



Cursos

➤ Listagem de disciplinas

Selecione uma disciplina

Aulas

- 01 Introdu  o a Banco de Dados
- 02 Modelo de Entidade e Relacionamento
- 03 Modelo Relacional
- 04 Transforma  es ER para MR
- 05 Transforma  es ER para MR e dicion rio de dados
- 06 Normaliza  o b sica
- 07 Normaliza  o avan ada
- 08 Introdu  o   Linguagem SQL e Sistemas Gerenciadores de Banco de Dados
- 09 Linguagem SQL - cria  o, inser  o e modifica  o de tabelas
- 10 Linguagem SQL - Consulta simples de tabelas
- 11 Linguagem SQL - Consulta avan ada de tabelas
- 12 Linguagem SQL - Altera  o da estrutura de tabelas e ambientes de m ltiplas tabelas
- 13 Linguagem SQL - Subconsultas
- 14 Linguagem SQL - VIS  ES
- 15 Linguagem SQL - STORED PROCEDURES
- 16 Linguagem SQL - Fun  es
- 17 Linguagem SQL - Seguran a
- 18 Engenharia Reversa
- 19 Utilizando SQL em Java
- 20 Utilizando conceitos avan ados de SQL em Java

⬅ Voltar 🖨 Imprimir ⬆ Topo

Sistemas de Banco de Dados

Aula 6 – Normaliza  o b sica

Professores autores

N lio Alessandro Azevedo Cacho (neliocacho@ect.ufrn.br)
Xiankleber Cavalcante Benjamim (xianklebercb@gmail.com)

Apresenta  o

Nas aulas anteriores voc  aprendeu como modelar seu banco de dados utilizando o modelo Entidade Relacionamento - ER e o modelo Relacional. Voc  j  aprendeu tamb m como fazer o mapeamento dos conceitos do modelo ER para o Modelo Relacional. Apesar de tudo isso, voc  pode estar se perguntando: como saber se o meu modelo Relacional foi bem feito? Nesta aula, iremos apresentar algumas t cnicas b sicas de como voc  pode verificar se seu modelo relacional foi bem definido ou n o. O conjunto destas t cnicas recebe o nome de normaliza  o. Nesta aula iremos mostrar como utilizar duas formas de normaliza  o.

Objetivo

Ao final desta aula, você será capaz de:

- Definir o que são anomalias de inserção, remoção e atualização;
- Normalizar suas tabelas segundo a Primeira Forma Normal (1FN);
- Normalizar suas tabelas segundo a Segunda Forma Normal (2FN).

Anomalia de inserção, remoção e atualização

Antes de falar sobre normalização, vamos entender o problema de anomalia de inserção, remoção e atualização. Anomalias são mudanças em dados que podem gerar uma inconsistência no banco de dados relacional. Tal inconsistência geralmente aparece quando o banco de dados é projetado de forma inadequada. Para exemplificar, usaremos a Figura 1 para mostrar os tipos mais comuns de anomalias. Note que na Figura 1 mostramos na parte superior a estrutura da tabela, como você está acostumado a ver, e também, na parte inferior da figura, exemplos de valores para os registros. Os valores são importantes para facilitar o entendimento dos cenários que produzem as anomalias.

Assim, na Figura 1 temos uma tabela chamada AgenciaFuncionario que armazena os dados dos funcionários de um banco e também os dados sobre as agências. Como essas duas informações (Funcionário e Agência) são armazenadas na mesma tabela, é possível também saber onde cada funcionário trabalha (agência, endereço, telefone). Ou seja, à primeira vista, a tabela AgenciaFuncionario parece ser uma ótima opção para reduzir o número de tabelas e aumentar a velocidade do sistema. Entretanto, tal solução pode gerar várias anomalias.

FuncN	Nome	Cargo	Salario	NumAg	Endereco	Telefone
01	Luiz	Gerente	5000	1524	Prudente de Moraes, 12, RN	3605-5223
02	José	Secretario	1500	1524	Prudente de Moraes, 12, RN	3605-5223
03	João	Gerente	6000	1550	Hermes da Fonseca, 15, RN	4225-5889
04	Luiz	Secretario	1800	1550	Hermes da Fonseca, 15, RN	4225-5889
05	Irene	Gerente	7000	2051	Eng. Roberto Freire, 20, RN	5223-8556

FuncN	Nome	Cargo	Salário	NumAg	Endereço	Tel
01	Luiz	Gerente	5000	1524	Prudente de Moraes, 12, RN	3605-5223
02	José	Secretario	1500	1524	Prudente de Moraes, 12, RN	3605-5223
03	João	Gerente	6000	1550	Hermes da Fonseca, 15, RN	4225-5889
04	Luiz	Secretario	1800	1550	Hermes da Fonseca, 15, RN	4225-5889
05	Irene	Gerente	7000	2051	Eng. Roberto Freire, 20, RN	5223-8556

Figura 1 - Exemplo de tabela AgenciaFuncionario
Fonte: Connolly e Begg (2000).

