:

Código pedido	Valor Pedido	Data pedido	Telefone de contato	Produto				
				Codigo produto	Nome produto	Valor unitário do produto	Quantidade	Valor pago por produto
100	3300,00	30/11/2009	2222-2222 9999-9999	1	Computador	1500,00	1	1500,00
				5	Impressora	600,00	2	1200,00
				6	Papel A4	12,00	50	600,00
101	3800,00	15/12/2009	2121-2121 9191-9191	2	Mouse	30,00	10	300,00
				5	Impressora	600,00	5	3000,00
				7	Teclado	50,00	10	500,00



Primeira forma normal (1FN)

- Toda tabela está na 1FN se os seus atributos forem atômicos. Isso significa que não serão permitidos atributos compostos, multivalorados ou grupos repetidos de dados (também conhecidos como tabelas aninhadas).
- Os grupos repetidos de dados ocorrem quando uma tabela aparece dentro de outra tabela. Por exemplo, na tabela `tbPedido` da figura, temos uma única tabela que armazena os dados de pedidos e de produtos. Assim, os dados referentes ao produto pertencem a uma tabela que está dentro da tabela pedido.
- É possível perceber que a tabela `tbPedido` não está na 1FN porque além de grupos repetidos ela também tem um atributo multivalorado.



Primeira forma normal (1FN)

Para deixar uma tabela na 1FN, é necessário fazer o seguinte:

- Os atributos compostos devem ser decompostos e armazenados como atributos simples.
- Para cada atributo multivalorado será criada uma tabela que irá conter o atributo multivalorado mais a chave primária da tabela inicial, que passa como chave estrangeira. A chave primária da nova tabela será composta.
- Para cada grupo repetido será criada uma tabela que irá conter os atributos do grupo repetido mais a chave primária da tabela inicial, que passa como chave estrangeira e irá ajudar a compor a chave primária.



Primeira forma normal (1FN)

tbPedido(codigo_pedido, valor_total, data_pedido)

tbTelefoneContatoPedido(codigo_pedido, telefone_contato)
codigo_pedido referencia tbPedido

tbPedidoProduto(codigo_pedido, codigo_produto, nome_produto, preco_unitario,
quantidade, valor_pago_por_produto)
codigo_pedido referencia tbPedido



Segunda forma normal (2FN)

- A 2FN só é aplicável para tabelas que possuem uma chave primária composta e que, além disso, tenham outros atributos que não façam parte da chave primária.
- Uma tabela está na 2FN se estiver na 1FN e todo atributo que não compõe a chave primária deve ter **dependência funcional total** em relação à chave primária.
- Para que a **dependência funcional** seja **total**, o atributo não chave deve depender de toda a chave primária composta. Por exemplo, no caso da tabela tbPedidoProduto o atributo “quantidade” representa a quantidade de um produto que foi solicitada em um pedido. Assim, o atributo “quantidade” depende tanto do `codigo_produto` quanto do `codigo_pedido`, caracterizando uma dependência funcional total. O atributo `nome_produto`, por outro lado, tem uma dependência funcional parcial, porque depende apenas de parte da chave primária, ou seja, depende apenas do `codigo_produto`.



Segunda forma normal (2FN)

Para deixarmos o modelo na 2FN devemos:

- Repetir as tabelas que tenham chave primária simples, pois já estão na 2FN.
- Repetir as tabelas que tenham chave primária composta, mas que não tenham nenhum outro atributo além dos que compõem a chave primária.
- Para cada tabela que tiver chave primária composta e pelo menos um atributo que não faz parte da chave, deve-se verificar se cada um dos atributos não chave têm dependência funcional total. Caso a dependência não seja total, deve-se criar uma tabela com o atributo que depende parcialmente, mais o atributo do qual ele depende (que será chave primária na nova tabela e será chave estrangeira na tabela inicial).



Segunda forma normal (2FN)

tbPedido(codigo_pedido, valor_total, data_pedido)

tbTelefoneContatoPedido(codigo_pedido, telefone_contato)
codigo_pedido referencia tbPedido

tbPedidoProduto(codigo_pedido, codigo_produto, quantidade, valor_pago_por_produto)
codigo_pedido referencia tbPedido
codigo_produto referencia tbProduto

tbProduto(codigo_produto, nome_produto, preco_unitario)



Terceira forma normal (3FN)

- Uma tabela está na 3FN se estiver na 2FN e ela não possuir **dependências transitivas** (ocorre quando existe um atributo que não é chave e nem faz parte da chave, mas que identifica outros atributos).

tbDepartamento(codigo_depto, nome_depto, codigo_gerente, nome_gerente)

- A tabela acima está na 1FN porque todos os seus atributos são atômicos e está na 2FN porque não tem chave primária composta. No entanto, temos uma dependência transitiva que ocorre com o atributo nome_gerente que depende do atributo codigo_gerente que não é chave e nem faz parte da chave primária.



Terceira forma normal (3FN)

- Para eliminar dependências transitivas, deve-se criar uma nova tabela que irá conter o atributo que depende (ex.: nome_gerente) mais o atributo do qual ele é dependente (ex.: codigo_gerente). As tabelas abaixo mostram como ficaria o modelo.

tbDepartamento(codigo_depto, nome_depto, codigo_gerente)
codigo_gerente referencia tbGerente

tbGerente(codigo_gerente, nome_gerente)