Anomalia de inserção

Uma anomalia de inserção acontece quando, ao inserir um dado, este dado pode gerar uma inconsistência no banco de dados. No exemplo da Figura 1, para inserir os detalhes (NumAg, Endereço, Tel.) de um novo funcionário, você deve tomar cuidado para usar exatamente os dados já cadastrados por outros funcionários. Por exemplo, um novo funcionário para a agência 1550 deve usar exatamente o mesmo endereço dos outros dois funcionários que também trabalham nesta agência. Se isto não for feito, teremos um problema de inconsistência de dados, onde dois funcionários que trabalham na mesma agência possuem endereços diferentes.

Anomalia de remoção

Uma anomalia de remoção acontece quando, ao remover um registro, você pode gerar inconsistência no banco de dados. No exemplo da Figura 1, uma anomalia de remoção acontece quando removemos o funcionário de número 05. Neste caso, o objetivo é apenas remover os dados do funcionário e preservar os dados da agência 2051. Entretanto, da forma como a tabela está estruturada, os dados da agência também são removidos.

Anomalia de atualização

Uma anomalia de atualização acontece quando, ao atualizar um registro, você pode gerar inconsistência no banco de dados. No exemplo da Figura 1, uma anomalia de atualização acontece quando modificamos o endereço da agência 1524. Neste caso, teremos que atualizar o endereço de todos os funcionários da agência 1524. Caso contrário, teremos uma inconsistência no banco de dados onde funcionários da mesma agência possuem endereços diferentes.

Note que todas as anomalias acontecem devido à existência de redundância de informação na tabela AgenciaFuncionario. Por exemplo, o mesmo endereço de uma agência é armazenado várias vezes quando ele poderia ser armazenado apenas uma vez. Assim, para evitar as anomalias é preciso evitar a redundância. A redundância é evitada através da normalização das tabelas que você verá logo a seguir.

Normalização

O processo de normalização foi proposto por Dr. E. F. Codd como uma forma de evitar as anomalias mostradas anteriormente. Assim, o objetivo da normalização é remover a duplicação de dados e, conseqüentemente, minimizar a redundância. Segundo [Powell], a remoção da duplicação de dados permite:

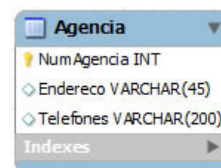
- reduzir o espaço físico necessário para armazenar o banco de dados;
- melhorar a organização dos dados;
- reduzir o impacto de mudanças, inserções e remoções nos dados do banco de dados.

O processo de normalização é constituído por um conjunto de formas normais. As formas normais especificam critérios que definem quando uma tabela está bem estruturada ou não. Assim, para saber se uma tabela está bem estruturada, você deve verificar se a estrutura da tabela satisfaz todas as formas normais. Nesta aula você verá duas formas normais que são bem definidas na literatura de banco de dados. Para cada forma normal, iremos ver também quais ações devem ser tomadas para adequar uma tabela a uma forma normal. Ou seja, o que fazer para concertar a estrutura de uma tabela de modo que a mesma satisfaça a forma normal.

Primeira Forma Normal (1FN)

Uma tabela está na Primeira Forma Normal (1FN) se e somente se todos os atributos contiverem apenas dados atômicos. Ou seja, cada atributo pode ter apenas um valor por registro (tupla).

A Figura 2 mostra um exemplo de uma tabela que não está na 1FN. Esta tabela não está na 1FN porque o atributo Telefones possui mais de um telefone por registro (tupla). Por exemplo, a agência 1524 possui três telefones.



NumAg	Endereço	Telefones
1524	Prudente de Moraes, 12, RN	3605-5223, 3605-5141, 3605-5142
1550	Hermes da Fonseca, 15, RN	4225-5889, 4225-5890
2051	Eng. Roberto Freire, 20, RN	5223-8556, 5223-8557

Figura 2 - Exemplo de estrutura de tabela que não está na 1FN
Fonte: Connolly e Begg (2000).

Para adequar uma tabela que não está na 1FN é necessário realizar os seguintes passos:

- 1 – criar uma tabela para conter os dados do atributo não atômico;
- 2 – criar na nova tabela um atributo para conter o atributo não atômico da tabela original;
- 3 – criar na nova tabela um atributo para conter a chave primária da tabela original;
- 4 – definir uma chave estrangeira para garantir a relação entre a nova tabela e a tabela original;
- 5 – definir a chave primária da nova tabela;
- 6 – remover o atributo não atômico da tabela original.

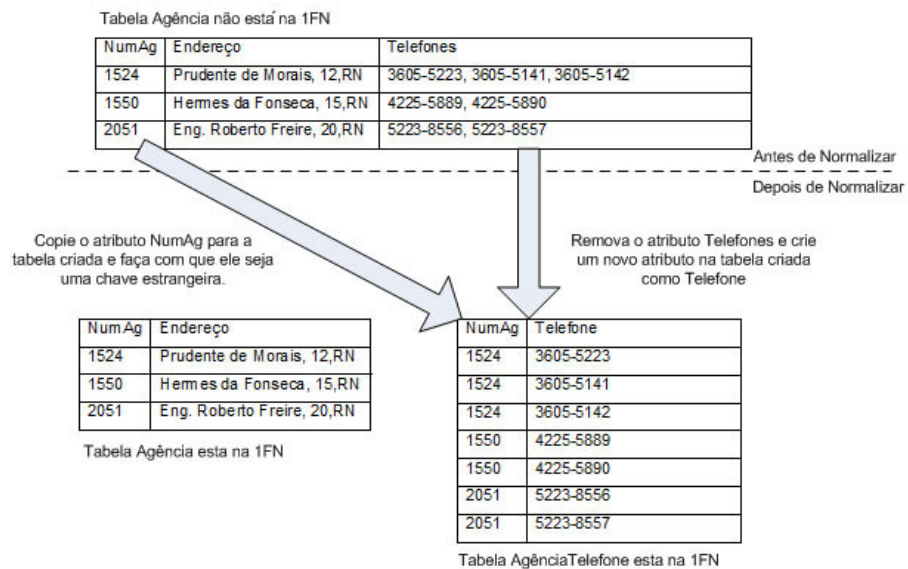


Figura 3 - Normalização da Tabela Agência
Fonte: Connolly e Begg (2000).

Os passos listados anteriormente são ilustrados na Figura 3 utilizando-se o exemplo da tabela Agência. O primeiro passo é criar uma nova tabela chamada AgenciaTelefone. Depois você cria na tabela AgenciaTelefone o atributo não atômico da tabela Agência, no nosso caso, o atributo Telefone. Depois você cria na tabela AgenciaTelefone o atributo chave da tabela Agência, no nosso caso, o atributo NumAg. Para manter a integridade com a tabela original, você deve definir uma chave estrangeira entre o atributo NumAg da tabela AgenciaTelefone e NumAg da tabela Agência. Depois você deve definir a chave primária da tabela AgenciaTelefone. No nosso caso, como não podemos ter Agências diferentes com o mesmo telefone, definimos como chave primária da tabela AgenciaTelefone o atributo Telefone. Finalmente, removemos o atributo não atômico da tabela Agência.

Para perceber a diferença, veja na Figura 4 a nova estrutura das duas tabelas. Note que o processo de normalização de uma tabela para a 1FN é equivalente ao mapeamento de um atributo multivalorado do modelo ER para o modelo relacional, mostrado na Aula 4. Ou seja, se você definir bem seu modelo ER e fizer o mapeamento correto, suas tabelas já estarão na 1FN.

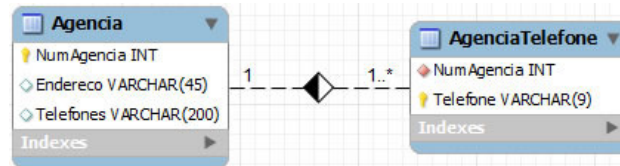


Figura 4 - Estrutura das tabelas depois da Normalização

Pratique 01

1. Descreva como a Normalização pode ser utilizada no projeto de banco de dados.
2. Descreva as características das tabelas que violam a 1FN e como tais tabelas podem ser modificadas para satisfazer a 1FN.

Segunda Forma Normal (2FN)

Uma tabela está na Segunda Forma Normal (2FN) se e somente se ela estiver na 1FN e todos os atributos não chave primária puderem ser obtidos da combinação de todos os atributos que formam a chave primária.

Para ilustrar o significado da 2FN, vamos olhar para a Figura 5, adaptada de [Connolly, Begg]. Nesta figura, a tabela Alocação armazena as horas trabalhadas por funcionários temporários em determinadas agências de um banco. Na parte superior da Figura 5 é mostrada a estrutura da tabela, onde é possível identificar que a sua chave primária é formada pelos atributos NumEmp e NumAg. Além destes dois atributos, a tabela Alocação possui mais três atributos que não fazem parte da chave primária. São eles: AgEnd, NomeEmp e horasSem.

Como você viu anteriormente, para estar na 2FN todos estes três atributos não chave (AgEnd, NomeEmp e horasSem) devem ser obtidos através dos **dois** atributos chaves (NumEmp e NumAg). Se for possível obter um atributo não chave primária através de apenas um dos atributos chave primária, então a tabela não está na 2FN.

Por exemplo, eu consigo saber o endereço da agência (atributo AgEnd) apenas através do atributo NumAg? A resposta é sim. O endereço da agência é uma informação intrínseca à agência e pode ser obtido através do atributo NumAg. Outro exemplo: eu consigo obter o nome do empregado (NomeEmp) através do código do empregado (NumEmp)? Sim, esta informação é intrínseca ao empregado e, através do atributo NumEmp, eu poderia obter seu nome. Finalmente, eu poderia obter as horas semanais trabalhadas pelo empregado apenas através de seu NumEmp? Não, isso acontece porque as horas semanais de cada empregado dependem da agência onde ele trabalha. Assim, o atributo horasSem depende dos dois atributos NumEmp e NumAg que compõem a chave primária para ser obtido.

Para concluir o exemplo, temos que a tabela Alocação mostrada na Figura 5 **não** está na 2FN porque possui pelo menos um atributo que pode ser obtido de apenas um dos atributos que formam a chave primária. Este é o caso do atributo NomeEmp, que pode ser obtido apenas de NumEmp, e um outro exemplo é o caso do atributo AgEnd, que pode ser obtido apenas do atributo NumAg.

O fato de que uma tabela não está na 2FN pode gerar as anomalias mostrados anteriormente. Por exemplo, para modificar o endereço da agência de número 1550 você teria que modificar dois registros: o segundo e quarto registro na tabela da Figura 5.

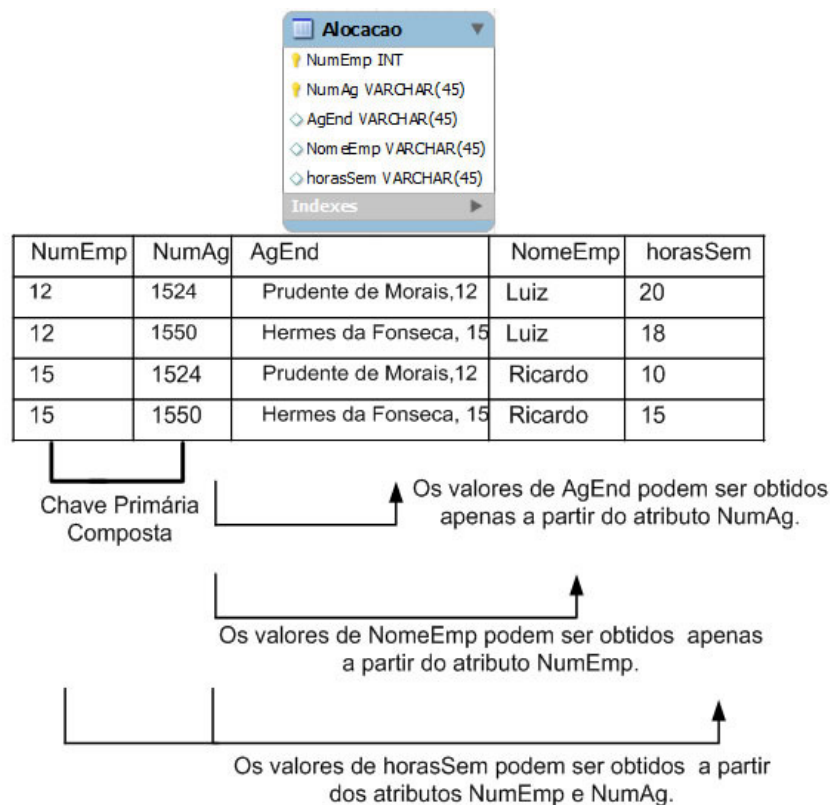


Figura 5 - Exemplo de tabela que não está na 2FN.
Fonte: Connolly e Begg (2000).

Bem, depois de aprender como identificar se uma tabela está ou não na 2FN, agora vamos aprender com re-estruturar uma tabela de modo que ela fique na 2FN. Para adequar uma tabela que não está na 2FN é necessário fazer os seguintes passos:

- 1 - criar duas novas tabelas para armazenar os dados dos campos redundantes, onde seus valores apresentam repetição de valores;
- 2 - remover os campos com valores redundantes da tabela original;
- 3 - criar chaves primárias nas novas tabelas criadas com base na chave primária da tabela original;
- 4 - criar relações um-para-muitos entre as novas tabelas criadas e a tabela original.

Os passos listados anteriormente são ilustrados na Figura 6, utilizando-se o exemplo da tabela Alocação. A primeira coisa a fazer é criar as tabelas Empregado e Agencia. Estas tabelas irão armazenar os dados intrínsecos de Empregado e Agencia que, na Figura 5, estavam na tabela Alocação.

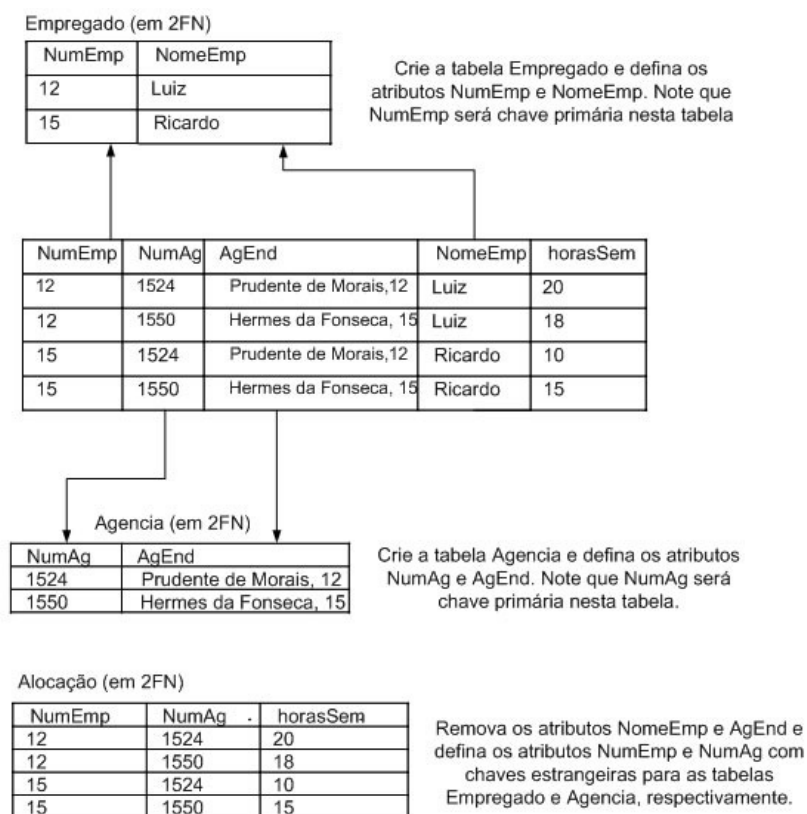


Figura 6 - Exemplo de normalização da tabela Alocação para a 2FN.
Fonte: Connolly e Begg (2000).

Depois disto, você deve definir as chaves primárias das tabelas Empregado e Agencia. Para saber qual campo será a chave primária das tabelas criadas, você deve olhar para a chave primária da tabela Alocação e fazer a pergunta: qual o campo da chave primária da tabela Alocação está relacionada ao Empregado? Qual está relacionada a Agencia? A resposta para estas perguntas é, respectivamente, NumEmp e NumAg. Assim, NumEmp será a chave primária da tabela Empregado e NumAg será a chave primária da tabela Agencia.

Em seguida, você deve inserir nas tabelas criadas os seus atributos relacionados. Por exemplo, NomeEmp vai para a tabela Empregado e AgEnd vai para a tabela Agencia. O próximo passo é remover os atributos NomeEmp e AgEnd da tabela Alocação. Note que na tabela Alocação só devem ficar os atributos que só podem ser obtidos pela combinação dos atributos que formam a chave primária. Assim, a tabela Alocação só ficará com os atributos que formam a chave primária (NumEmp e NumAg) e o atributo horasSem. Para finalizar, você deve definir os atributos NumEmp e NumAg como chaves estrangeiras para as tabelas Empregado e Agencia, respectivamente. Finalmente, a Figura 7 mostra a estrutura das tabelas que estão na 2FN.

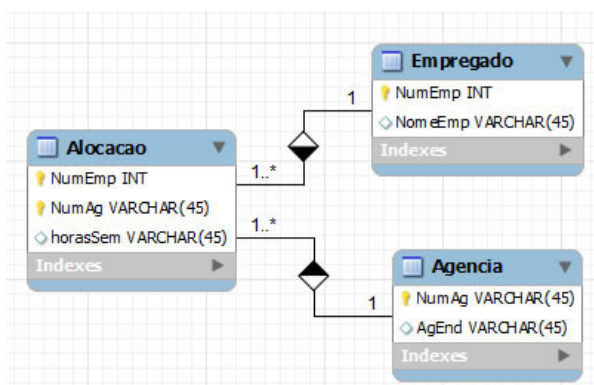
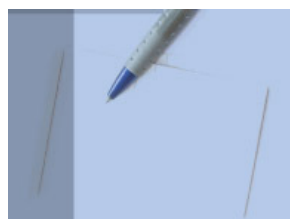


Figura 7 - Exemplo de tabelas que estão na 2FN





1. Descreva as características das tabelas que violam a 2FN e como tais tabelas podem ser modificadas para satisfazer a 2FN.
2. Verifique se a Tabela Agenda (Figura 8) está na 2FN. Caso ela não esteja, faça o processo de adequação como mostrado anteriormente.


Agenda	
idDentista	INT
idPaciente	VARCHAR(45)
NomeDentista	VARCHAR(45)
NomePaciente	VARCHAR(45)
TelefonePaciente	VARCHAR(45)
EnderecoPaciente	VARCHAR(45)
LocalAtendimento	VARCHAR(45)
DataAtendimento	DATETIME
Indexes	

Figura 8 - Tabela Agenda

Agora você já sabe como evitar algumas anomalias através de um processo de normalização. Tente praticar as regras e aplicar nos próximos projetos de bancos de dados que você fizer.

Resumo

Nesta sexta aula, você aprendeu que uma tabela incorretamente definida pode gerar anomalias de inserção, modificação e remoção. A existência de anomalias pode gerar inconsistência no banco de dados e comprometer sua utilização. No sentido de eliminar as anomalias, você aprendeu como normalizar tabelas através da 1FN e 2FN. Na próxima aula iremos ver formas avançadas de normalização.



Evolucao

Agora vamos relembrar e fixar nossa aula.

1. Explique o que é redundância de dados.
2. Cite as principais características das anomalias de inserção, remoção e atualização.
3. Dada as tabelas OrdemCompra e ItemCompra (Figura 9), verifique se estas tabelas estão na 2FN. Caso não estejam, faça o processo de adequação destas tabelas.

Ordem Compra	
idOrdem Compra	INT
Data	DATE
NomeCliente	VARCHAR(45)
EnderecoCliente	VARCHAR(200)
TelefoneCliente	VARCHAR(20)
TotalCompra	DECIMAL(2)
TotalImposto	DECIMAL(2)
ValorTotalCompra	DECIMAL(2)
Indexes	

ItemCompra	
idOrdem Compra	INT
idProduto	INT
DescricaoProduto	VARCHAR(45)
QuantidadeProduto	INT
PrecoProduto	DECIMAL(2)
CidadeDoProduto	VARCHAR(45)
Indexes	

Diagrama de relacionamento: 1 (Ordem Compra) para 1..* (ItemCompra)

Figura 9 - Tabelas OrdemCompra e ItemCompra

Referencias

CONNOLLY, Thomas M.; BEGG, Carolyn E. **Database Solutions**: a step-by-step approach to building databases. 2nd ed. New Jersey: Pearson Education Limited, 2000.

DATE, Christopher J. **Introdução a sistemas de banco de dados**. Rio de Janeiro: Campus, 2000.

DATE, C. J. **Introduction to Database Systems**. 7th ed. 1999.

HEUSER, C. A. **Projeto de banco de dados**. 6. ed. Porto Alegre: Editora Bookman, 2009.

_____. **Projeto de banco de dados**. 5. ed. Porto Alegre: Sagra-Luzzatto, 2004.

POWELL, Gavin. **Beginning Database Design**. San Francisco: Wiley Publishing, 2006.

[Voltar](#)[Imprimir](#)[Topo](#)